G C

RCarson10

Abstract

This project implements a solution for a Dynamical System using the code provided as well as solving queuing theory problems.

Project 7- Code Errors and the Butterfly Effect

By Gaj Carson

# **Abstract**

**Queuing theory: Definition, history & real-life applications**

Queuing theory is a powerful tool to analyze the daily phenomenon of waiting in line. Queuing theory (or queueing theory) refers to the mathematical study of the formation, function, and congestion of waiting lines, or queues.

At its core, a queuing situation involves two parts.

1. Someone or something that requests a service—usually referred to as the customer, job, or request.
2. Someone or something that completes or delivers the services—usually referred to as the server.

To illustrate, let’s take two examples. When looking at the queuing situation at a bank, the customers are people seeking to deposit or withdraw money, and the servers are the bank tellers. When looking at the queuing situation of a printer, the customers are the requests that have been sent to the printer, and the server is the printer.

Queuing theory uses the [Kendall notation](https://en.wikipedia.org/wiki/Kendall%27s_notation) to classify the different types of queuing systems, or nodes. Queuing nodes are classified using the notation A/S/c/K/N/D where:

* A is the arrival process
* S is the mathematical distribution of the service time
* c is the number of servers
* K is the capacity of the queue, omitted if unlimited
* N is the number of possible customers, omitted if unlimited
* D is the queuing discipline, assumed first-in-first-out if omitted

For example, think of an ATM. It can serve: one customer at a time; in a first-in-first-out order; with a randomly-distributed arrival process and service distribution time; unlimited queue capacity; and unlimited number of possible customers. Queuing theory would describe this system as a M/M/1 queue (“M” here stands for [Markovian](https://en.wikipedia.org/wiki/Markov_chain), a statistical process to describe randomness). Queuing theory calculators [out there](https://www.supositorio.com/rcalc/rcalclite.htm) often require choosing a queuing system from the Kendall notation before calculating inputs. Waiting in line is a part of everyday life because as a process it has several important functions. Queues are a fair and essential way of dealing with the flow of customers when there are limited resources. Negative outcomes arise if a queue process isn’t established to deal with overcapacity. Queuing theory is important because it helps describe features of the queue, like average wait time, and provides the tools for optimizing queues. From a business sense, queuing theory informs the construction of efficient and cost-effective workflow systems.

### Little’s Law

Little’s Law connects the capacity of a queuing system, the average time spent in the system, and the average arrival rate into the system without knowing any other features of the queue. The formula is quite simple and is written as follows:

[](https://www.codecogs.com/eqnedit.php?latex=\fn_jvn&space;L=\lambda&space;W)

or transformed to solve for the other two variables so that:

[](https://www.codecogs.com/eqnedit.php?latex=\fn_jvn&space;\lambda=\frac%7bL%7d%7bW%7d)

[](https://www.codecogs.com/eqnedit.php?latex=\fn_jvn&space;W=\frac%7bL%7d%7b\lambda%7d)

Where:

* L is the average number of customers in the system
* λ (lambda) is the average arrival rate into the system
* W is the average amount of time spent in the system

Project management processes like Lean and Kanban [wouldn’t exist](https://www.process.st/littles-law/) without Little’s Law.

**Dynamical Systems**

A [dynamical system](https://mathinsight.org/definition/dynamical_system) is all about the evolution of something over time. To create a dynamical system we simply need to decide [(1)](https://mathinsight.org/dynamical_system_idea#statespace) what is the “something” that will evolve over time and [(2)](https://mathinsight.org/dynamical_system_idea#therule) what is the rule that specifies how that something evolves with time. In this way, a dynamical system is simply a model describing the temporal evolution of a system.

**The state spaces**

The first step in creating a dynamical system is to pin down what is the special “something” that we want to evolve with time. To do this, we need to come up with a set of variables that give a complete description of the system at any particular time. By “complete description,” we don't necessarily mean that the variables will completely describe a real-life system we may be trying to model. But the variables must completely describe the state of the mathematical system. In a dynamical system, if we know the values of these variables at a particular time, we know everything about the state of the system at that time. To model some real-life system, the modeler must clearly make a choice of what variables will form the complete description for the mathematical model. The variables that completely describe the state of the dynamical system are called the [*state variables*](https://mathinsight.org/definition/state_variable). The set of all the possible values of the state variables is the [state space](https://mathinsight.org/definition/state_space).

The state space can be discrete, consisting of isolated points, such as if the state variables could only take on integer values. It could be continuous, consisting of a smooth set of points, such as if the state variables could take on any real value. In the case where the state space is continuous and finite-dimensional, it is often called the [phase space](https://mathinsight.org/definition/phase_space), and the number of state variables is the dimension of the dynamical system. The state space can also be infinite-dimensional.

**The time evolution rule**

The second step in creating a dynamical system is to specify the rule for the time evolution of the dynamical system. This rule must be defined to make the state variables be a complete description the state of the system in the following sense: the value of the state variables at a particular time must completely determine the evolution to all future states. If the time evolution depends on a variable not included in the state space, then the rule combined with the state space does not specify a dynamical system. One must either change the rule or augment the state space by the necessary variables to form a dynamical system.

The time evolution rule could involve discrete or continuous time. If the time is discrete, then the system evolves in time steps, and we usually let the time points be the integers t=0,1, 2,… . We can write the state of the system at time  as . In many cases, the time evolution rule will be based on a [function](https://mathinsight.org/definition/function) ff that takes as its input the state of the system at one time and gives as its output the state of the system at the next time. Therefore, starting at the initial conditions x0x0 at time t=0t=0, we can apply the function once to determine the state

 at time , apply the function a second time to get the state  at time , and continue repeatedly applying the function to determine all future states. We end up with a sequence of states, the [*trajectory*](https://mathinsight.org/definition/trajectory_dynamical_systems) of the point  In this way, the state at all times is determined both by the function ff and the initial state . We refer to such as system as a [*discrete dynamical system*](https://mathinsight.org/definition/discrete_dynamical_system).

In a [*continuous dynamical system*](https://mathinsight.org/definition/continuous_dynamical_system), on the other hand, the state of the system evolves through continuous time. One can think of the state of the system as flowing smoothly through state space. As time evolves, the state  at time can be thought of as a point that moves through the state space. The evolution rule will specify how this point x(t)x(t) moves by giving its velocity, such as through a function , where  is the velocity of the point at time . In this case, starting with an initial state  at time , the [*trajectory*](https://mathinsight.org/definition/trajectory_dynamical_systems) of all future times  will be a curve through state space.

**Classifications of dynamical systems**

Dynamical systems are mainly represented by a state that evolves in time. The input as well as the current state of a dynamical system determine the evolution of the system. Typically, an output is generated from the state of the system. This is a rather general definition of a dynamical system, where many different systems fit into. For investigating dynamical systems, it is necessary to specify some characteristics that provide a subdivision into special classes of dynamical systems. Specific methods are available for some of these classes; thus, such a classification can help to simplify the analysis.

An important characteristic of a dynamical system is whether it is continuous or discrete. Continuous systems (often called flows) are given by differential equations whereas discrete dynamical systems (often called maps) are specified by difference equations. Autonomous systems are characterized by the fact that input and output are omitted from the definition.

An important criterion for the analysis of a dynamical system is whether it is time-dependent or not. For time-dependent dynamical systems the function that specifies  (continuous case) or  (discrete case) depends on the time itself whereas for time-independent systems this function does not change over time.

For the analysis it is very important whether a dynamical system is linear or not. Linear dynamical systems are simple to analyze as opposed to non-linear systems, which typically do have intricate dynamical behavior. Often linearization at specific locations is used to get insights into these complex non-linear dynamical systems.

Using linearization, another classification of dynamical systems is crucial to separate simple cases from more complex ones. Hyperbolic dynamical systems can be analyzed by linearization efficiently, whereas non-hyperbolic systems may cause major troubles in combination with linearization. Hyperbolic systems are structurally stable, i.e., small perturbations of the system parameters do not change the qualitative behavior of the system. Non-hyperbolic systems are difficult to investigate, occur rarely and can be considered the transitional phase between two hyperbolic systems of different nature.

**Lorenz System**

* 1. The Lorenz system for is

,

* 1. = ,

= .

The system depends on three positive parameters σ, r, b; a commonly studied case is σ = 10, r = 28, and b = 4/3. Lorenz obtained as a truncated model of thermal convection in a fluid layer, where σ has the interpretation of a Prandtl number (the ratio of kinematic viscosity and thermal diffusivity), r corresponds to a Rayleigh number, which is a dimensionless parameter proportional to the temperature difference across the fluid layer and the gravitational acceleration acting on the fluid, and b is a ratio of the height and width of the fluid layer. Lorenz discovered that solutions of behave chaotically, showing that even low-dimensional nonlinear dynamical systems can behave in complex ways. Solutions of chaotic systems are sensitive to small changes in the initial conditions, and Lorenz used this model to discuss the unpredictability of weather (the “butterfly effect”).

