CSC 614 Computer Performance Analysis

Dr. Jozo Dujmovic

Analysis of Continuous vs. Discrete Simulation Models

G/G/1 Simulator

*With Graphed Queue Lengths and a Simulation GUI*

By: Andrey Barsukov

November 12, 2015

**Table of Content**

**I. About ………………………………………………………………………………………… 2**

**II. Continuous vs. Discrete models ………………………………………………… 3**

**III. G/G/1 Simulator Program ………………………………………………………... 6**

**IV. Simulation Data ……………………………………………………………………….. 8**

**- Console output, error analysis**

**V. Simulation GUI ………………………………………………………………………….. 12**

**VI. Queue Length Graphs ………………………………………………………………. 13**

**VII. Conclusion ………………………………………………………………………………. 15**

**VIII. C++ Code …………………………………………………………………………………16+**

**I. About**

One of the ways the effectiveness of a system can be analyzed and studied is by simulation.

In this project we simulate a G / G / 1 Server System.

We model and simulate the performance of this system using both simulations and analysis, then comparing the results to give us an idea of how accurate our simulation was.

*G/G/1: A general case of single server system with general independent arrivals and general service times*

Some of the things we look at when simulating a queuing system are:

• interarrival time

• service time

• queue length

• server utilization

• response time

Beyond that we can also analyze the distribution of the systems queue length to get a better understanding of utilization.

In the following program I will simulate a range of jobs with the following parameters and distributions.

The G/G/1 System simulated here follows all the possible combinations of

the following distributions:

# Distribution of interarrival time | # Distribution of service time

1 Constant value: 2 sec | 1 Constant value: 1 sec

2 Exponential with mean value of 2 sec | 2 Exponential with mean value of 1 sec

3 Uniform from 1 to 3 seconds | 3 Uniform from 1 to 2 seconds

**II. Continuous versus Discrete Simulators**

One of the biggest questions in faced when constructing a simulator is weather to make a continuous or a discrete simulator.

General Definitions:

* **Discrete model**: the state variables change only at a countable number of points in time. These points in time are the ones at which the event occurs/change in state.
* **Continuous**: the state variables change in a continuous way, and not abruptly from one state to another (infinite number of states).

In our case,

Continuous: Our G/G/1 Simulator would operate on a simulated “clock”. We set a delta time t which we increment in very small increments and only take action when something happens. Each clock tick we would check if something is supposed to happen. If nothing happens we just increment the clock and keep going. When we create a clock state where an even does happen we would take appropriate actions.

Discrete: A discrete simulator is based on events. We also follow a timeline just like in a continuous simulation but we skip all of the downtime where nothing happens. This means that we advance the clock forward to the first event, take actions corresponding to that even, then jump to the next event.

For our simulator, the discrete events are:

Jobs arriving to queue

Job going into server

Job done being served -> Job leaving

At each of these specific events we take actions like incrementing like recording time spent in queue, total wait time and so on. We will talk about the specifics of this in the next part.

But the question still stands, which one do we pick: continuous or discrete?

The simple answer is – it depends.

Do you want to implement a graphical representation of your simulation? Then you would most likely go with a continuous simulator.

Or maybe you want to simulate a very high number of jobs (or events), with high result accuracy. Then go with discrete.

*Some things to consider when picking a simulation model*

* **Deterministic model** is one whose behavior is entire predictable. The system is perfectly understood, then it is possible to predict precisely what will happen.
* **Stochastic model** is one whose behavior cannot be entirely predicted.
* **Chaotic model** is a deterministic model with a behavior that cannot be entirely predicted

What kind of model are we working with? Can we easily predict its events?

