University of Pisa

Master Degree in

Computer Engineering

Deficit Scheduler

Gabriele Lagani	501505
Luca Maddau	422259
Indrit Kertusha	465141

Introduction

This project deals with the study of a system made up as follows: there is a source that produces and sends jobs and a server that receives the jobs and put them in a FIFO queue. The server performs the following operations:

- It serves the jobs in the queue atomically, service occurs within a period of time called *turn*, whose expected duration is t_q
- If the next job cannot be served within the end of the turn, the turn is terminated in anticipation and the remaining time, called *deficit* and denoted with t_d, will be added to the duration of the next turn (which consequently will last $t_q + t_d$)
- If the server finds the queue empty, the turn is terminated in anticipation
- When a turn is terminated, the server goes into a state of inactivity for a period of time t_v called *vacation*, at the end of which a new turn is started

The purpose of our project is to study the response time of the system as a function of the turn length. In order to do this, the following assumptions have been taken:

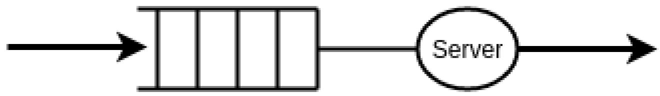
- Jobs interarrival times are IID RVs with mean t_i
- Jobs service times are IID RVs with mean t_s
- Vacation times are IID RVs with mean t_v
- All the above RVs are mutually independent
- The channel between source and server is ideal
- t_s , t_i, t_v, $t_{q,}$, are the only non negligible times in place (e.g. we consider negligible the time required by the server to insert a new job in queue or to switch from vacation to turn)

The analysis was focused on the following scenarios:

- deterministic interarrival times, service times and vacation time, in order to become familiar with the system and test the behavior of its implementation
- exponentially distributed interarrival times, service times and vacation times, in order to obtain more general and realistic insights

System model

With respect to the assumptions made, the system can be modeled as a queueing network, as shown in the following picture



The server schedules the jobs on the base of the protocol described before.
This model was implemented on the Omnet++ simulation framework.

It is useful to introduce by this very moment the stability condition of the system, since it will be necessary for choosing the parameters of the simulated scenarios properly.
A first condition is of course that the interarrival time has to be larger than the service time, like in a classic M/M/1 system without vacation: if the latter isn't stable, a fortiori our system isn't. Hence:

$$t_i > t_s$$

By the way, this is not enough: in fact even if the condition above mentioned is respected, our system could be unstable because the turn is too short or the vacation too long. More in general, without going deep into theoretical aspects, in order to be stable the system has to be able to serve in one turn more jobs than it receives during the previous vacation and the turn itself. In formula:

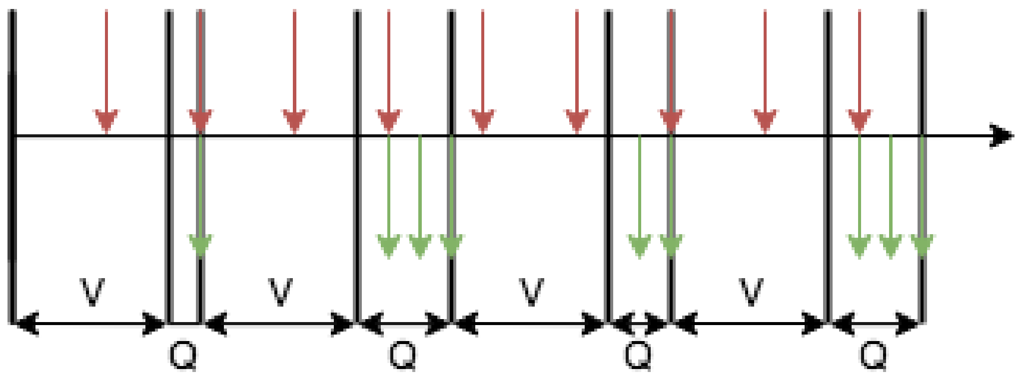
$$\frac{t_q}{t_s} > \frac{t_v}{t_i} + \frac{t_q}{t_i}$$

Verification

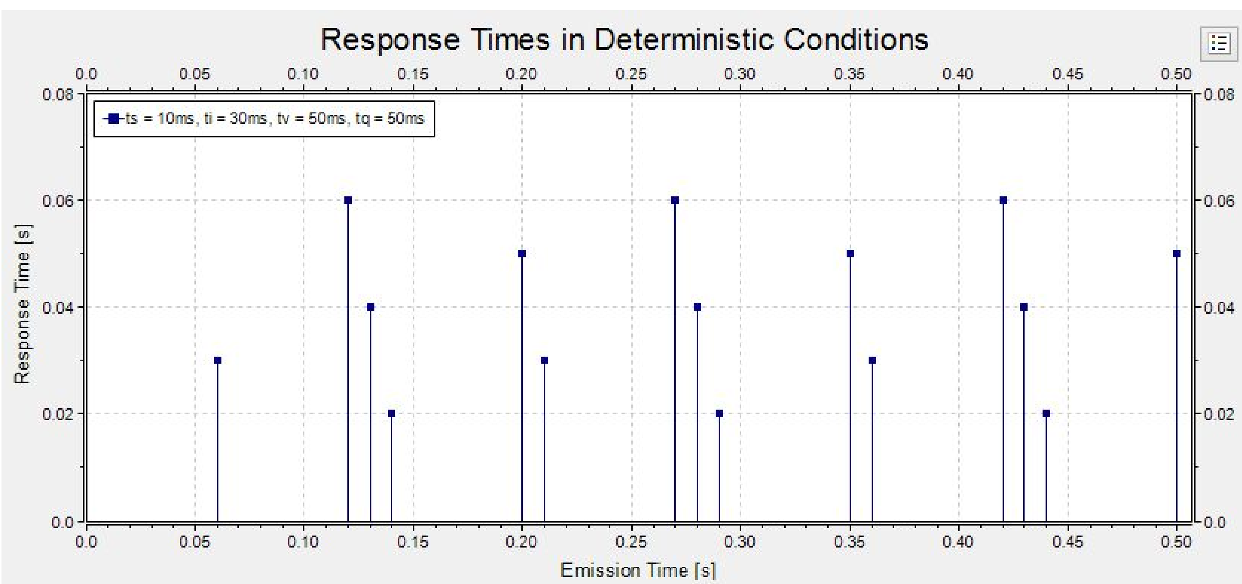
In this section a series of tests, performed in order to ensure that the implementation is coherent with the system model, are presented. The first part of the section shows the tests made running the network under a deterministic regime: the output of the program must be the same of the one that can be manually (or, in general, mathematically) computed according to the abstract model. In the second part, the code is tested under a stochastic regime.