If is a zero of , meaning that

(1.3) ,

then has the constant solution . We call an equilibrium solution, or steady state solution, or fixed point of .An equilibrium may be stable or unstable, depending on whether small perturbations of the equilibrium decay — or, at least, remain bounded — or grow. The determination of the stability of equilibria will be an important topic in the following. Other types of ODEs can be put in the form. This rewriting does not simplify their analysis, and may obscure the specific structure of the ODEs, but it shows that is rather a general form.

**RAM Exhaustion:**

Out of memory (OOM) is a state of computer operation (often undesired) where no additional memory can be allocated for use by programs or the operating system. Such a system will be unable to load any additional programs and since many programs may load additional data into memory during execution, these will cease to function correctly. This occurs because all available memory, including disk swap space, has been allocated. The typical OOM case in modern computers happens when the operating system is unable to create any more virtual memory, because all of its potential backing devices have been filled.

**Memory Fragmentation**

When memory is allocated in a system, not all of the available is always consumed in a linear manner, which can lead to fragmentation. There are two core types of memory fragmentation, internal and external: Internal Fragmentation is when memory is allocated to a process or application and isn’t used, leaving un-allocated or fragmented memory. External Fragmentation happens as memory is allocated and then deallocated, there can be small spaces of memory leftover, leaving memory holes or “fragments” that aren’t suitable for other processes.

**Part 1**

The Lorenz equations, the implementation can look something like:

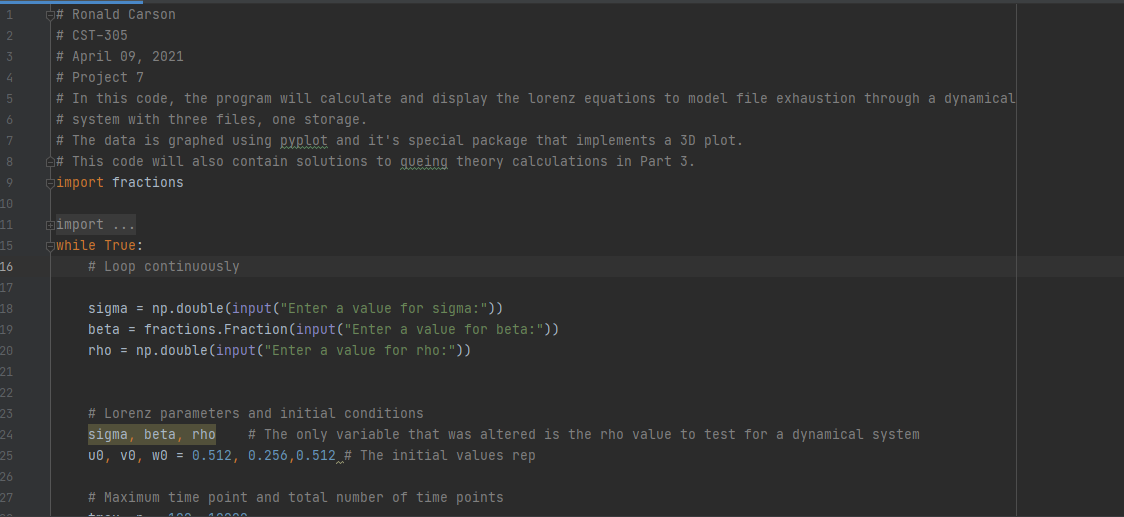
,

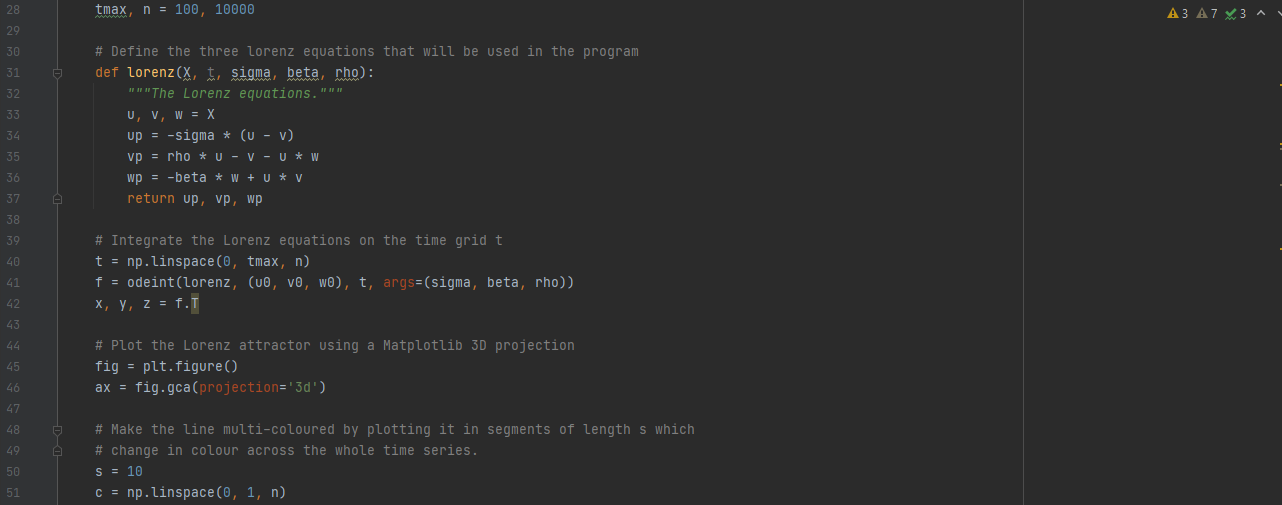
= ,

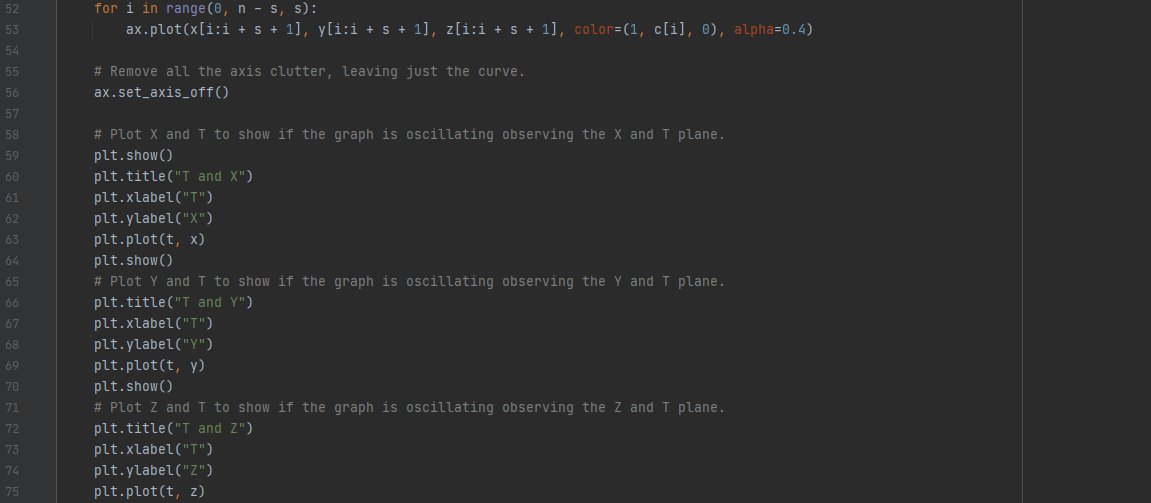
= .

The constants , , and are system parameters proportional to the Prandtl number, Rayleigh number, and certain physical dimensions of the layer itself. Different values of \rho give qualitatively different solution curves for this system of equations. For example, for \rho = 28 the solution is chaotic, meaning that nearby points diverge approximately exponentially rapidly (so prediction with time is difficult to impossible). Where rho >= 25, beta = 8/3, and sigma = 10. To properly implement to create a dynamical system, beta and sigma remained the same while rho continued to be altered. The code section below shows the outputs at each value of rho I tested. Rho influences how many rings are created in the 3D plot as well as determining the strength of the oscillation in the graphs that showing the 2D planes separately (x, t: y, t: z, t). In this problem, rho increasing can represent a number of things that are affecting the memory of a disk. The dynamical system shows itself when rho >= 25 (as shown below) meaning the disk does not have any more space. I believe the dynamical system that is created to be *continuous, non-linear,* and *stochastic*. I find the model to be continuous because the change in state is not abrupt and the change is continuous but not quite countable. I find it also be non-linear because of the x, y, and graphs, the analysis is not straight forward and do not follow the definition of a linear system. Lastly, I think the system is stochastic because the behavior of the system is not entirely predictable. This makes sense because the model I am basing the problem off of is the Chaos Theory and the Lorenz attractor.

