Lab4 - Transient elimination and batch means

Martina Martini s306163
Politecnico di Torino

Assignment Re-adapt the code you have developed within lab L1 to simulate a M/G/1 queue. Add procedures to eliminate the transient in run time (by inventing a proper algorithm) and to evaluate confidence intervals by adopting a batch means technique.

Consider the following distributions for service time:

- Deterministic equal to 1.
- Exponential with average 1.
- Hyper-exponential with two branches; average=1 and standard deviation=10.

For each of three considered systems produce a plot in which you report the average queueing delay vs lambda.

Choose the following vales for lambda: 0.2, 0.4, 0.6, 0.8, 0.9, 0.95, 0.99, 0,999

Compare the results you have got with theoretical predictions given in Pollaczek-Khinchine formula.

Verify experimentally the Little formula.

At last, consider a M/M/1 with finite waiting line N=1000, and produce two plots: average delay vs lambda; dropping probability vs lambda. Use the following values for lambda: 0.2, 0.4, 0.6, 0.8, 0.9, 0.95, 1, 1.1, 1.2.

Deliver the code and a brief report in which you report the results you got (with a brief comment) and you describe: i) the algorithm you have implemented for the transient elimination; ii) how you compute confidence intervals; iii) the parameters of the hyper exponential and how you obtained them;

1 Input parameters, main assumptions and procedure

1.1 Requirements

The M/G/1 queue provides three basic assumptions, from which we can start:

- The arrival times are Markovian distributed, which means that distributions like Poisson or Exponential are suitable for this kind of system.
- The service times are Generical distributed, which means that there is no restriction in the choice of the distributions - in our simulation, the service times are evaluated in three cases:
  - deterministic ($service\_time = 1$),
  - exponential ($service\_time \sim Exp(SERVICE\_RATE)$ where $SERVICE\_RATE = 1$)
  - hyper-exponential ($service\_time \sim HyperExp(p, \lambda_1, \lambda_2)$), where p is the probability that the hyper-exponentially distributed r.v. would take the form of the exponential distribution with rate λ_1 and $(1 - p)$ with rate λ_2.

- The number of servers is equal to 1.

To conclude the discussion about the Hyper-exponential distribution, we can notice that for construction, ideally, if we take a r.v. $X \sim Exp(\lambda_1)$ and a r.v. $Y \sim Exp(\lambda_2)$, then our hyper-exponentially distributed r.v. $Z \sim HyperExp(p, \lambda_1, \lambda_2)$ would take the form of $Z = pX + (1-p)Y$. Moreover, such a variable Z would have an expectation equal to

$$\mu = p\mu_x + (1-p)\mu_y$$

and a variance

$$\sigma^2 = p\mu_X^2 + (1-p)\mu_y^2 + p(1-p)(\mu_x - \mu_y)^2$$

. Consequently, in the simulation, the hyper-exponential service times are constructed by taking randomly two lambdas from the given arrival lambdas and then by applying the reasoning overcited to generate the r.v. Finally, the information about variance and mean are exploited for the computations for the Pollacezk formula, explained later on.

As mentioned above, the arrival times are considered exponentially distributed, so the arrival rates are given and they are taken one by one from the following list: 0.2, 0.4, 0.6, 0.8, 0.9, 0.95, 0.99, 0.999.

To see how the system behaves in terms of average delay based on the choice of the lambda, one plot for each arrival time and each service time distribution is shown (Fig.1).

In terms of computations, the delay averages are computed more than one time: in fact, the simulation was run a number of times equal to NUM_OF_SEEDS and each time a different seed was used. This procedure allows us to produce the desired number of samples as independently as possible one by another. Also, the chosen seeds are very distant one from another for the same independence purposes. Given that, the cumulative average delays are computed and shown based on the different lambdas and the different service time distributions.

In addition, the Pollacezk formula and the Little's Law are printed and compared with the cumulative average graph. The theoretical Pollacezk-Khinchine formula about the mean waiting time is expressed as

$$W = \frac{\rho + \lambda\mu\sigma^2}{2(\mu - \lambda)}$$

and considers that in all the three cases the mean μ is equal to 1. Instead, the variance σ^2 changes over the three distributions: it is 0 for the deterministic, 1 for the exponential and 10 for the hyper-exponential one. It follows that also in the computation of the Little's Law ($L = \lambda W \rightarrow W = \frac{L}{\lambda}$), where L is the average number of clients in the system , W is the mean waiting time and λ is the arrival rate.

1.2 Transient detection

In this first part of the simulation, a transient elimination algorithm is performed to not have to consider the unstable phase of a system when a quantity is to be evaluated.

Starting from the cumulative average plot, we can notice that in all the cases, the transient phase is present: to detect it the following algorithm was used:

Algorithm 1: Transient Detection

```
Divide the cumulative in batches of
  size $BATCH\_SIZE$
for each batch do
    compute the variance of the points
      in it
    if variance > mean variance then
        select the indexes of these
          points
        fill the transient indexes' list
    end
end
return transient indexes' list
```

Once the transient phase is detected, shown in figure Fig.4 and stored, we proceed by taking under consideration just the steady-state and by computing the batch means method.