Deterministic tests

Here is a graph showing the expected behaviour of the system under the conditions :
t_s = 10ms, t_i = 30ms, t_v = 50ms, t_q = 50ms.



The expected response times measured are, in sequence: 30ms, 60ms, 40ms, 20ms, 50ms, 30ms and so on, periodically. The system output matches our predictions, as shown in the following graph.

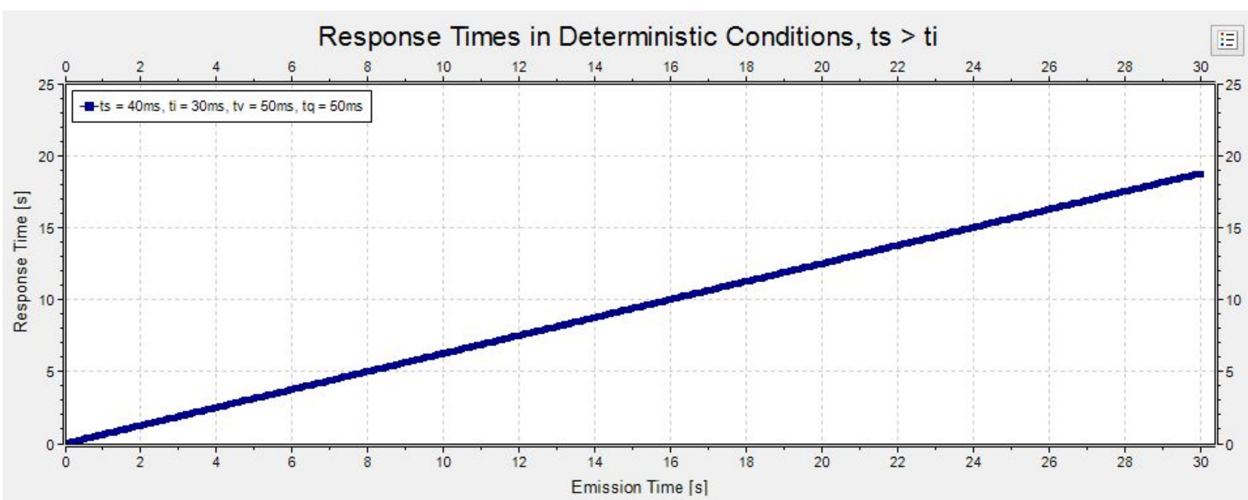


Other similar tests were performed with various configurations of the parameters. For the sake of brevity we synthesize the results in a table showing the parameters set up, the average response time expected and the one actually obtained in each scenario.

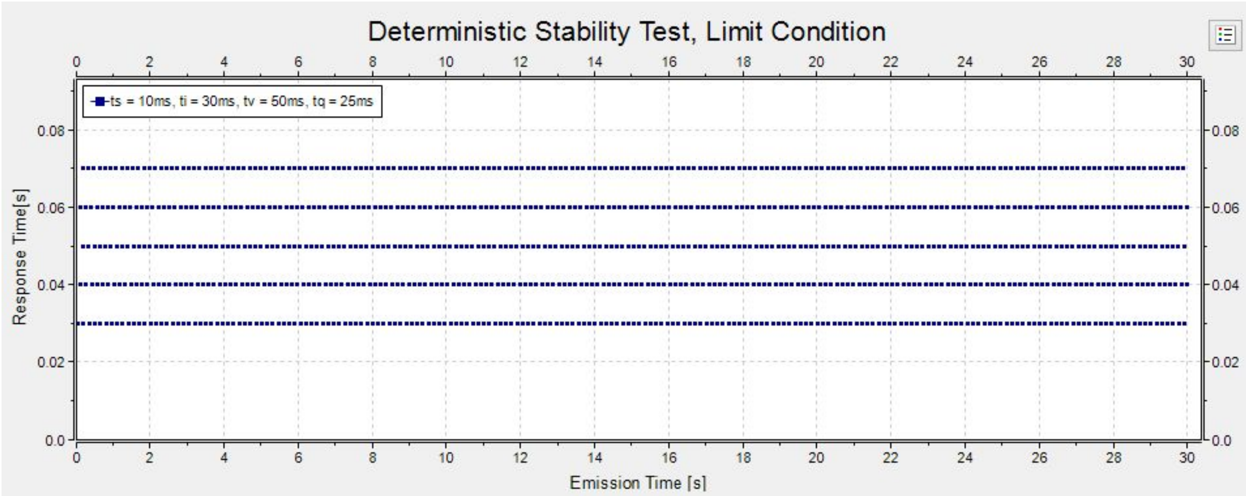
t_s	t_i	t_v	t_q	Expected t_r	Actual t_r
10ms	30ms	50ms	50ms	40ms	40ms
20ms	80ms	100ms	100ms	80ms	80ms
10ms	30ms	10ms	20ms	20ms	20ms
20ms	80ms	20ms	10ms	60ms	60ms