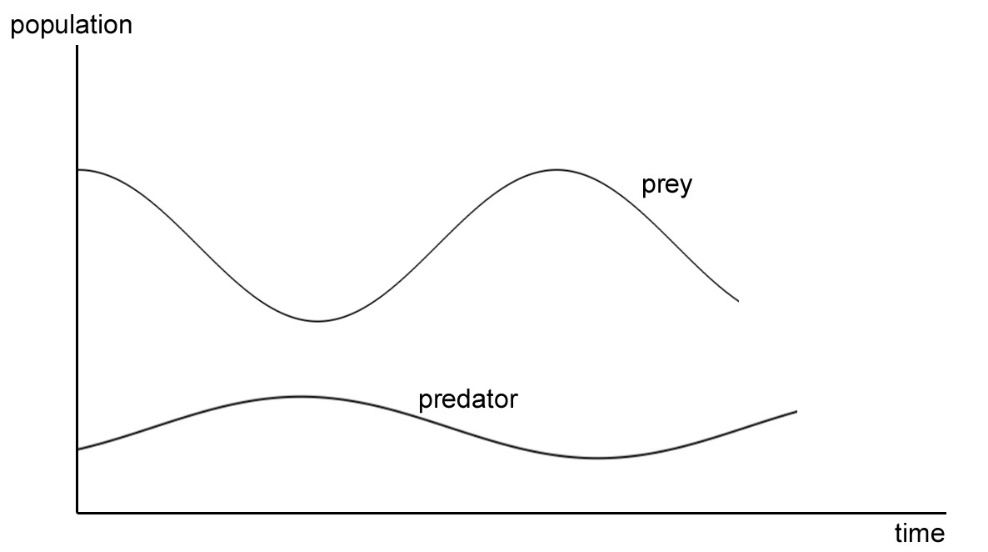
**Deterministic model (G/G/1):**

So, knowing that the behavior of our system is Deterministic we can confidently choose a **Discrete simulation model.**

**Stochastic model – Predator/prey**

An example of a stochastic model would be a predator/prey model.

This model is typical for revealing the dynamics of populations. As long as the population of the prey is on the rise, the predators population also rises, since they have enough to eat. But very soon the population of the predators becomes too large so that the hunting exceeds the recreation of the prey. This leads to a decrease in the prey’s population and as a consequence of this also to a decrease of predators population as they do not have enough food to feed the entire population.

Simulation of any population involves counting members of the population and is therefore fundamentally a discrete simulation. However, modeling discrete phenomena with continuous equations often produces useful insights. A continuous simulation of population dynamics represents an approximation of the population effectively fitting a curve to a finite set of measurements/points (<https://en.wikipedia.org/wiki/Continuous_simulation>, 11/11/2015).

**What if we use a continuous model for our G/G/1 simulation?** *(Sketch on next page)*

**III. G/G/1 Simulator Program**

Not realizing the difference between a continuous and discrete models, I starting this project on the premise of making a continuous model simulator. As I was programming it I started to see the behavior of a continuous model and realize the inherent accuracy errors and large overhead. This actually pushed me to further research into the difference between these two.