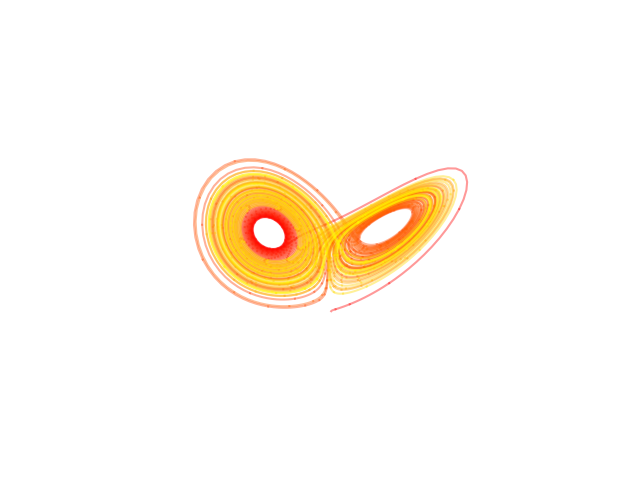
### Code Output and Code Part 1

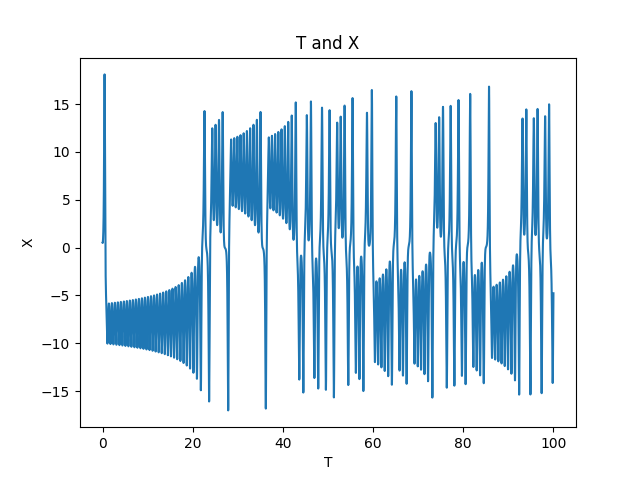


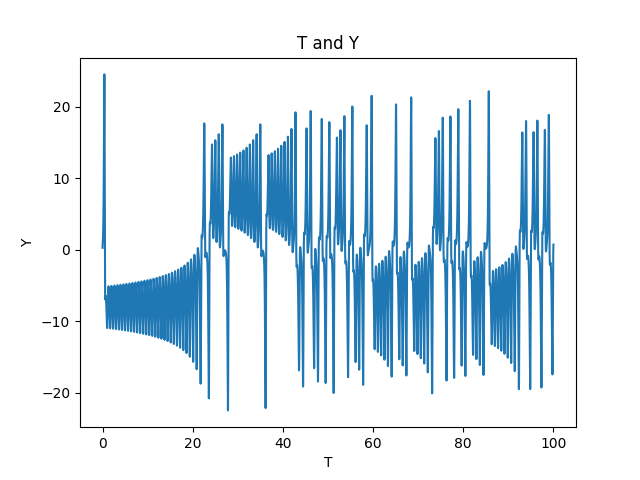


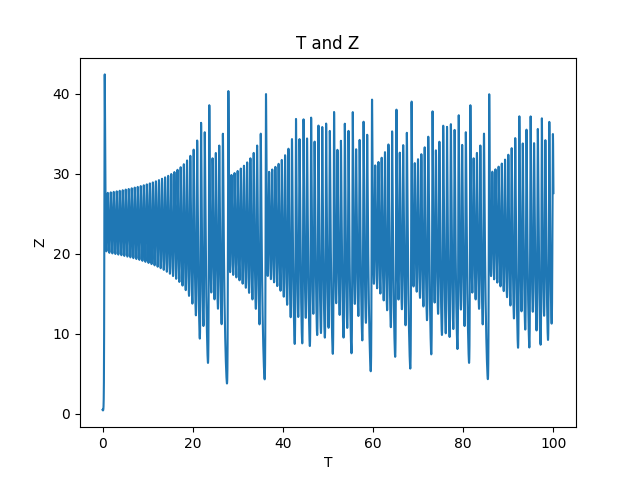


Initial Dynamical System when This is where the system starts to become unstable, the rings have formed, the x, y, and z graphs are oscillating after the beginning of the graph shows signs of a stable system.

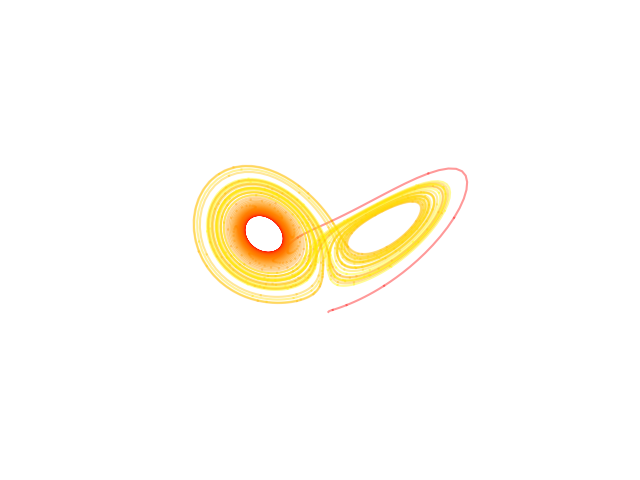


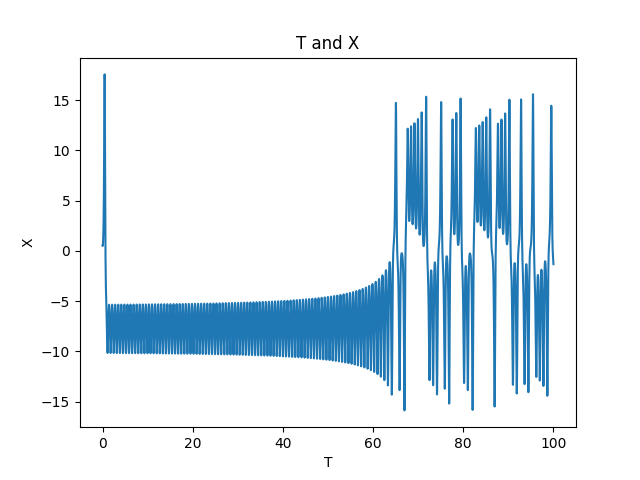


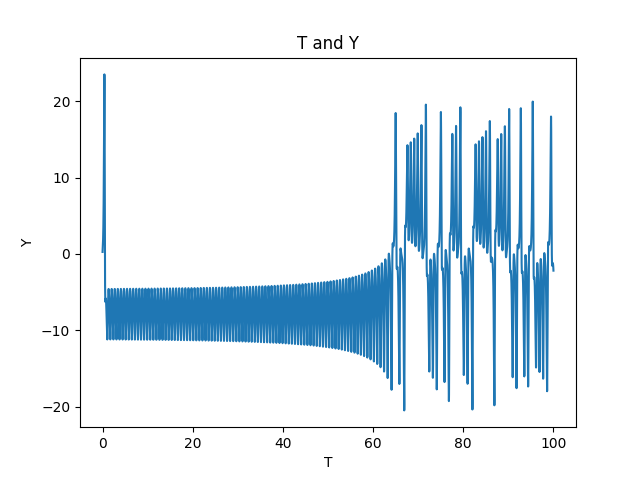


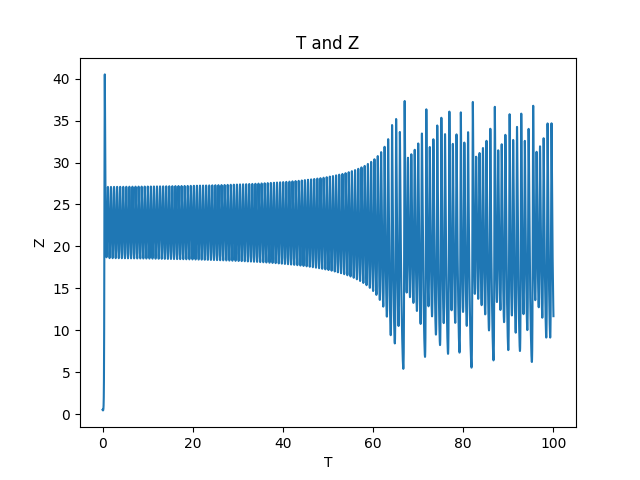


Dynamical System when . Here, you can see the 3D has formed rings but the system has not converged quite yet. The graphs maintain stability for the majority of the output until it reaches the limit. However, it shows signs of it reaching an unstable system. The disk is reaching the point where it does not have the necessary space.