1.3 Batch means

Basically, the adopted procedure is shown in the pseudo-code that follows. Moreover, the experimental mean is computed and shown in the plot to better understand if the batch means are correctly calculated. Notice that the batch means are drawn in the correspondence of the middle point of the relative interval (Fig.5).

Algorithm 2: Batch means and CI

```
Store the index of the last transient
  point
From the cumulative, select just the
  points from the last transient on
Select just the indexes of the
  non-transient points.
Choose the $BATCH\_SIZE$ and the
  confidence level
Divide the cumulative in batches of
  size $BATCH\_SIZE$
for each batch do
    compute the mean of the points
    add it to the list
    compute the standard error
    compute the t-student value
      ($BATCH\_SIZE$ dof)
    compute the margin of error
    compute the lower bound
    add it to the list
    compute the upper bound
    add it to the list
end
return means, lower bounds, upper
  bounds
```

1.4 M/M/1/b queue

Lastly, the case of M/M/1 queue is taken under consideration. This system provides Markovian arrival times, Markovian service times, 1 server and a limited waiting line (here $b = 1000$). For this case, the arrival rates are fixed to 0.2, 0.4, 0.6, 0.8, 0.9, 0.95, 1, 1.1, 1.2 and the analyses regarding the average delays and dropping probability are shown in the plots (Fig.2 and Fig.3).

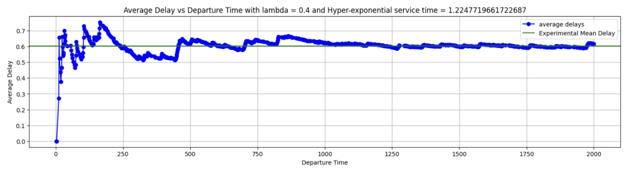

Figure 1: Average delay plot (M/G/1)

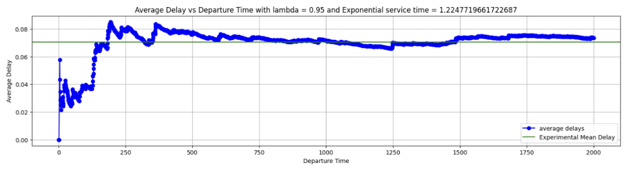

Figure 2: Average delay plot (M/M/1)

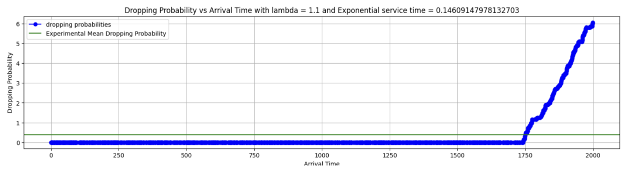

Figure 3: Dropping Probability (M/M/1)

2 Output metrics and further considerations

Firstly, the average delays increase with the simulation time. Then, we can notice that the system always depicts a transient phase before becoming stable. Unfortunately, in some of the combinations of service distribution-arrival rates the Pollacezk mean does not match with the experimental one because of the unstable transient phase. Also, Little's Law provides different results in terms of estimated mean waiting time. Nevertheless, the transient deletion algorithm and the consequent batch means algorithms return great results: the former detects correctly the warm-up phase and selects just the steady-state, the latter computes the batch means, which perfectly lie on the experimental one.