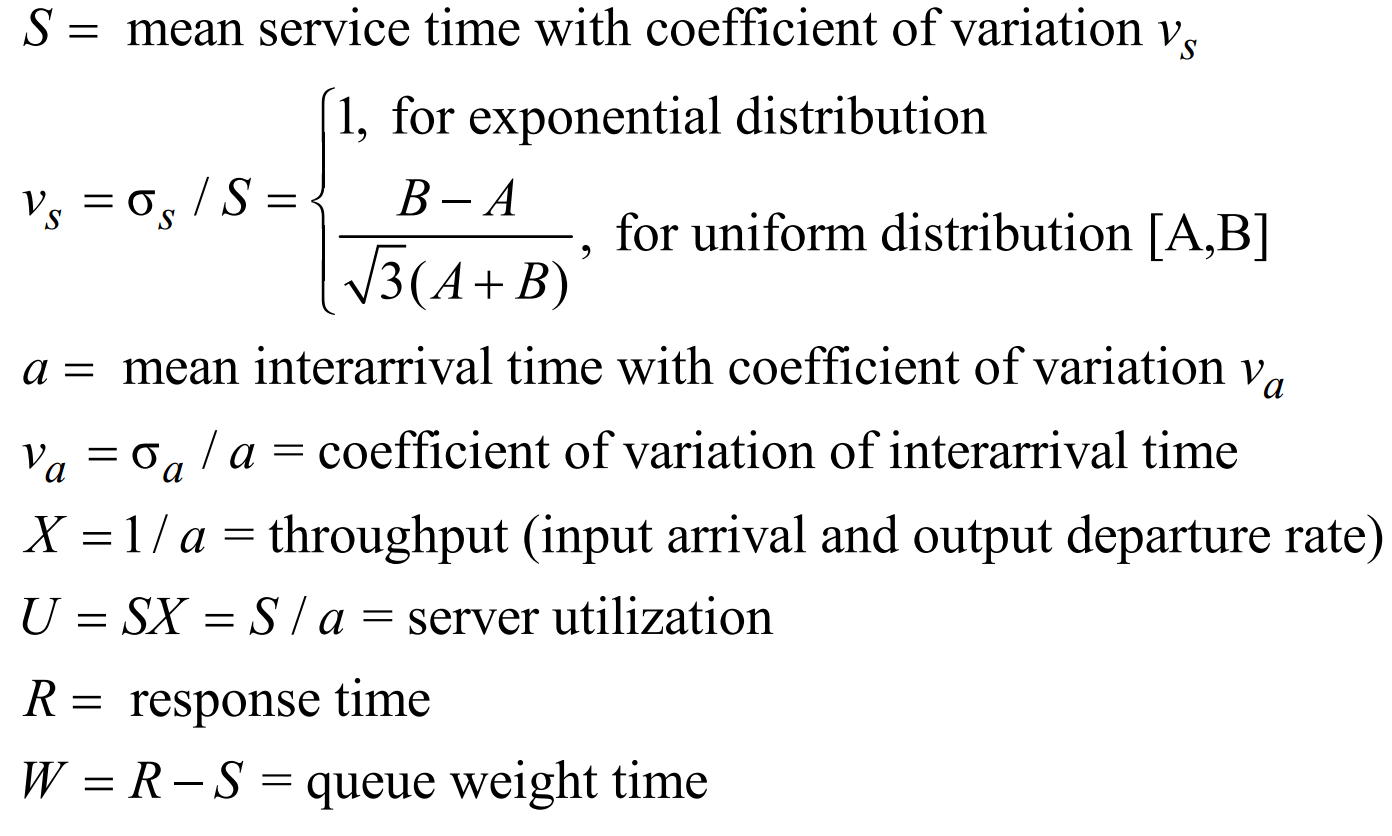
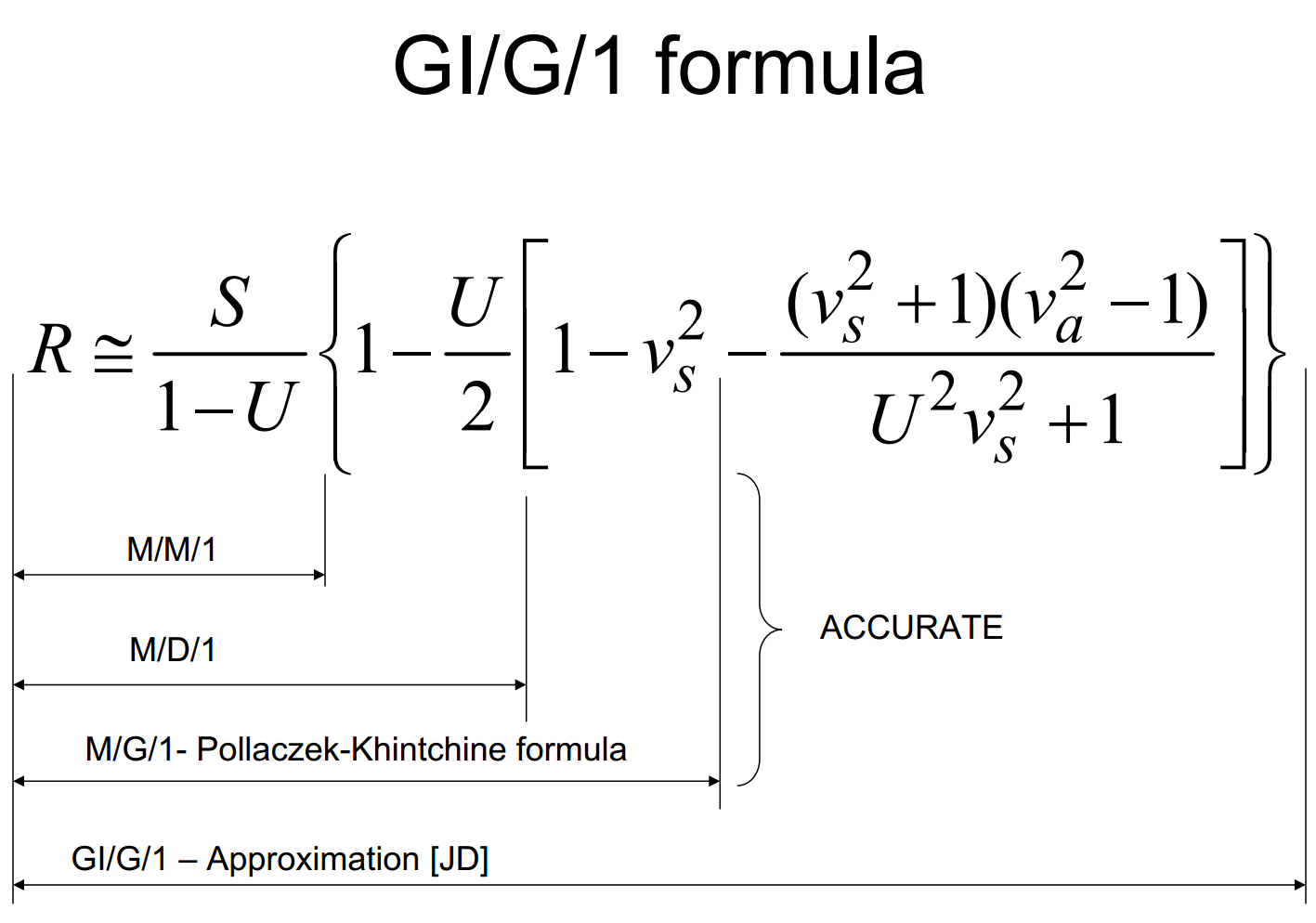
The results I was getting from my continuous model where not too far off but the randomness of their accuracy worried me.

