Nima Mohammadi (95013021) - nima.mohammadi@ut.ac.ir
Prof. Movaghar
Assignment #2

Assignment #2 - MM1K-PS

The simulation keeps two values for each customer that arrives (and not blocked). This pair **(remaining_service_time, remaining_waiting_time)** denotes an individual customer at a specific time. At each iteration, the simulation progresses for **next_event_t** amount of time unit. It then subtracts **next_event_t / current_waiting_count** from the **remaining_service_time** of each customer, as every job receives identical share of processing time. It also subtracts **next_event_t** from **remaining_waiting_time.**

upcoming_job_t is time until the arrival of the next job, that is drawn from an Exponential distribution parameterized by Lambda.

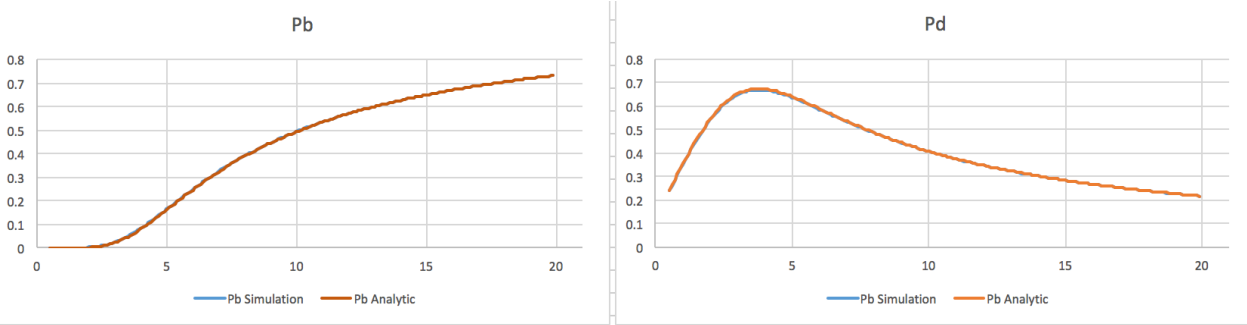
next_event_t determines the simulation time at next iteration. It's value is dynamically calculated by taking the minimum of **upcoming_job_t**, minimum of **remaining_service_time** and **remaining_waiting_time*current_waiting_count**, which correspond to the arrival of next customer, completion of one job, or desertion of queue by a customer, respectively. We choose the duration of **next_event_t in** to be the maximum time it can go without the necessity to handle an event.

sim_counts is the total number of customers who arrive and determines the length of the simulation. **blocked** and **deserted**, respectively, denote the number of customers who was rejected from the queue, as it was full, and the number of customers who left the queue as the duration they could stay and wait has passed without fully being serviced. **theta_type=0** corresponds to fixed-time waiting time while **theta_type=1** indicates waiting time to be drawn from an exponential distribution.

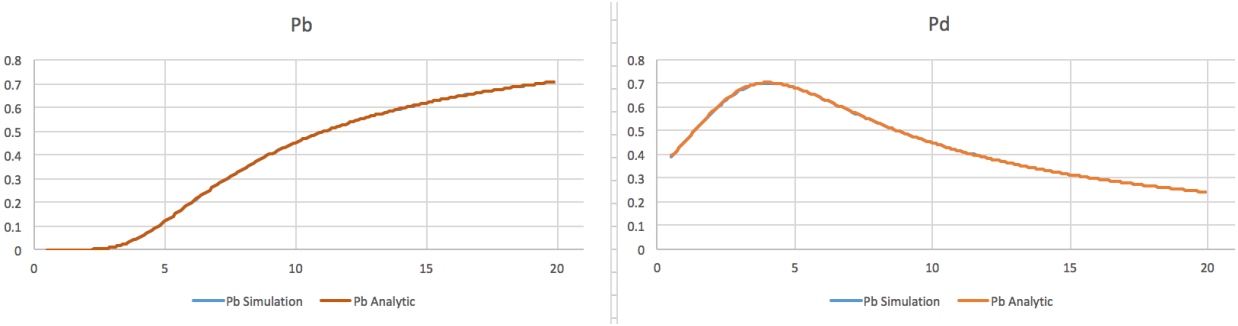
The simulation supports warming up, that is to neglect preliminary results to get more accurate statistics from the simulation.

Given adequate amount of the length of the simulation and warm-up period, and as can be seen from the Excel files and the plots, the simulation reaches a very acceptable error and the curves for analytical and simulated probabilities abundantly match and only differ with a negligible error.

Fixed Waiting Time



Exponential Waiting Time



In both cases probability of blocking customers increases with increasing the arrival rate, hence the strictly increasing curve for P_b. On the other hand P_d increases at first since there are more non-blocked customers reaching the processor. Though it decreases for lambda>~4 as the queue get congested and more customers get blocked.

		P_b Error	P_d Error
Fixed	Max	0.003105943	0.00442664
	Avg	0.000684498	0.001338039
Exponential	Max	0.000697894	0.00360629
	Avg	0.000146395	0.00057329

The code is implemented in C++ (C++11 to be accurate). It has two main functions

```
vector<double> queue_simulator(double lambda, double mu, double theta, int K, int theta_type, int sim_counts, int warmup=100000)

vector<double> closed_form(double lambda, double mu, double theta, int K, int theta_type)
```

which correspond to simulation and analytical computation of the queue, respectively.

Running "**make**" compiles the code into two executable files, namely "**mm1k_runparams.o**" and "**mm1k_analysis.o**". Running "**make run**" runs the first executable (i.e. "**mm1k_runparams.o**").

* **mm1k_runparams.o:** The first one reads the parameters, Theta and Mu, from "**parameters.conf**" file and reports P_b and P_d as Lambda assumes three different values for Theta being drawn of two different random distributions (Fixed and Exp).

* **mm1k_analysis.o:** The second executable also reports P_b and P_d, for the exponential and deterministic arrival distribution and outputs two .CSV files. Running "make run" executes the first executable file generated.

The actual code for the simulator and the analytical computations resides in "MM1K/mm1k.hpp".

As for the absolute and relative error, these are calculated as below:

```
abserr = abs (analval - simval)
relerr = abserr / simval
```

The codes were tested on macOS Sierra with gcc [Apple LLVM version 8.0.0 (clang-800.0.42.1)]. **mm1k_analysis.o** (consisting 40 simulations) took ~15 minutes to run on my laptop.