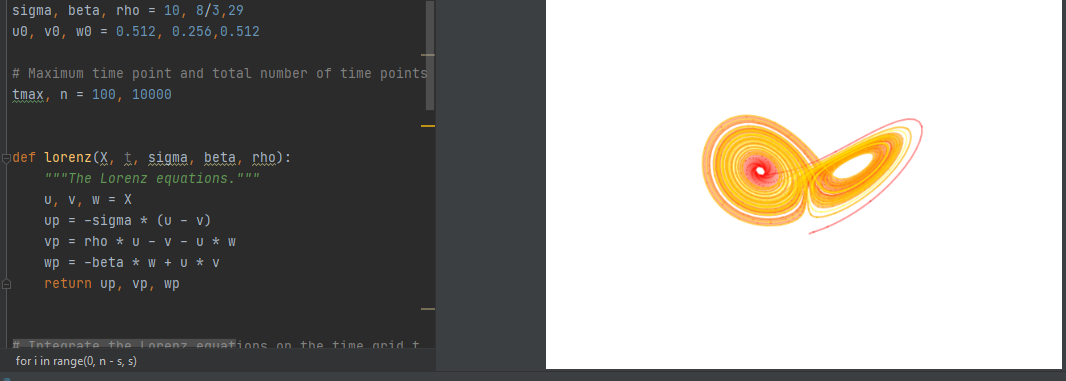


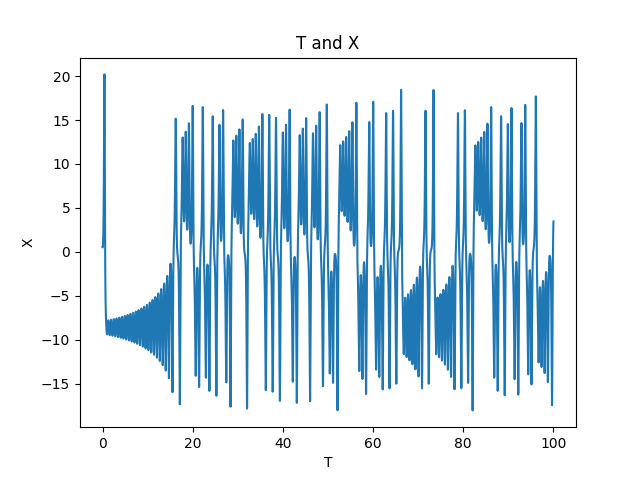


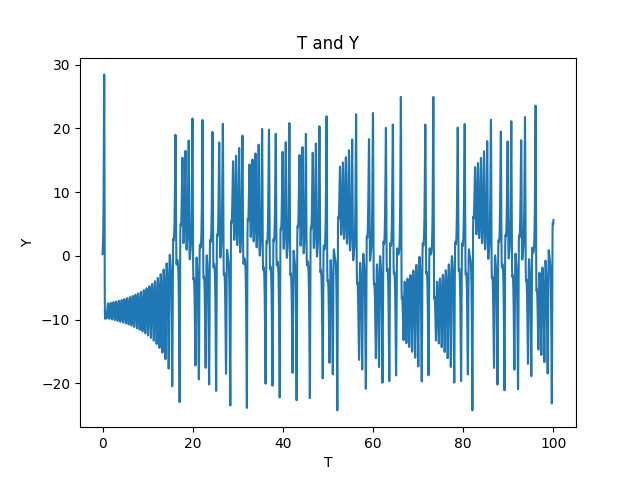


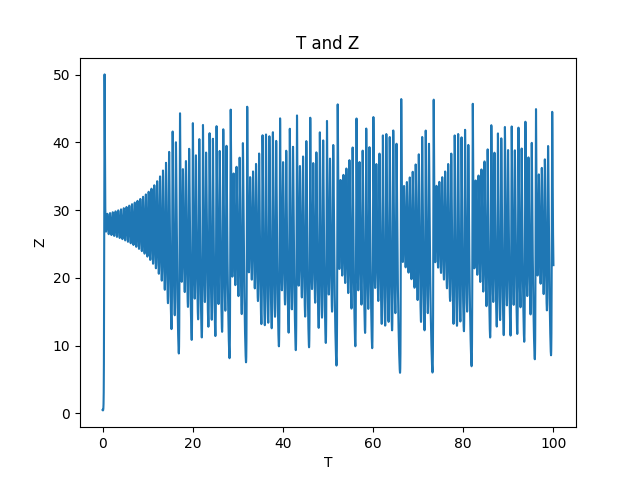


Dynamical System when . Plunging further into chaos, the file system is over it’s limit for memory and rho increasing is adding much more rings to the 3D graph and strength in the oscillation in the x, y, and z graphs. As time goes on, the chaos and instability increase along with rho.

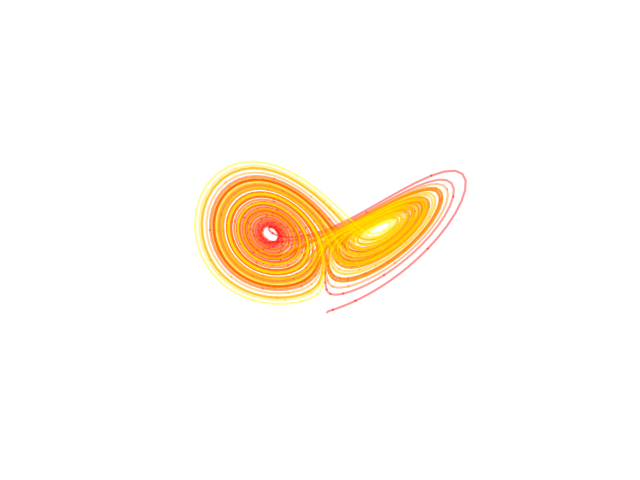


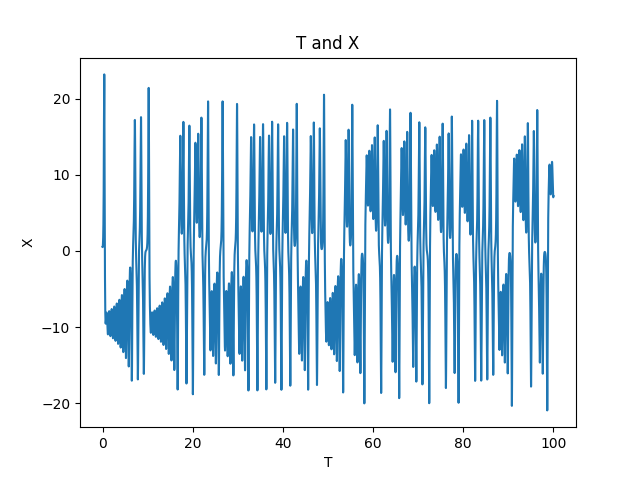


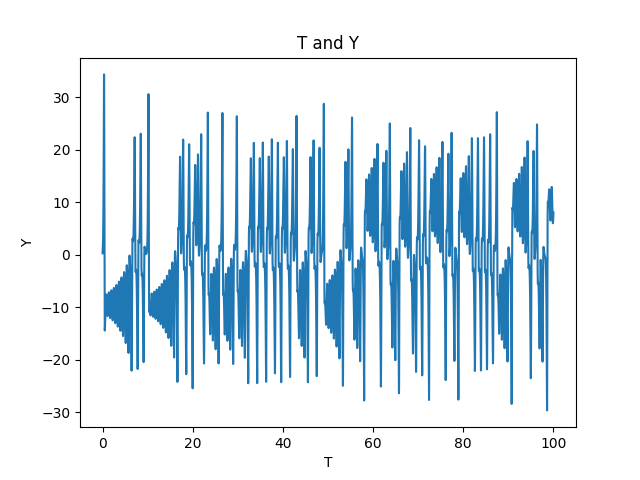


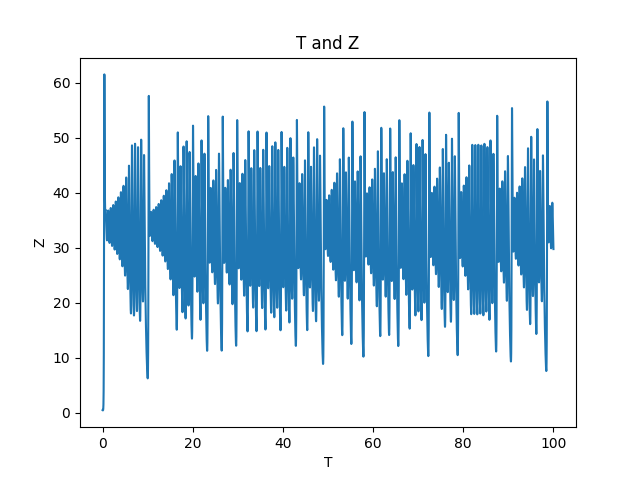


Dynamical System . Rho continues to increase and add rings. The colors showing the change, and the graphs continuously repeating the strong oscillation patterns.