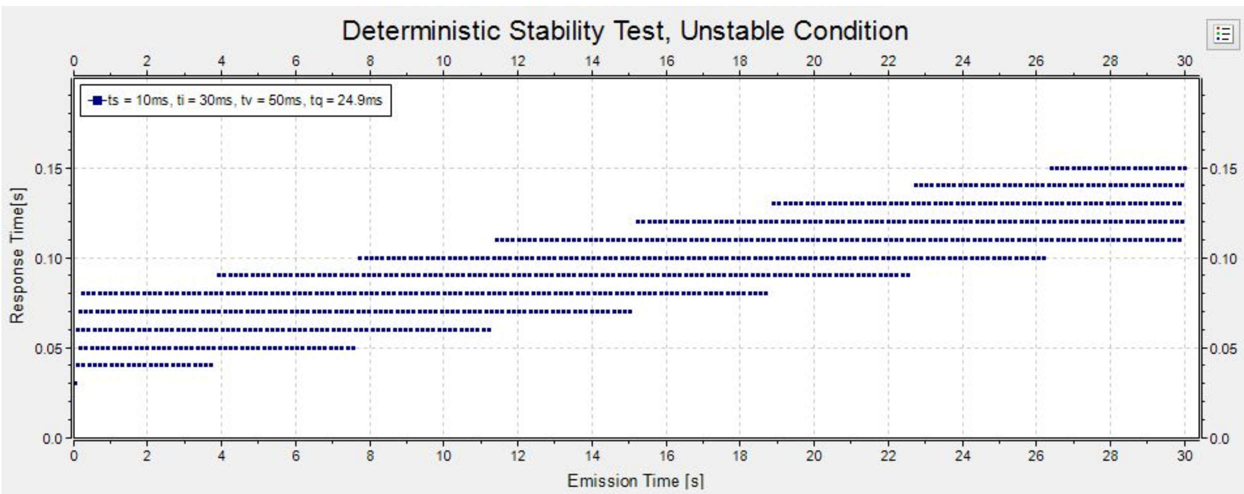
The system was also tested against the stability conditions:

1. For values $t_s >= t_i$, the system is not be stable and the response times grow indefinitely.



2. The system has been tested also in the neighbourhood of the second stability condition, making t_q vary slightly. We can see the system going from a stable to an unstable behaviour while we move through the following scenarios:
    1. t_s = 10ms, t_i = 30ms, t_v = 50ms, t_q = 25ms (limit of stability condition, which in deterministic regime makes the system stable)
    2. t_s = 10ms, t_i = 30ms, t_v = 50ms, t_q = 24.9ms (unstable condition)

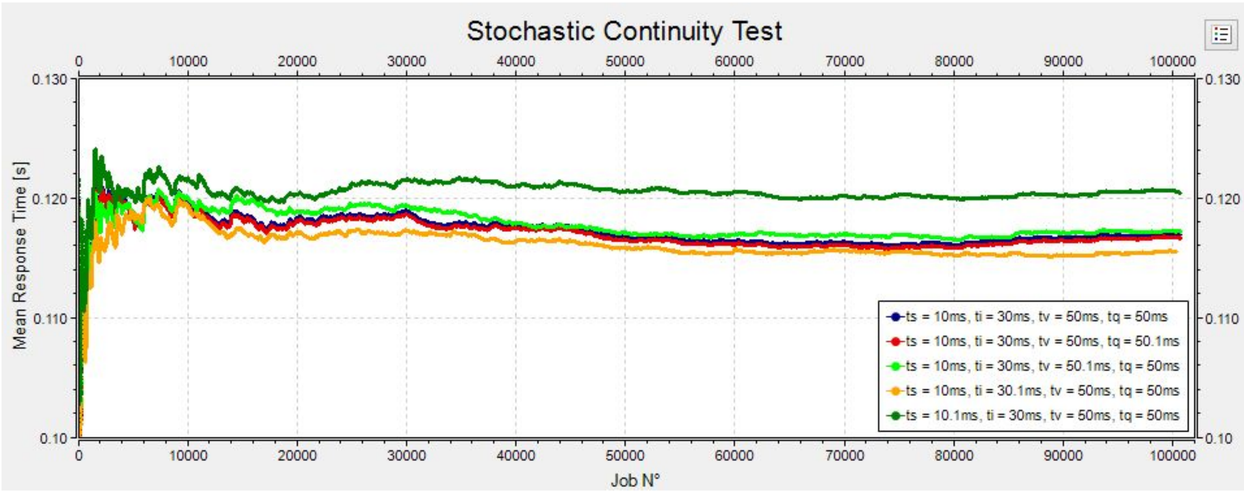




Stochastic tests

The previous tests have given a positive feedback about the model implementation. The following tests are made running the system under a stochastic regime, where interarrival, service and vacation times are exponentially distributed with means t_i , t_s and t_v. Each of the following scenarios was simulated over 10 independent repetitions.

