Model B Overview

This simulation model simulates airtankers fighting fires and then graphs the average waiting time in a queue for fires, the standard deviation of this waiting time, the average service time over time, as well as the percent of fires that did not have to wait in the queue. It is written in python, and use the matplotlib library for graphing the results, the random library for generating random time values based on the probability distributions, simpy for the simulation processes, and numpy for improved numerical computation performance.

This program has two classes (objects), one called Parameters that stores all of the parameters and intermediate results from the simulation, and one called Outputs that stores all of the data that will be graphed. This model has three main sections: The simulation, analyzing the data into relevant statistics, and finally graphing these statistics.

The Simulation:

Using Simpy a simulation is created that will run for a user entered number of hours. A random number based on an exponential distribution with a user entered value for the mean times between fires is then continuously generated to get the times of the fires. There is also a user entered bin size that allows the program to consume less memory (at the cost of graphing less points at the end of the simulation). The number of fires generated in each bin is saved in a list. Every time there is a new fire, the function airtanker\_process is called to request an airtanker to fight the fire. If an airtanker is not currently busy then it will be immediately sent to fight the fire. In this case the fires wait time is considered 0. If all of the user inputted number of airtankers are currently busy fighting fires, then the fire is added to a FIFO queue. The time the fire spends in this queue is the wait time. The wait times of every fire are saved in one list, and a list is also generated that records the percent of fires that that had no wait time. Note that when the end of the simulation is reached, all fires that are still in the queue waiting are ignored because their wait time could not have determined yet. Finally randomly generated values for a travel time to the fire, an on-scene time and a return time are generated based on Erlang distributions with user inputted parameters. This service time determines the length of time an airtanker spends servicing a fire.

Analyzing Data

The next step of this model is to use the list outlining the number of fires in each bin, and the wait times of every fire to determine the statistics we wish to plot. The percent of fires with no wait time output is generated in the simulation. This step creates 4 new numpy arrays for the mean wait times, standard deviation of the wait times, the mean service times and the x-axis values. Note that the statistics are commulative: The first bin will have the stats for just the first bin, the second will create stats from all of the fires up to the end of the second bins time, all the way to the end of the simulation. Therefore as time increases into the simulation the statistics represent the overall averages and standard deviations of all the fires up to the point, not just the ones in the current bin. This is done by using the number of fires in each bin to determine how far in the wait time list we calculate the statistics. The calculations are done using numpy functions because it is written in C and therefore has better performance. I will give a quick example of how the looping works here:

Number\_fires = [1, 3, 2]

Wait\_times = [0, 3, 2, 0, 1, 3]

Average\_wait\_times = [mean(0), mean(0, 3, 2, 0), mean(0, 3, 2, 0, 1, 3)]

Std\_dev\_wait\_times = [std\_dev(0), std\_dev(0, 3, 2, 0), std\_dev(0, 3, 2, 0, 1, 3)]

Every next element in the array we will plot has however many new fires there were in the next bin more elements than the previous one.

The x axis values are just the values from t = 0 to t = simulation\_length - bin size, with subsequent points one bin size apart.

Graphing:

The final section is using matplotlib to graph the results. I just take each of the average wait times, average service times and standard deviation of wait times arrays as the y-coordinates and the x-axis array values as the x-coordinates, then create a line graph by connecting all of the points. This graph can optionally be saved as an image and is always shown to the screen.