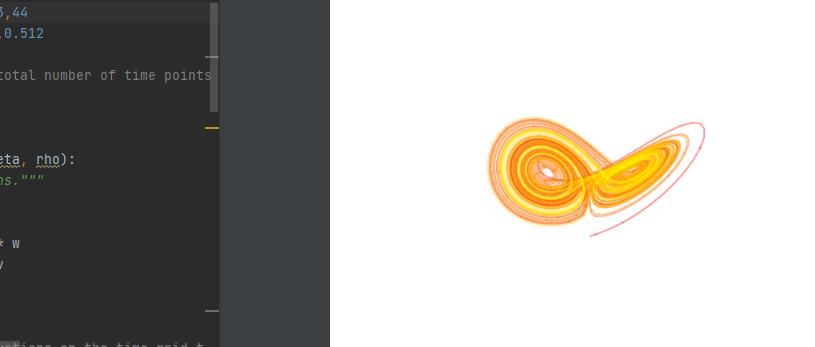


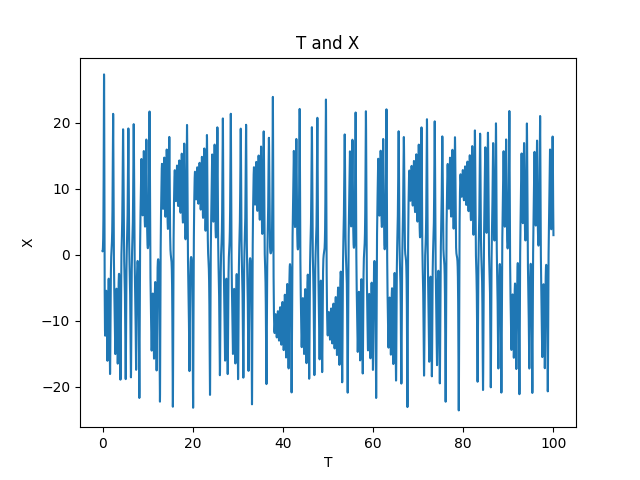


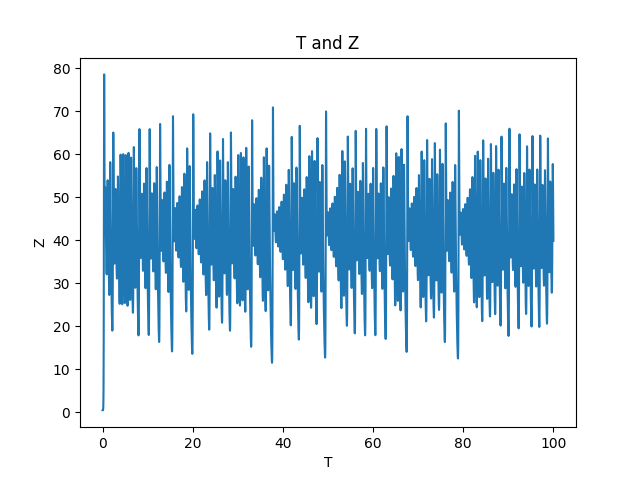


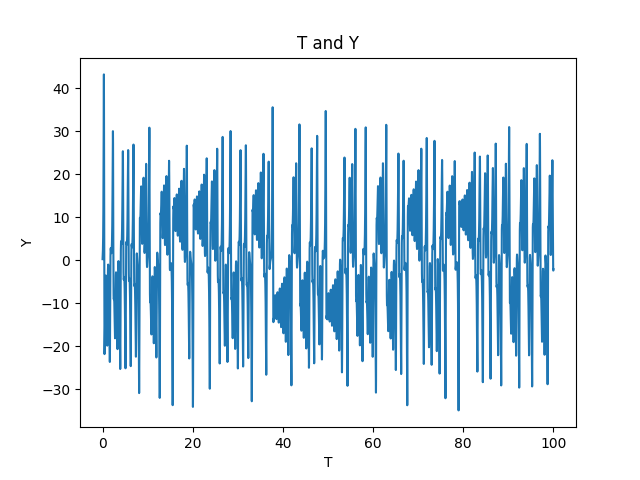


Dynamical System when

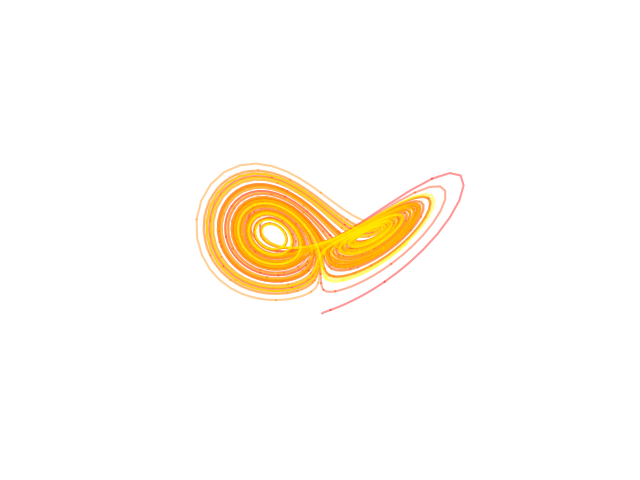


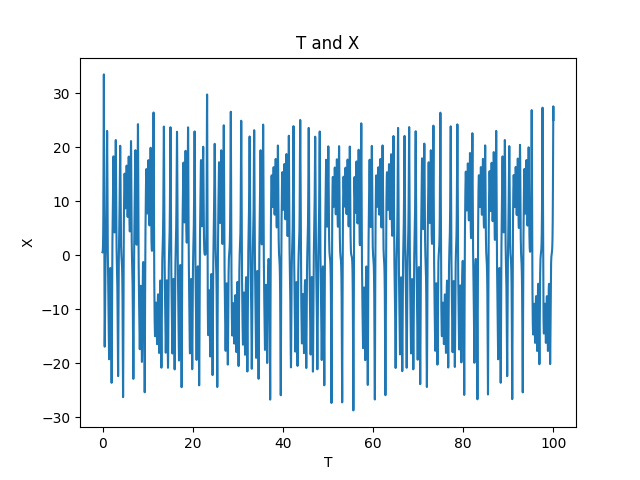


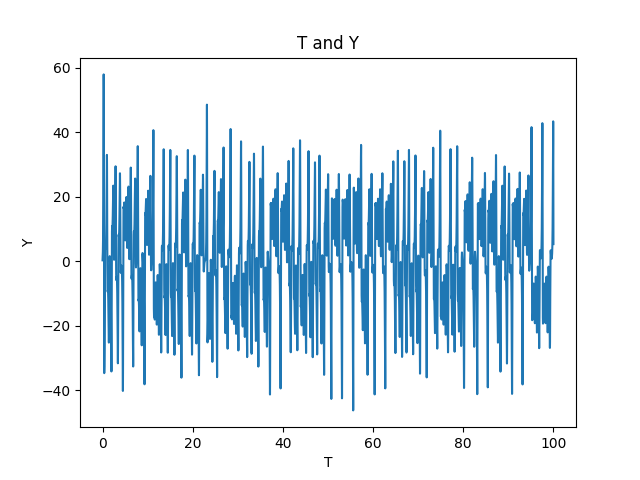