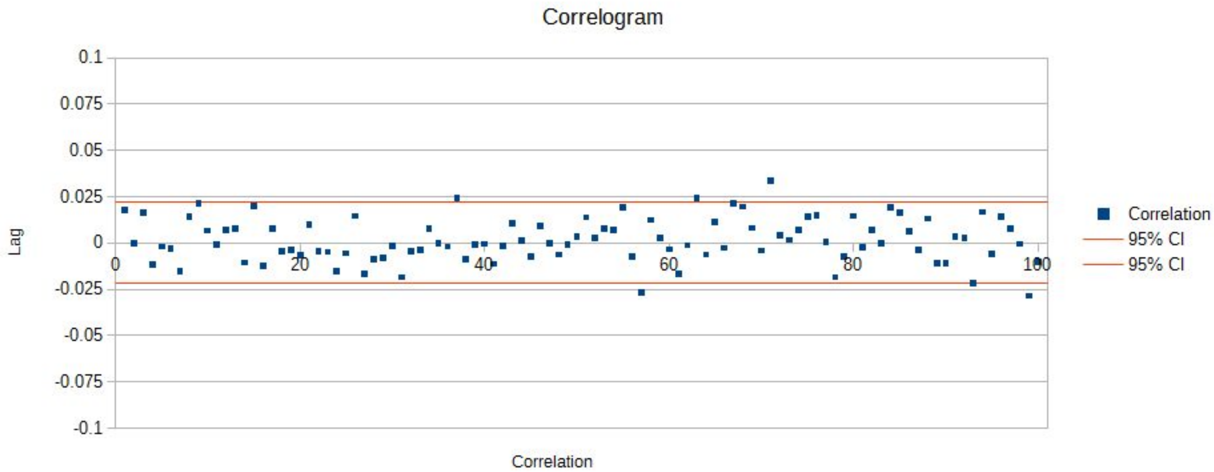
Under a stochastic regime it is useful to run a continuity test. We have to be careful while doing this since the more the condition of the system approaches the limit of the stability condition, the more the system should work in saturation regime, in which a continuity test may be difficult to run. Considering this observation, let t_s = 10ms, t_i = 30ms, t_q = 50ms, t_v = 50ms and increase by 0.1 one factor at a time. By measuring the mean response times in the queue, what should happen is that they does not change dramatically from one scenario to the other.



The following table includes statistics for each scenario. It is clear that the system reacts pretty much the same, as conjectured, since a relative input variation of the order of 10^{-2} causes an output variation of the same order.

t_s	t_i	t_v	t_q	E[R]	95% CI for E[R]
10ms	30ms	50ms	50ms	0.11678	[0.11610, 0.11745]
10ms	39ms	50ms	50.1ms	0.11651	[0.11584, 0.11719]
10ms	30ms	50.1ms	50ms	0.11716	[0.11649, 0.11783]
10ms	30.1ms	50ms	50ms	0.11538	[0.11472, 0.11603]
10.1ms	30ms	50ms	50ms	0.12012	[0.11942, 0.12082]

It must be precised that, before computing statistics, a test to ensure with 95% confidence that the samples were IID has been made. For this purpose, subsampling was performed and correlograms were plotted. As an example, a correlogram of the response times sample is shown in the following graph.

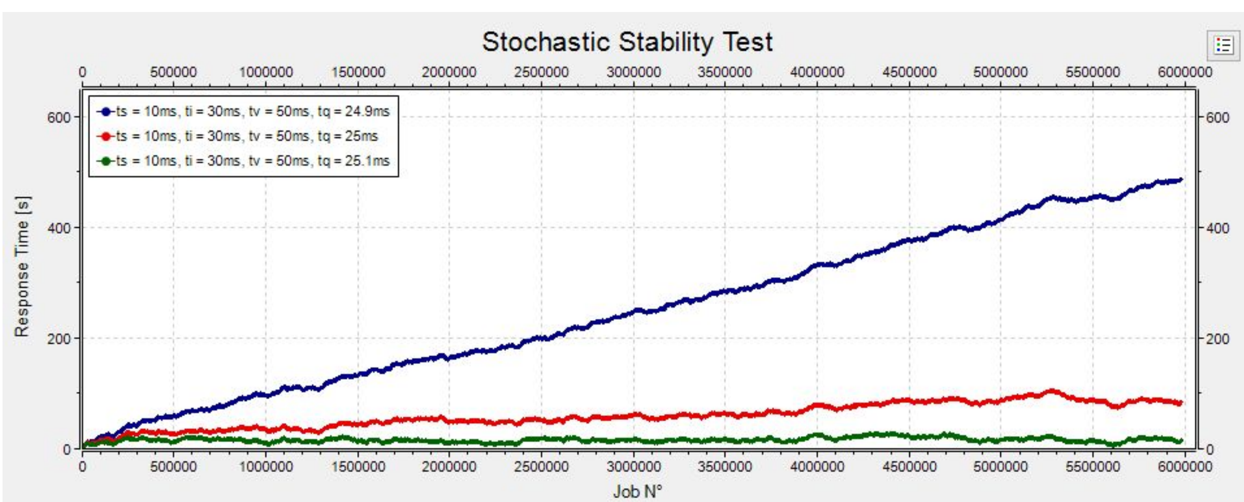


As in the deterministic regime case, the system was tested again in the neighbourhood of the limit of the stability condition, in the following scenarios:

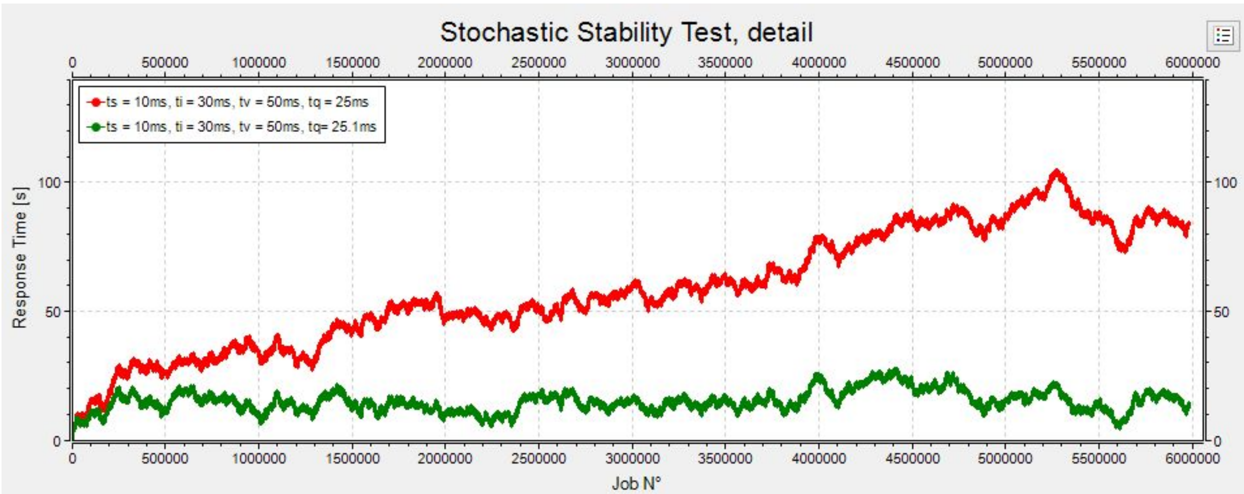
1. t_s = 10ms, t_i = 30ms, t_v = 50ms, t_q = 24.9ms
2. t_s = 10ms, t_i = 30ms, t_v = 50ms, t_q = 25ms
3. t_s = 10ms, t_i = 30ms, t_v = 50ms, t_q = 25.1ms

The simulations have been run for 180,000 s, because the behavior of such a stressed system may be clear after a very long time. In order to obtain the following graphs, we used the merge function of Omnet, to merge the repetitions of each scenario, and finally we applied the moving average.

We can clearly see that in the first case the system is unstable. The second case is also unstable, since we are at the limit of the stability condition.

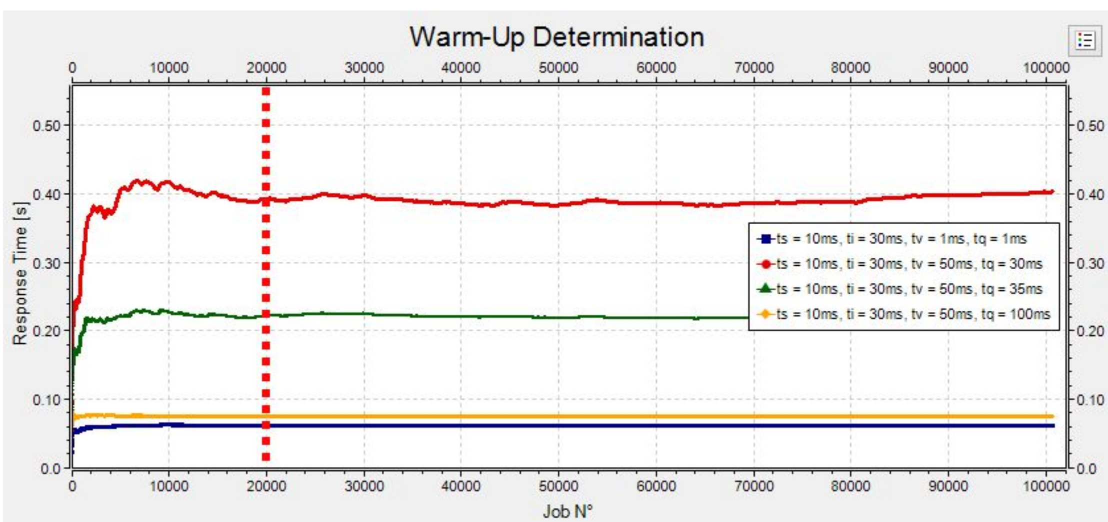


The following graph shows in more detail the difference between the last two scenarios, where we can see the instability of the second case. If we keep increasing the simulation time the response times will keep growing.



Warm-up and simulation duration determination

The length of the warm-up depends on how close the system is to saturation: the more the scenario is close to the limit of the stability condition, the more is the time that the system needs to reach the steady state. Therefore, in order to compute a warm-up time coherent with any kind of situation, various scenarios with different stress conditions were tested and warm-up time was set in such way to guarantee even to the slowest scenario to reach the steady state.



As shown above, the warm-up for the simulation campaign has been carefully set to 600 seconds (corresponding to 20000 jobs).