After some research and explanation from Dr. Jozo Dujmovic I was convinced that I needed to rework my simulation into a discrete one, or at least fix the calculation errors that a continuous model produced.

There was benefit to this, with both models in hand I was able to compare and contrast the two to see first-hand the difference in their results. I was also able to take the continuous model further and implement a simple GUI for the single server system.

**General program steps.**

* As stated above, this simulation is deterministic – we can predict its behavior. Because of this, we can pre-generate some of the data.
  + Create a **struct** Job which holds the job’s id, service time, arrival time, queue wait time
  + Generate a specified number of jobs with randomly generated service time and arrival time based on the distributions given above and store in an array **ReadyQ[MAX\_JOBS].**
* Which this set of Jobs, we can run the simulation and record data
  + Monitor job arrivals by keeping a set of variables that indicate the virtual queue of the system, when a job arrives -> endOfQueue++, when job leaves startOfQueue++
    - System queue therefore is ReadyQ[startOfQueue] to ReadyQ[endOfQueue]
  + Before a job leaves the system, record the total time it spent in the system, and the time it was waiting in queue
    - Record the number of jobs inside waiting queue at a constant interval
      * This is later used to graph the ready queue lengths to get a feel for its distribution
* Using the array of queue lengths recorded during simulation we graph the queue
* Calculate the same values that we got through simulation by using Dr. Jozo Dujmovic’s G/G/1 Approximation formula based on the same distribution of interarrival and service times. (formula on next page)
* Compare Simulation results to calculated analysis and computer the error percentage
* Display data on console and write a more detailed breakdown of the data to a text file (text file name specified at end of program)
  + Compute the average error percentage for continuous vs discrete comparison
* Using my first continuous implementation of the simulation, a user can pick a specific combination of distributions that were previously calculated and run a ASCII based GUI of the system.

