Hospital Simulation Manual

The source codes provided contain functions to help simulate a hospital with people arriving in a single reception queue and then splitting into multiple check-up rooms.

The simulation has multiple input parameters such as the number of rooms (M), patients arrival rate (lambda), the patients fatigue time (alpha), reception service rate (mu), service rate of doctors in each room and the number of patients.

You can enter the input parameters manually by using **getInputParameters()** in your command line.

By running code segment below a simple input gets created:

```
input.M = 3;
input.lambda = 20;
input.alpha = 1000;
input.mu = 20;
input.rates{1} = [0.5, 1, 1.5];
input.rates{2} = [0.7, 1.5];
input.rates{3} = [2, 3];
input.patient_count = 100;
```

Now by running the funtion **simulate(input)** you can run your simulation and the output will be an object which contains the history of all that has happend in the process.

```
hospital = simulate(input);
```

How to output mean time in system, mean time in queue, and mean bored patients

By running the function below you can see these parameters for patients with and without corona and generally.

```
outputMeanQueueAndSystemTime(hospital);
```

As you can see most of the time spent for patients is actually waiting in queue, by increasing the service rates we can reduce this:

```
input1 = input;
input1.mu = 40;
input1.rates{1} = [4, 10, 5];
input1.rates{2} = [5, 20];
input1.rates{3} = [10, 14];
hospital1 = simulate(input1);
outputMeanQueueAndSystemTime(hospital1);
```

By decreasing the alpha and reducing reception rate you can increase the number of bored people in the queue:

```
input1 = input;
input1.mu = 1;
input1.alpha = 1;
hospital1 = simulate(input1);
outputMeanQueueAndSystemTime(hospital1);
```

Monitoring queue lengths in the system

By running the function **outputQueueLengths(hospital)** multiple outputs appear on screen two figures appear with one of them showing queue length for reception during time, one of them counts all the patients in queue, the other counts the number of infected patients in queue and the other shows the number of healthy people in queue.

In the command line the average queue length of each room and reception is shown.

```
outputQueueLengths(hospital);
```

Simulation accuracy

The function **calcSimulationAccuracy(input)** calculates the accuracy a simulation has on input variables set to **input.**

The funtion **findOptimalPatientCount(input)** uses a binary search implementation to find out the smallest patient count such that the accuracy is at least 0.95.

```
calcSimulationAccuracy(input);
optimal_value = findOptimalPatientCount(input);
```

The effect of service rates on queue length

By running the function **findOptimalRate(input)** multiple simulations will run on output each time the rates (eg. mu and chechup rates) increase by a value exponentially until the average queue time becomes approximately zero.

Then the rates increase by a fixed slope and the average waiting time of all the system is plotted, by running the section below you can see that increasing rates makes the average waiting time decrease.

```
input1 = input;
input1.mu = 0.01;
input1.rates{1} = [0.004, 0.01, 0.005];
input1.rates{2} = [0.005, 0.002];
input1.rates{3} = [0.01, 0.0014];
findOptimalRate(input1);
```

Histogram of time in system, waiting time and service time

The code sections below plot a histogram that indicates the time that each type of patient (infected, healthy, general) has spent in service, in queue, or in system.

System time:

```
outputSystemTimeHistogram(hospital);
```

note: bored people are not included in this output.

Queue time (Waiting time):

```
outputQueueTimeHistogram(hospital);
```

Service time:

```
outputServingTimeHistogram(hospital);
```

Ploting number of patients in system during time

The function **outputNumberOfPatientsInSystem(hospital)** a 3 graphs are shown each indicating the number of patients in system, for all, infected and healthy accordingly.

```
outputNumberOfPatientsInSystem(hospital);
```