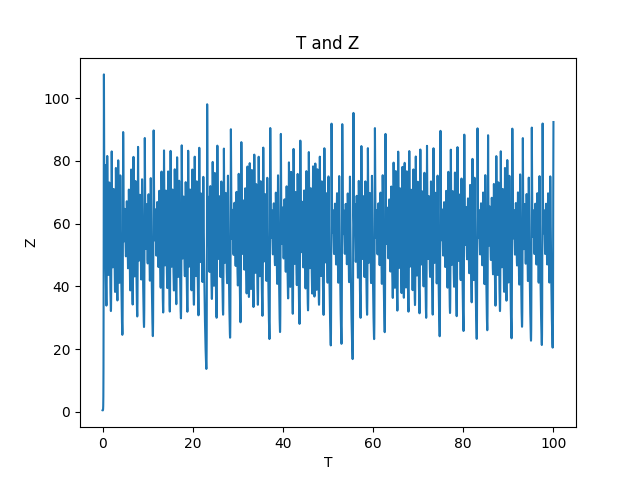


Super Dynamical System when



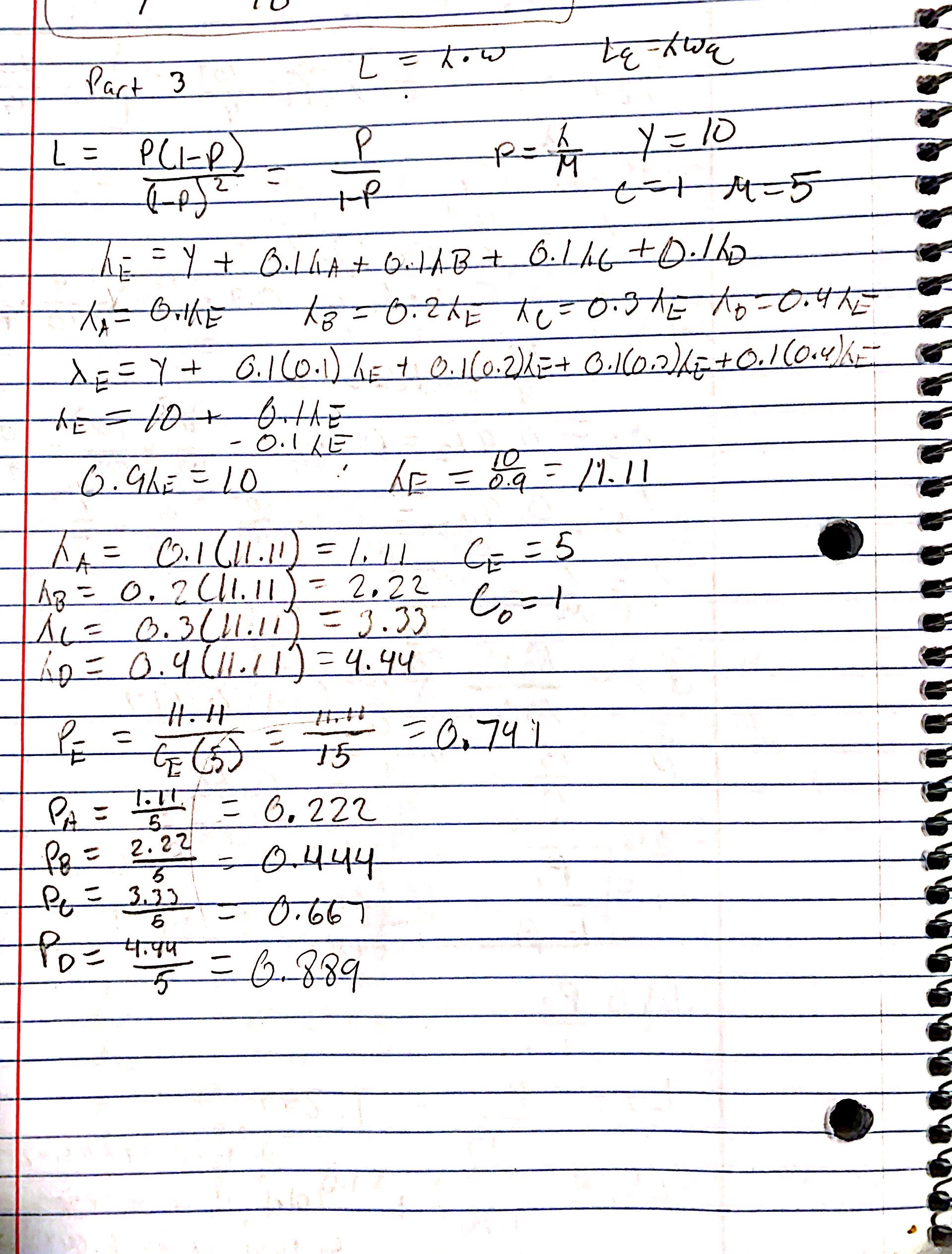


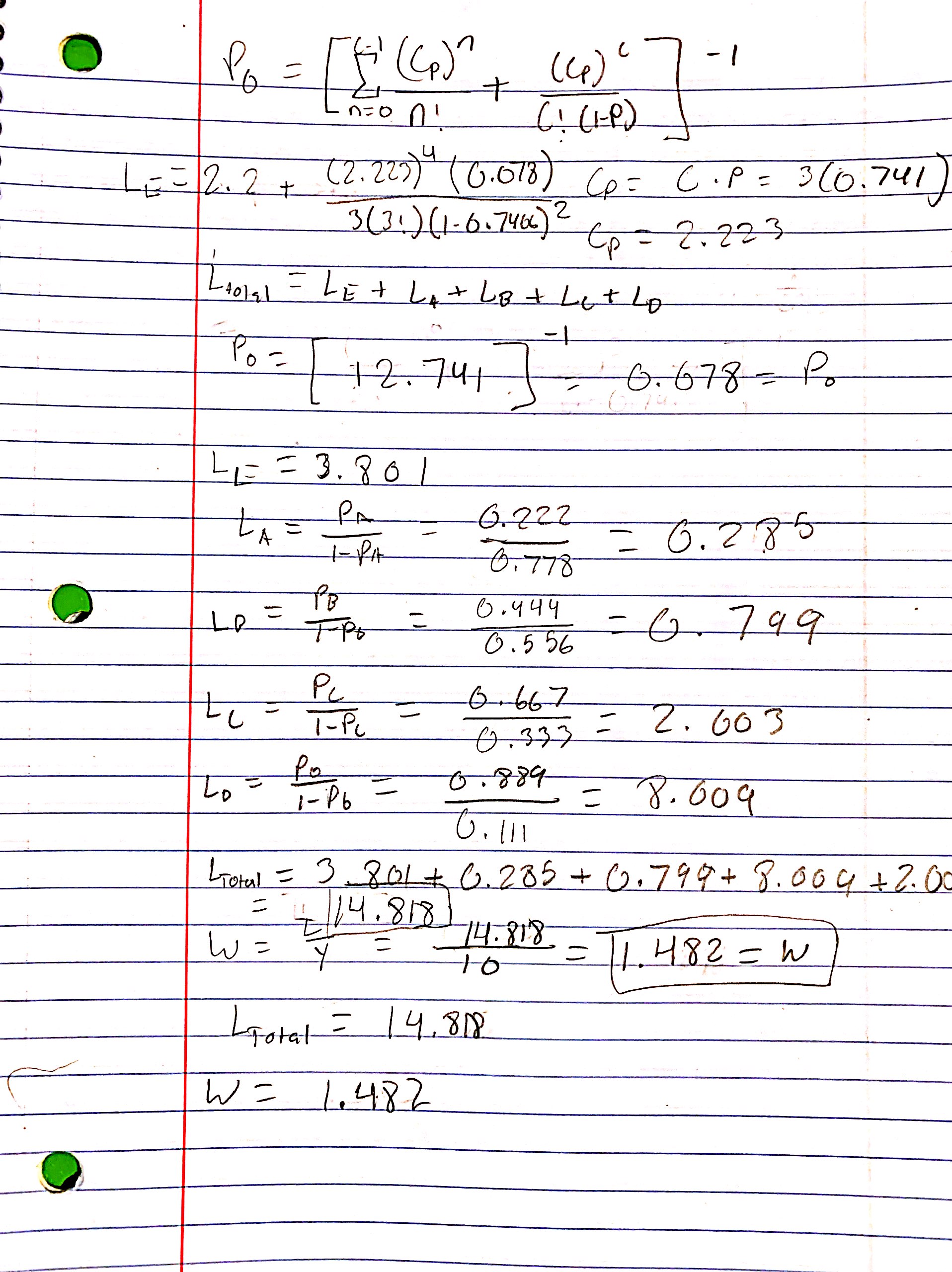




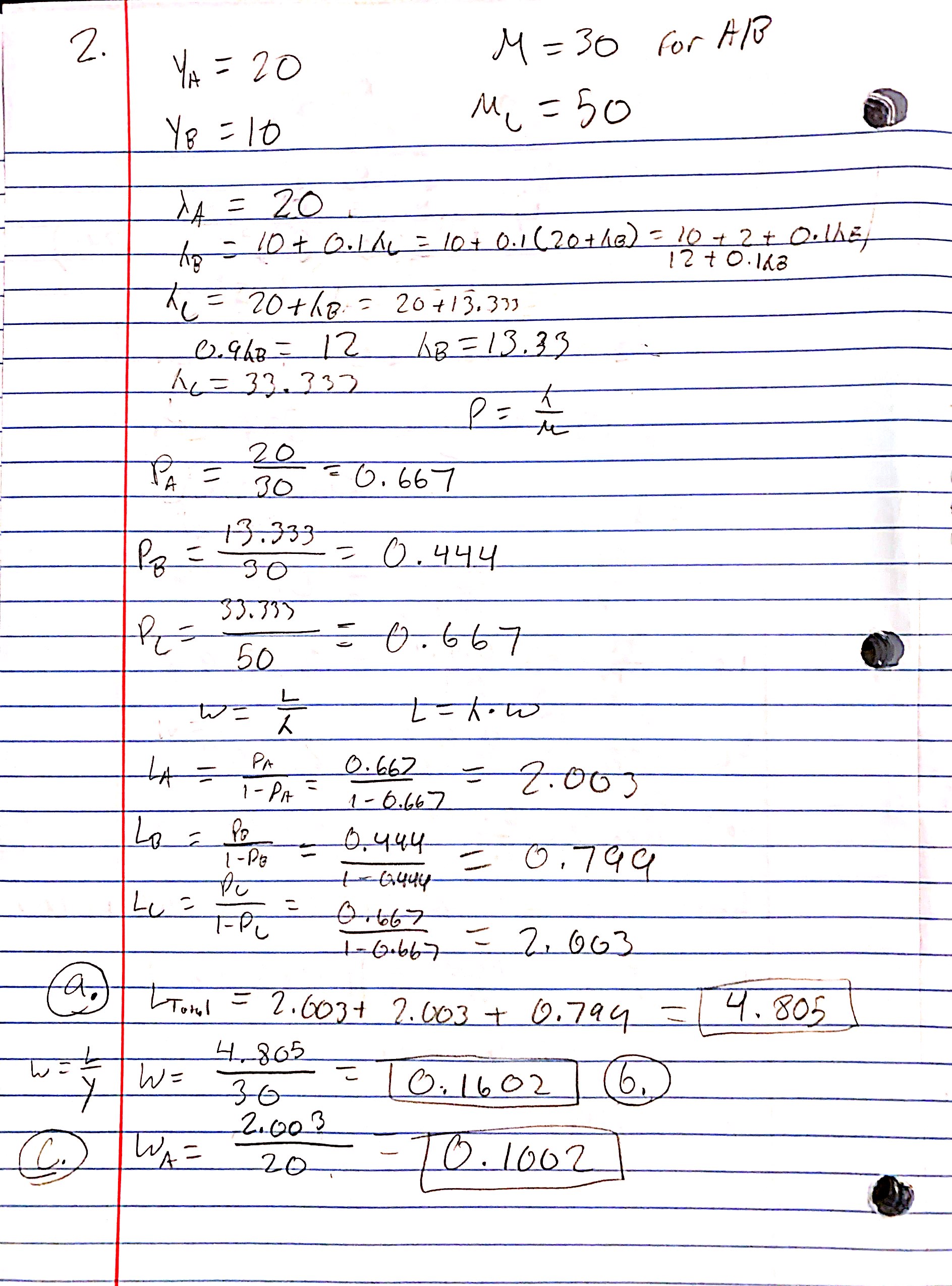
**Handwritten Work for Part 2**

**Question 1**

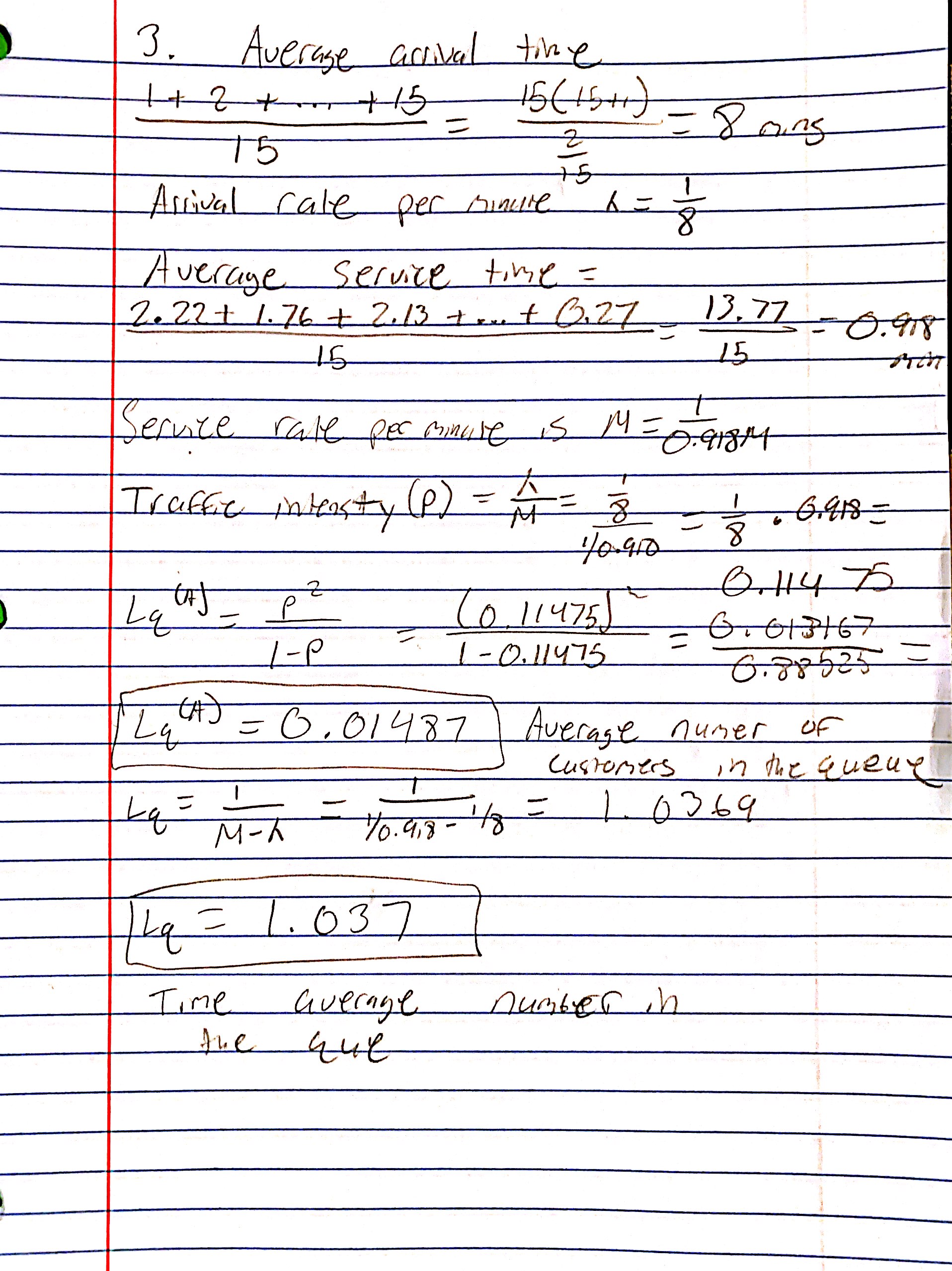