**Dr. Jozo Dujmovic’s formula **

**IV. Simulation Data**

The data is presented in the following manner:

Both interarrival time and service time have three different possible distributions

- A constant value

- An exponential distribution with a specified mean value

- A uniform distribution with a range from low - high

------------------------------------------------------------------------------------------------------------------------------------------

The G/G/1 System simulated here follows all the possible combinations of

the following distributions

# Distribution of interarrival time | # Distribution of service time

1 Constant value: 2 sec | 1 Constant value: 1 sec

2 Exponential with mean value of 2 sec | 2 Exponential with mean value of 1 sec

3 Uniform from 1 to 3 seconds | 3 Uniform from 1 to 2 seconds

------------------------------------------------------------------------------------------------------------------------------------------

a - Arrival Distribution | s - Service time distribution

simU - Simulated server utilization | an - Analytical server Utilization

E%U - Utilization % error | E%R - Response time % error

simR - Simulated Response time | anR - Analytical responnse time

simQ - Simulated Queue size average | anQ - Analytical Queue size average

simW - Simulated average wait time | anW - Analytical average wait time

simA - Average simulated interratival | simS - Average simulated service time

To insure the validity of data, a random seed is used, in this case we seed the time stamp of the given system

Random seed: time(0)

**The following data uses a continuous model with delta t = 1/1000 seconds for comparison with a discrete model**

------------------------------------------------------------------

G/G/1 Simulator - Simulating 2000 jobs.

