M.Sc. in Mathematics (Finance)

2016/2017, 1º semester

Computational Methods

Project 1 – Study of a Call Centre

1. Introduction

The management of a company decided to create a call centre to attend remotely (by telephone) its clients. To do so a study was made that showed that the calls from clients

a) arrive following approximately an exponential distribution; and
b) require a service time that is adequately modelled by a Erlang distribution.

Remind that the probability density function of an Erlang distribution is

$$\mathrm{Erlang}(t;\, k, \mu)\ = (t^{k-1}\, e^{-t/\mu})/\ (k-1)!\ \mu^{k})$$

where parameter **k** is a (small integer) and **μ** a time constant (you may notive that an exponential distribution is the special case of an Erlang distribution with **k = 1**).

The service is organised such that whenever a call arrives and all assistants are occupied it is left on hold if no more than **c** calls are already on hold, otherwise the call is rejected.

2. Objective

More specifically, your goal is to simulate a system and check the quality on the service provided, according to the data obtained from file "**system_data.txt**" (available from the web page) with the following type of information (where the place holders <…> denote the actual values provided

```
DATA
 - Number of calls: <ccc>;
 - Mean time between calls: <sss> seconds;
 - Number of assistants: <aa>
 - Parameters for Erlang modelling of assistants:
          k = <kk> and mu = <sss> seconds;
 - Maximum number of calls on hold: <hh>.
```

Upon simulating the system with the parameters read from this file, the results obtained should be written in a file "results.txt" with the following information (replacing the place holders <…> should be replaced by the actual values obtained I the simulation).

```
REPORT
 - <X> clients called for assistance, <Y> of them being rejected.
 - The first call was at 8:00 and the last finished at <hh:mm:ss>.
 - For those attended, the average waiting time was <mm:ss> minutes.
 - The average length of the waiting queue was <L>.
 - The assistants were busy <pp.pp>% of the time.
```

3. Final Report

You should write a small report explaining how you carried out your simulation, namely:

a) The data structures that you used to model the system;
b) The events that you considered;
c) The functions you used to simulate the timing of the events;
d) The functions you used to model the whole system;

The report, as well as the files with your code and results must be sent by email to the lecturer (pb@fct.unl.pt) with subject **Project_MC_1_by_XXXXX+YYYYY** (where XXXXX and YYYYY are the numbers of the students - max 2 per group), **no later than** Friday, 30 December at 23:59,

4. Implementation Notes:

1. Your program should have a main function with signature

   ```
   function simulate_system(fileIn, fileOut)
   ```

2. Decompose your program in an adequate number of functions (possibly recursively) and make sure their interfaces are appropriate, namely their signatures include the needed parameters.
3. Although you may not solve the problem completely, your program should at least read the specification file (with name *fileIn*) and write the results (possibly from a simplified system) in the output file with name *fileOut*.
4. The simplest system you should implement is one similar to that studied in the classes, i.e.
   i. with a single server ;
   ii. a queue of max length of 1;
   iii. a fixed service time;
   iv. reporting the number of rejected clients; and
   v. the time the last attended call has finished.
5. More complex systems (possibly adapted from the simplest one) should include some or all of the following improvements:
   a) An Erlang distribution to model service time;
   b) An arbitrary number of servers;
   c) An arbitrary max queue length (also read from the data file), or both;
   d) The average waiting time of the accepted calls;
   e) The average length of the waiting queue;
   f) The percentage of time each assistant was busy.
6. To debug your program,
   a) As usual, you should do unitary tests for each of the implemented functions
   b) For the simulation, test systems with an increasing number of requests, starting with a single request.