****

****

**Question 2**

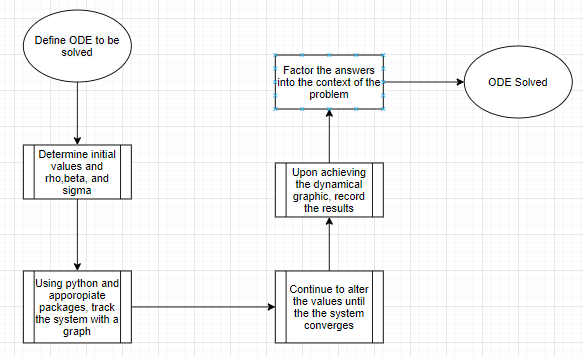
****

**Question 3**

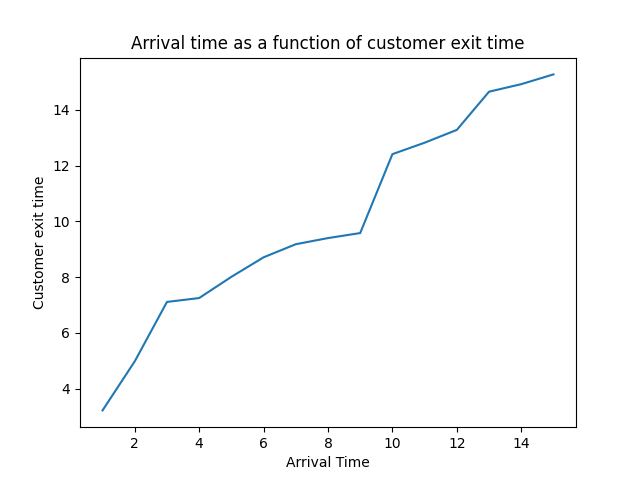
****

**Flowchart**

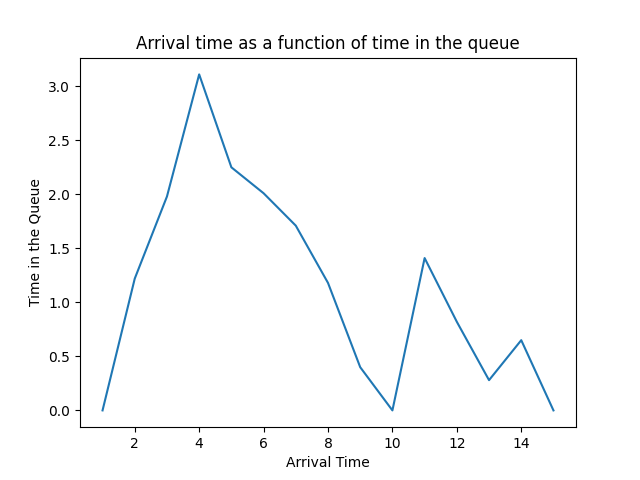
The flowchart below explains the steps that are to be taken to solve a dynamical system with the code of the Lorenz attractor.