==================================================================

a s simU anU E% simR anR E% simQ anQ E%

1 1 | 0.4999 0.5000 -0.0250 1.0000 1.0000 0.0000 0.4999 0.5000 -0.0250

1 2 | 0.5069 0.5000 1.3781 1.2775 1.2000 6.4603 0.6387 0.6000 6.4470

1 3 | 0.7527 0.7500 0.3541 1.5060 1.5476 -2.6877 0.7527 0.7738 -2.7337

2 1 | 0.5120 0.5000 2.3903 1.5410 1.5000 2.7323 0.7886 0.7500 5.1533

2 2 | 0.5229 0.5000 4.5788 2.1364 2.0000 6.8179 1.0949 1.0000 9.4882

2 3 | 0.7665 0.7500 2.2061 4.0598 3.8333 5.9090 2.0581 1.9167 7.3798

3 1 | 0.4997 0.5000 -0.0607 1.0005 1.0417 -3.9511 0.4997 0.5208 -4.0583

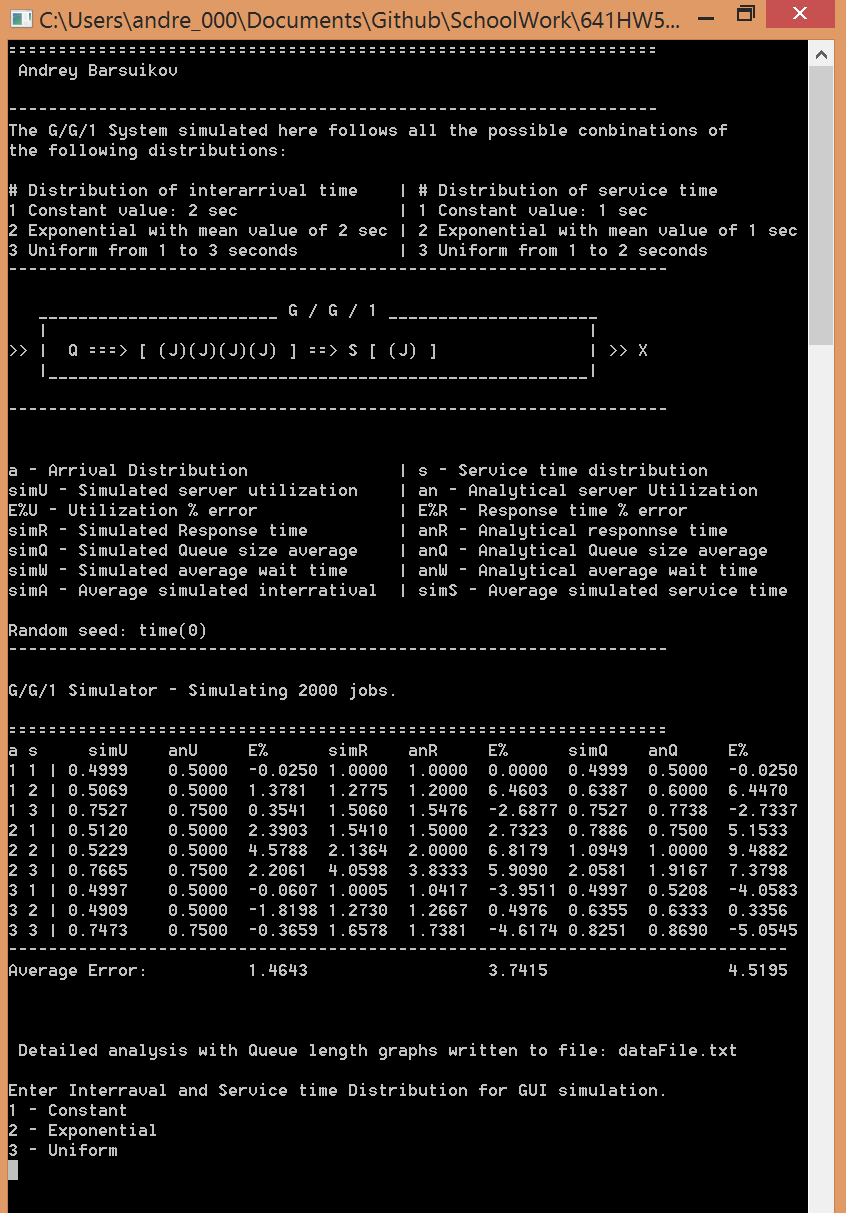
3 2 | 0.4909 0.5000 -1.8198 1.2730 1.2667 0.4976 0.6355 0.6333 0.3356

3 3 | 0.7473 0.7500 -0.3659 1.6578 1.7381 -4.6174 0.8251 0.8690 -5.0545

------------------------------------------------------------------------------

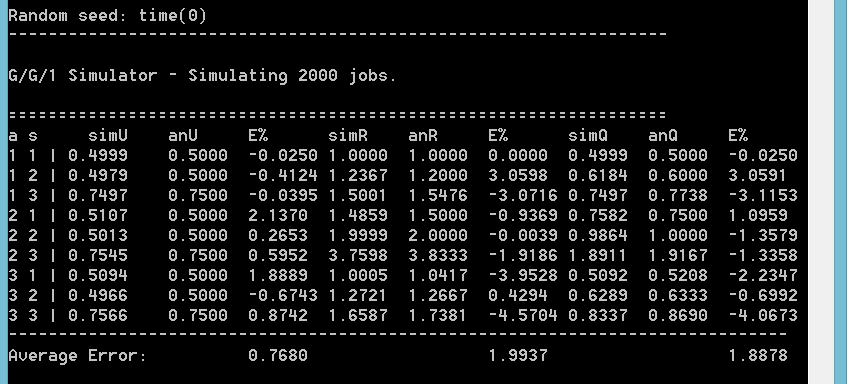
Average Error: 1.4643 3.7415 4.5195

**Output of console**



**Program fixed for a discrete simulation model.**

Lets looks at the data from a fixed discrete model and see how the error percentage compares.