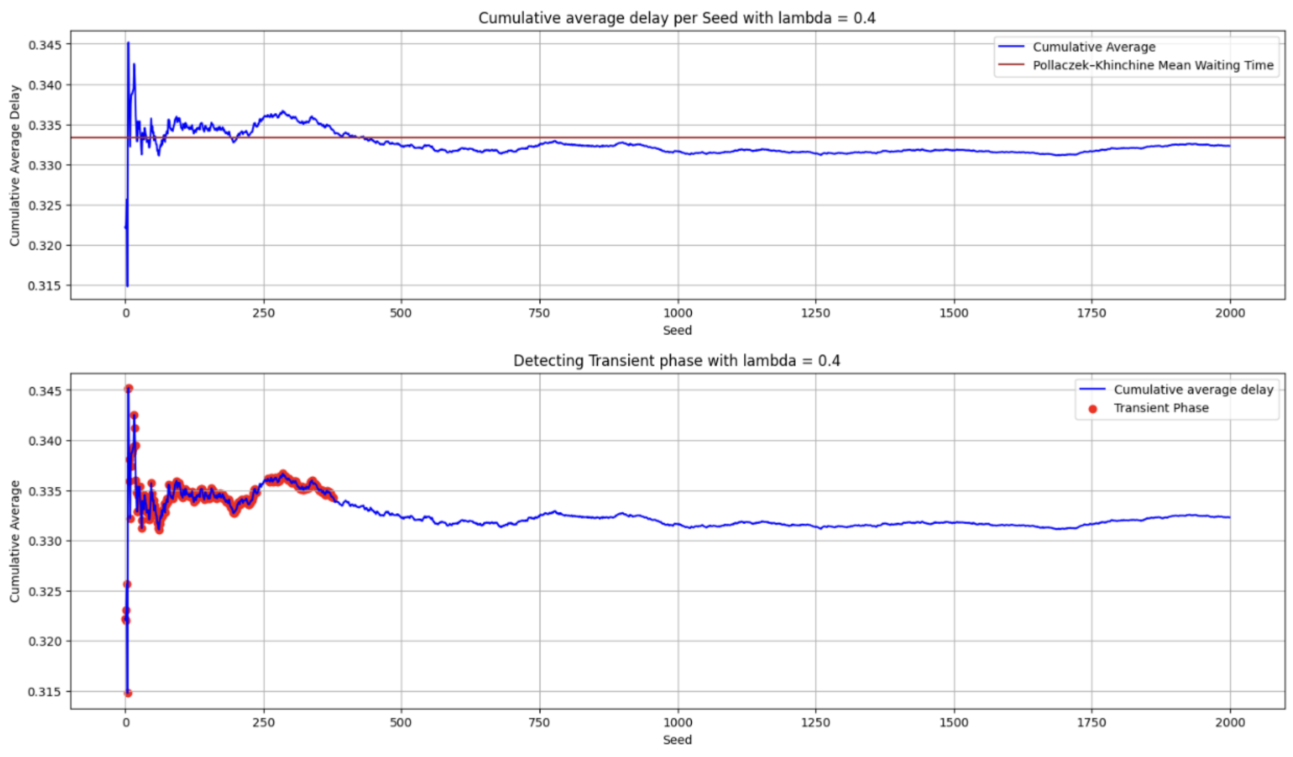

Figure 4: Pollaczek comparison and transient phase detection

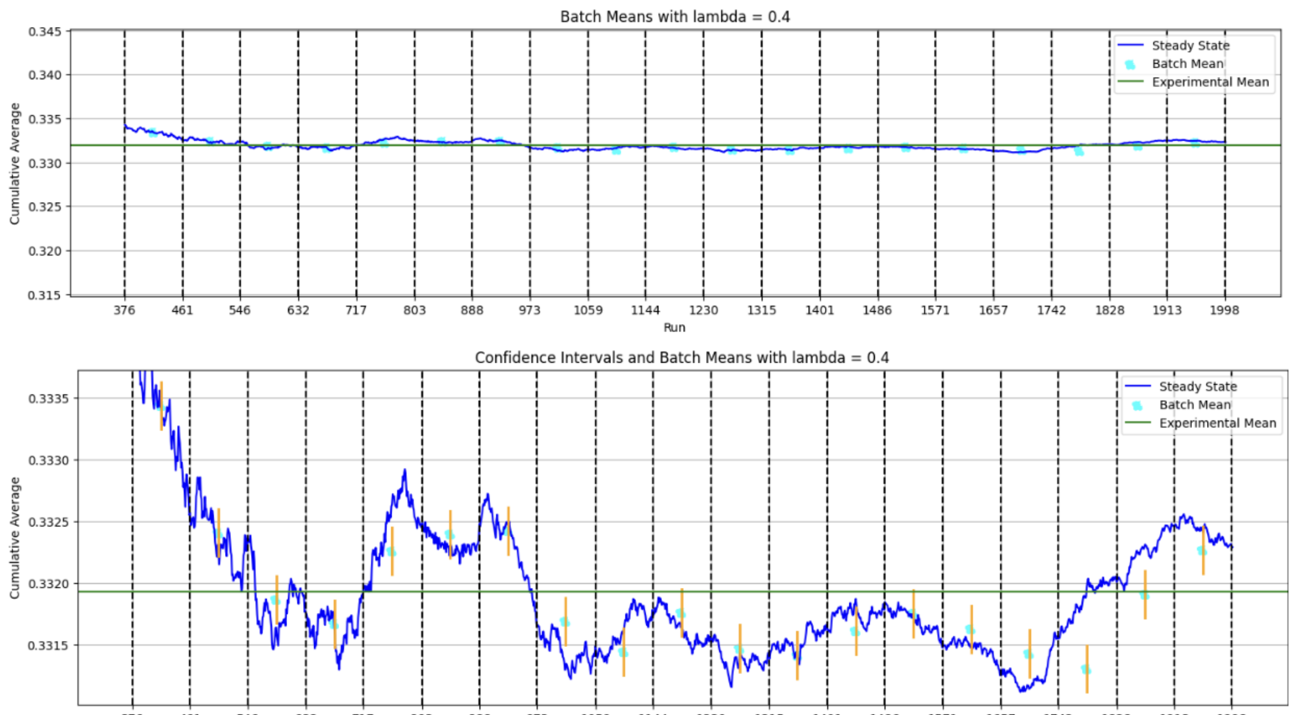

Figure 5: Zoom on the batch means method (the orange lines are the confidence intervals)