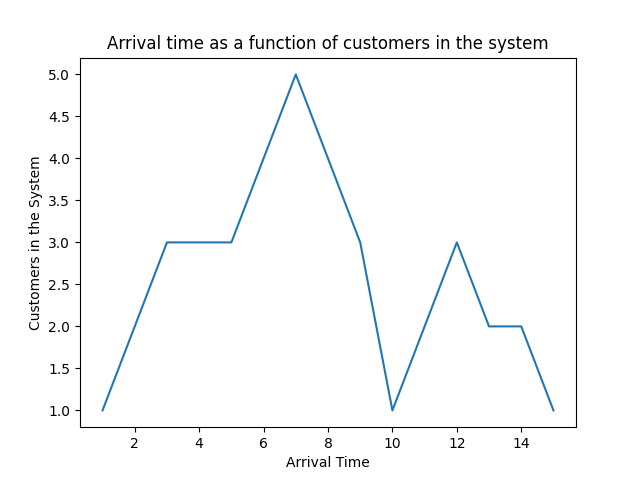


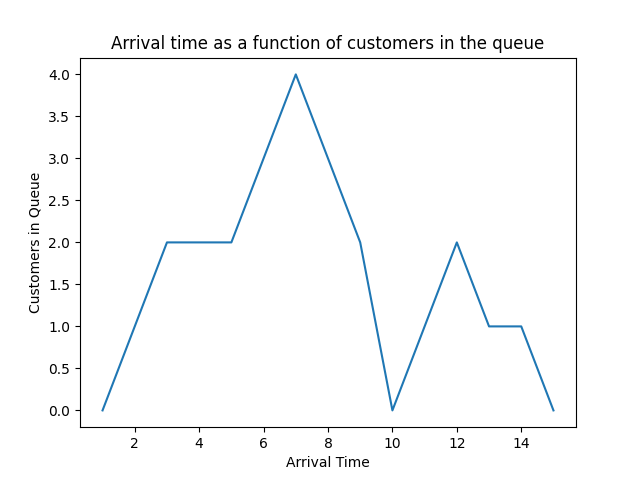
**Part 3 Code and Output**

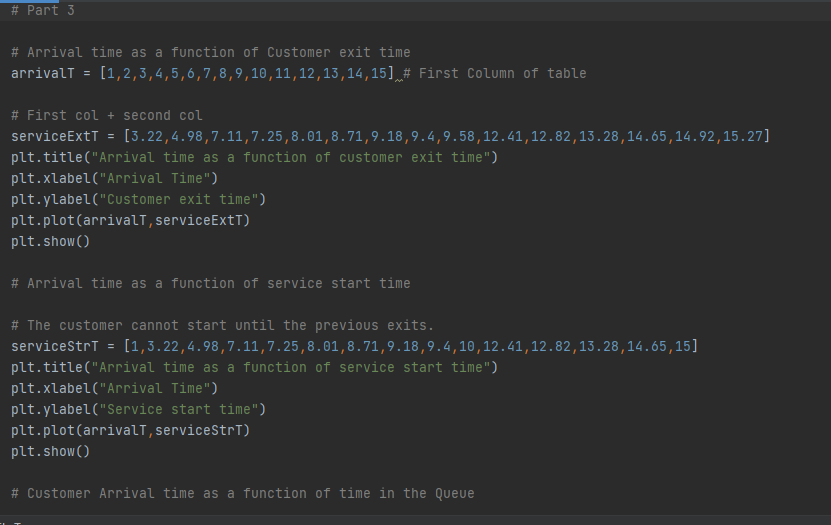
****

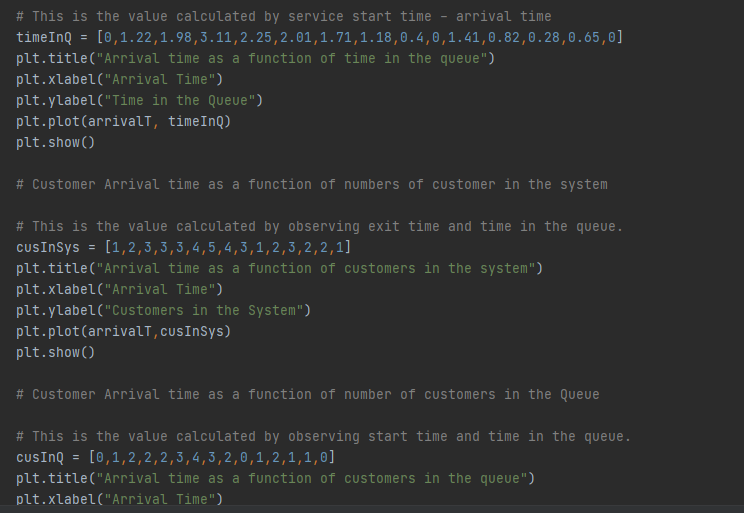
****

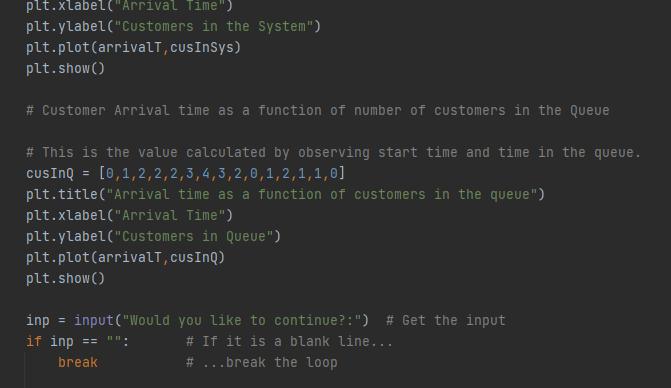
****

****

****

****

****

****

**­­**

**Final Analysis:**

As time goes on and rho continues to grow, the chaos and instability grow with it as show in the code output from rho = 25 to infinity. In the real-world problem, I believe anything past rho = 35 would represent the disk being way too full to even function or repair itself. Anything more than that would further exhaust the memory. The Lorenz equations and attractor serve as great models for chaos theory and chaotic dynamical systems in the world. Queues have a number of great uses, serving requests on a single shared resource, like a printer, CPU task scheduling etc. In a real-life scenario, call center phone systems use Queues to hold people calling them in an order, until a service representative is free. Queues also take care of the handling of interrupts in real-time systems. M/M/1 refers to negative exponential arrivals and service times with a single server. This is the most widely used queueing system in analysis as pretty much everything is known about it. M/M/1 is a good approximation for a large number of queueing systems. M/M/C is great when dealing with multiple series, probabilities, as well as the Markov chain. All of these concepts were explored in great detail in this project.

## References

1. <https://www.math.ucdavis.edu/~hunter/m207a/ch1.pdf>
2. <https://mathinsight.org/dynamical_system_idea>
3. <https://users.cg.tuwien.ac.at/helwig/diss/node14.htm#:~:text=Dynamical%20systems%20are%20mainly%20represented%20by%20a%20state%20that%20evolves%20in%20time.&text=Continuous%20systems%20(often%20called%20flows,by%20difference%20equations%20%5B83%5D>.
4. <https://manage.dediserve.com/knowledgebase/article/145/what-is-ram-exhaustion---also-known-as-oom--/>
5. <https://queue-it.com/blog/queuing-theory/#:~:text=Queuing%20theory%20(or%20queueing%20theory,customer%2C%20job%2C%20or%20request>.
6. https://en.wikipedia.org/wiki/Lorenz\_system