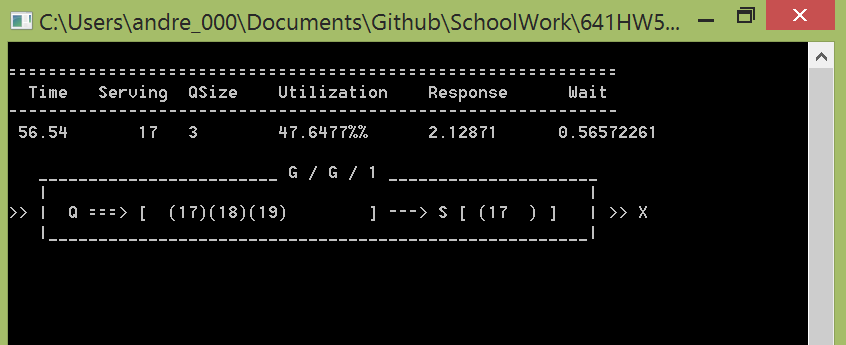
As we can see, the errors in each simulated field have gone down drastically.

|  |  |  |  |
| --- | --- | --- | --- |
|  | U | R | Q |
| Continuous | 1.4643 | 3.7415 | 4.5195 |
| Discrete | 0.768 | 1.9937 | 1.8878 |

**V. Simulation GUI**

With the continuous model we can implement the GUI for a selected simulation, this is a snapshot of this GUI.

**GUI using exponential interrarival time (mean 2) and uniform service time (1 – 2)**

****

All values are updated live and continuous throughout the simulation.

It is interesting to see the server utilization, response time, and queue wait time converge on to the same values that we got in our simulated and analytical data from above.

This GUI is achieved by resetting the console curser to a specific place when it needs to be updated

**VI. Queue length distribution graphing**

Something else that I was able to achieve through the continuous model is extracting the distribution of the graph lengths.

I declare a vector<int> and push\_back the current size of the system queue with every tick of the continuous clock.

Since I cannot post the whole file here, this is a small sample of the data that I was able to extract on the distribution of queue lengths.

To get the full detailed version, run the program and see the attached file created.



Every graph in the file reflected the expected behavior of its corresponding distribution.

Distribution combination 1, 1 (Constant arrival 2, constant service 1) for example has only two queue sizes. There are just as many appearences of queue size zero as there are one. This makes sense, but in this system the server is idle 1 second for every second that it is busy.

**Graph based on the raw data in the detailed analysis file**

Based on the same data as above, with the same distribution combinations

# Distribution of interarrival time | # Distribution of service time

1 Constant value: 2 sec | 1 Constant value: 1 sec

2 Exponential with mean value of 2 sec | 2 Exponential with mean value of 1 sec

3 Uniform from 1 to 3 seconds | 3 Uniform from 1 to 2 seconds

*I would really like to find out how to extrapolate this type of data from a discrete simulation model.*

**VII. Conclusion:**

I found this project challenging but very enjoyable. I am glad that I started this project the wrong way, it is what lead me to researching more about simulations. I must admit there are not many other projects to which I devoted a full week+ of non-stop work, but I feel satisfied having done it.

Through this project I have learned how simulators are use and why certain ones are used over others. I have learned how to build a continuous simulator and modify it to fix the error produced by it, compare simulated results to analytical and computer error calculations.   
I have dabbled with simple ASCII based simulation GUI and would like to pursue building more complex models.

I will continue to build on this simulator, perhaps expand it into a multi-server system with a graphical component.