We also made some considerations about the simulation duration. Assuming that we want to compute 95% CIs for the average response time with a precision of the order of 10^{-3}s, using the formula

$$n = \left(\frac{z_{\alpha/2} S}{r\overline{X}}\right)^2$$

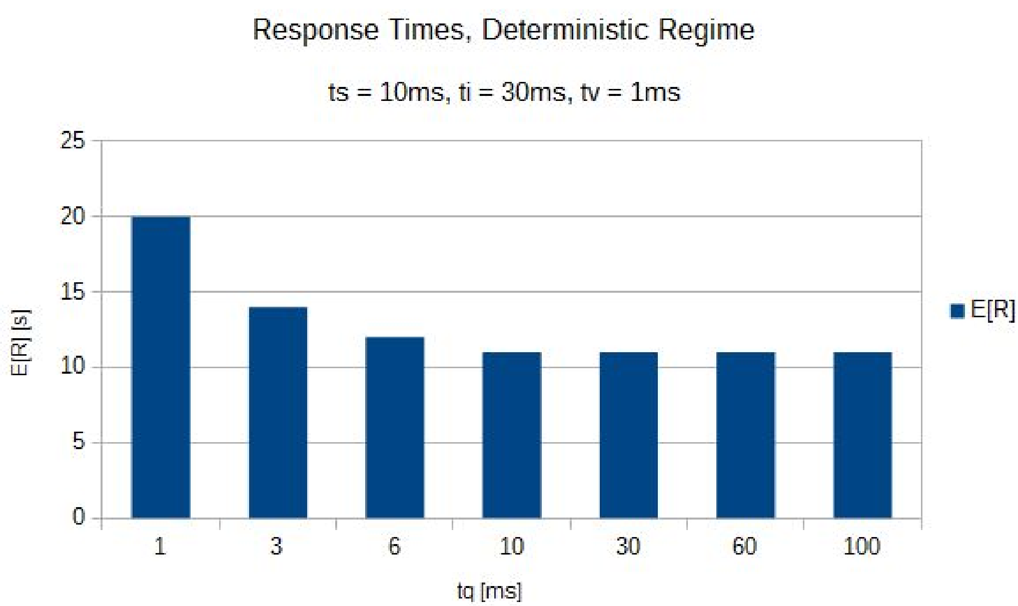
with the denominator of the order of 10^{-3}, $z_{0.025}$ of the order of 10^0, S of the order of 10^{-1} (as we estimated with preliminary simulations), we obtained n of the order of 10^4. Last, considering the necessity of a future subsampling, we cautiously opted for a time that allowed 10^5 observations. On the base of the scenario, we could compute the simulation time needed just by multiplying 10^5 by the interarrival time used in that scenario.

Results

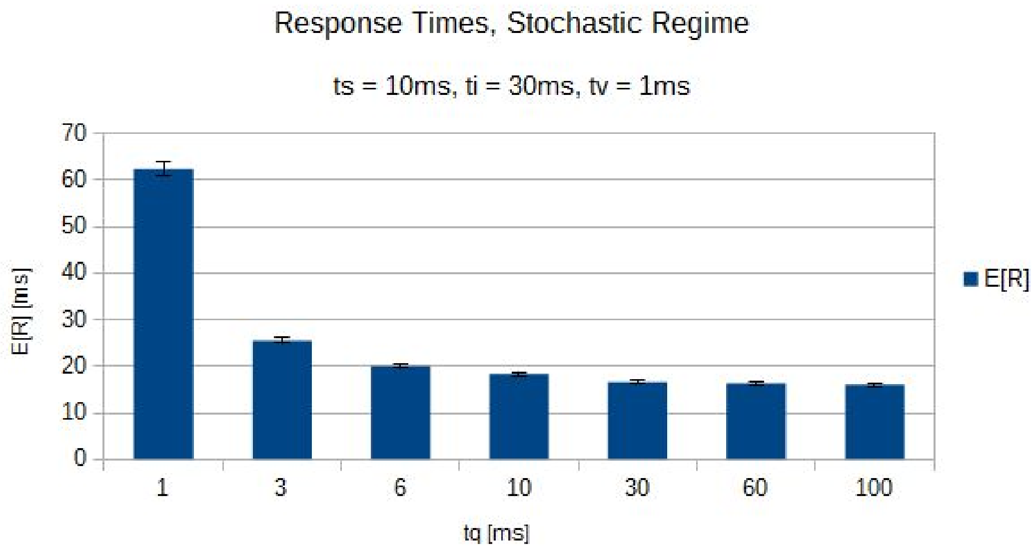
A first simulation was performed in order to study the behaviour of the system in deterministic regime (having care to verify that the results obtained met our expectations before switching to stochastic), having t_q varying from 1/10 to 10 times the service time. We set up the following configuration:

t_s = 10ms, t_i = 30ms, t_v = 1ms, t_q = 1 : 100 ms

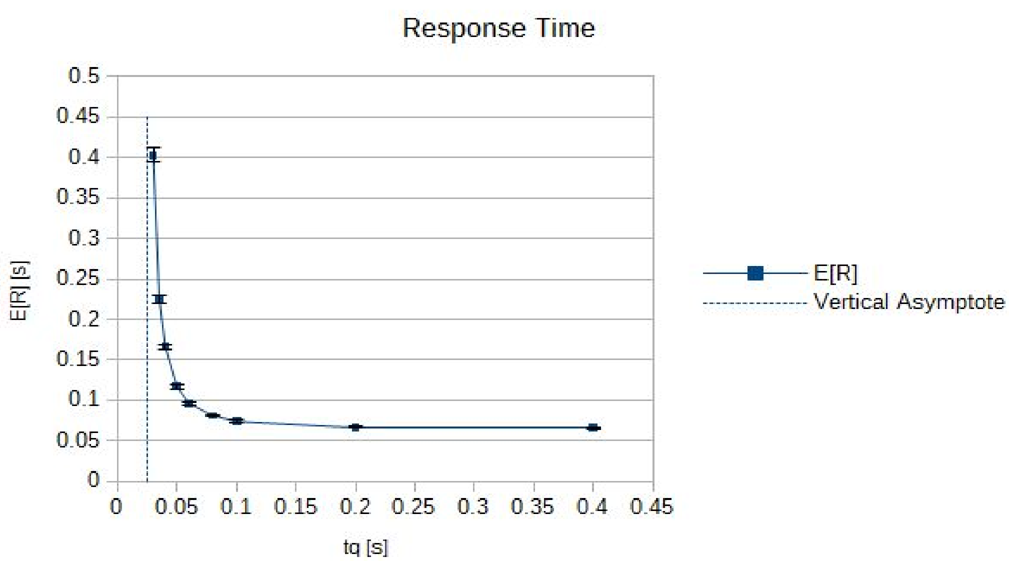
The choice of such a small vacation was made for the purpose of having the system stable for any value of t_q. The following graph shows how the response time gets smaller while t_q increases, but further increments of the turn length produce a less evident improvement in the performance of the system.



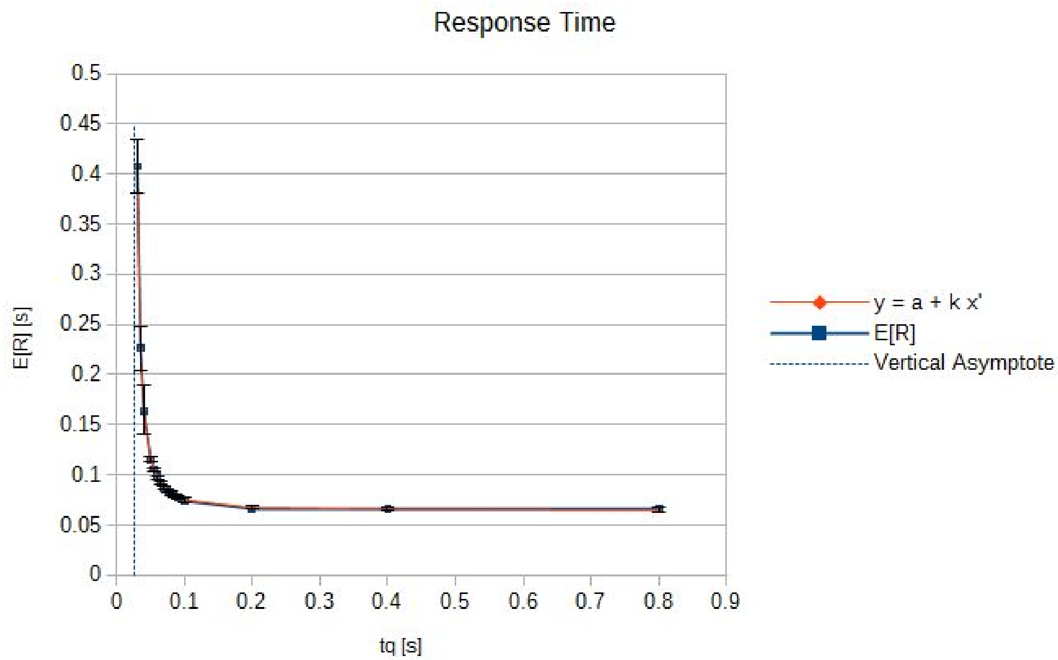
The second step of our simulation campaign consisted in studying the same scenarios as in the previous case, but in stochastic regime. Results about the average response time obtained in each scenario are shown in the following chart.



Since we were also interested in studying the behaviour of the system in a setup with a larger vacation and larger values of the turn length, we simulated also some scenarios with the following values for the parameters: t_s = 10ms, t_i = 30ms, t_v = 50ms, t_q = 30 : 10000 ms.
In the following graph, the mean response time obtained in each scenario is shown as a function of t_q. For reasons concerning the scale of the graph, we haven't plotted the points corresponding to too large values of t_q.



Again we have noticed that when t_q increases, the response time decreases, but the more the increment, the less evident is the benefit. This is due to the fact that whenever the system finds the queue empty it goes on vacation; of course when the turn gets larger it is more probable that the system manages to empty the queue, hence it is more probable that the turn will be interrupted in anticipation and this implies the manifestation of a smaller benefit.



The orange curve in the above graph is a regression model used to describe the behaviour of the system as a function of t_q. In order to compute it we first gathered more points around the knee (since this is the most critical zone), and then we performed a linearization through the following transformation:

$$x' = \frac{1}{x-b} \qquad x \le 40ms$$

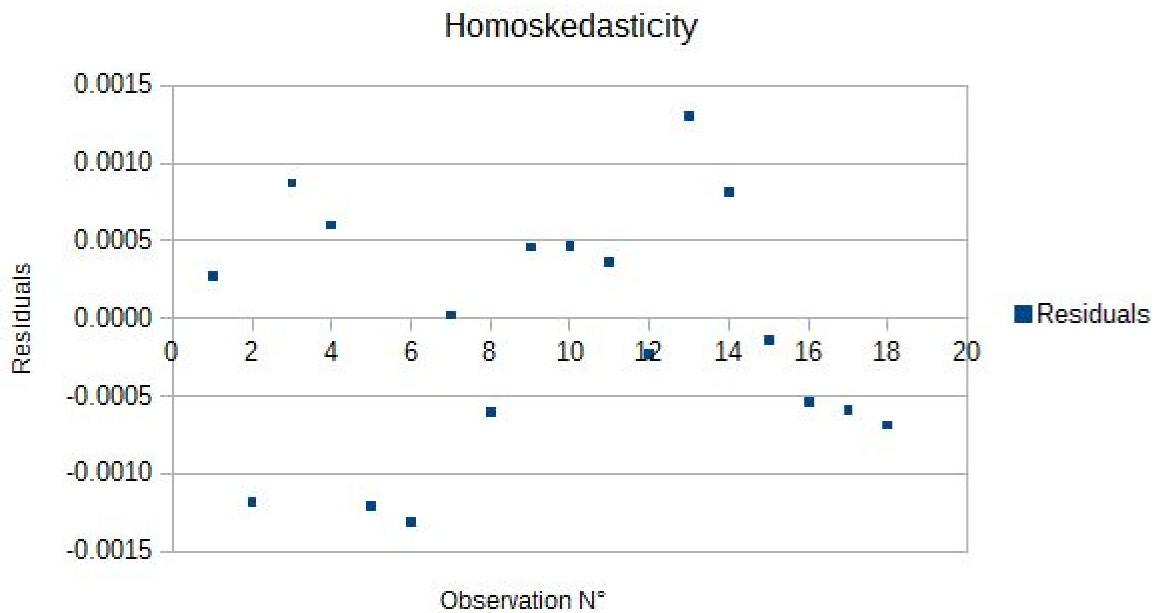
$$x' = \frac{1}{x^2-b^2} \qquad x > 40ms$$

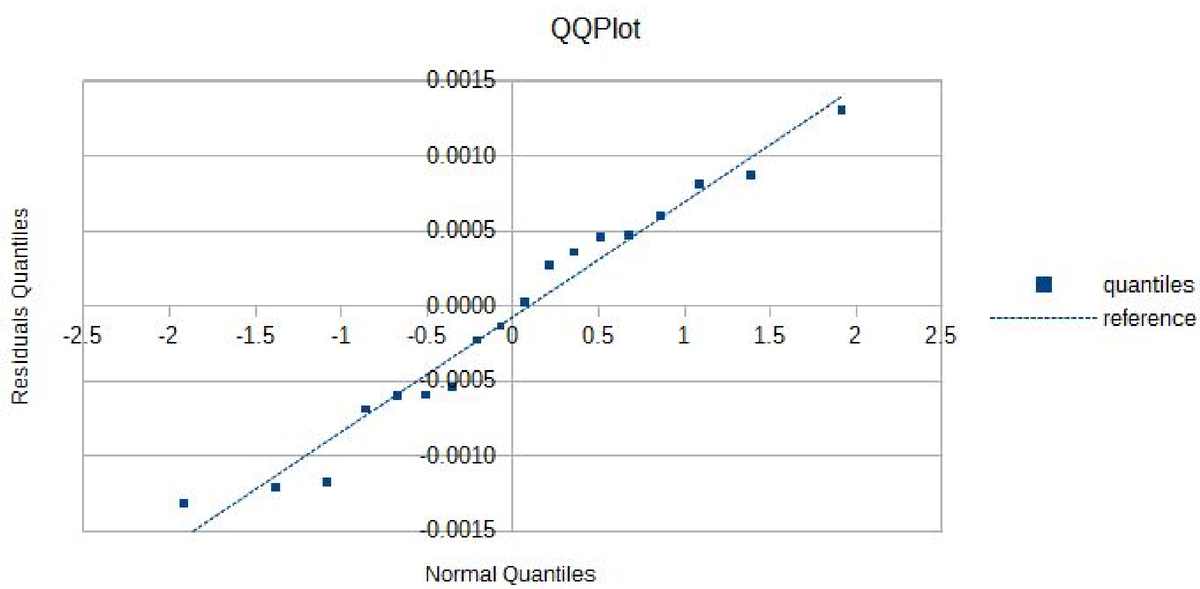
Where b is the vertical asymptote, corresponding to the value of q for which the system becomes unstable, i.e. 25ms

Then we computed a regression model in the form y = kx' + a with the linear regression techniques, obtaining what follows:

x	a	95% CI for a	k	95% CI for k
$\le$ 40ms	0.043355	$a \mp$ 0.013625	0.001823	$k \mp$ 193.08E-06
> 40ms	0.065025	$a \mp$ 0.000124	0.000095	$k \mp$ 2.77E-06

Eventually the tests to verify that residuals were homoscedastic and normally distributed were performed:





During our work, we noticed the following behaviour. Whenever we were executing the system with the same parameters, but without vacation, the mean response time was that of an M/M/1 queue. By adding small vacations (in the range of 1-30ms), the mean response times could easily be obtained by adding the mean response time of the M/M/1 and the mean vacation time.

With large vacation this was no longer the case, because the utilization of the system also increased.

$$E[R] = \frac{1}{\lambda} \frac{\rho}{1-\rho} + V$$

The response times of the M/M/1 system are exponentially distributed. Also the vacation time is exponentially distributed. Given that the response times of the system with vacation can be expressed as the sum of the two RVs, we believe that these response times are distributed as the sum of two exponentials (with different rates). Thus, after some research, we tried fitting the response times with the hypoexponential distribution.

$$\text{PDF: } f(x) = \frac{\lambda_1 \lambda_2}{\lambda_1 - \lambda_2}\left(e^{-x\lambda_2} - e^{-x\lambda_1}\right)$$

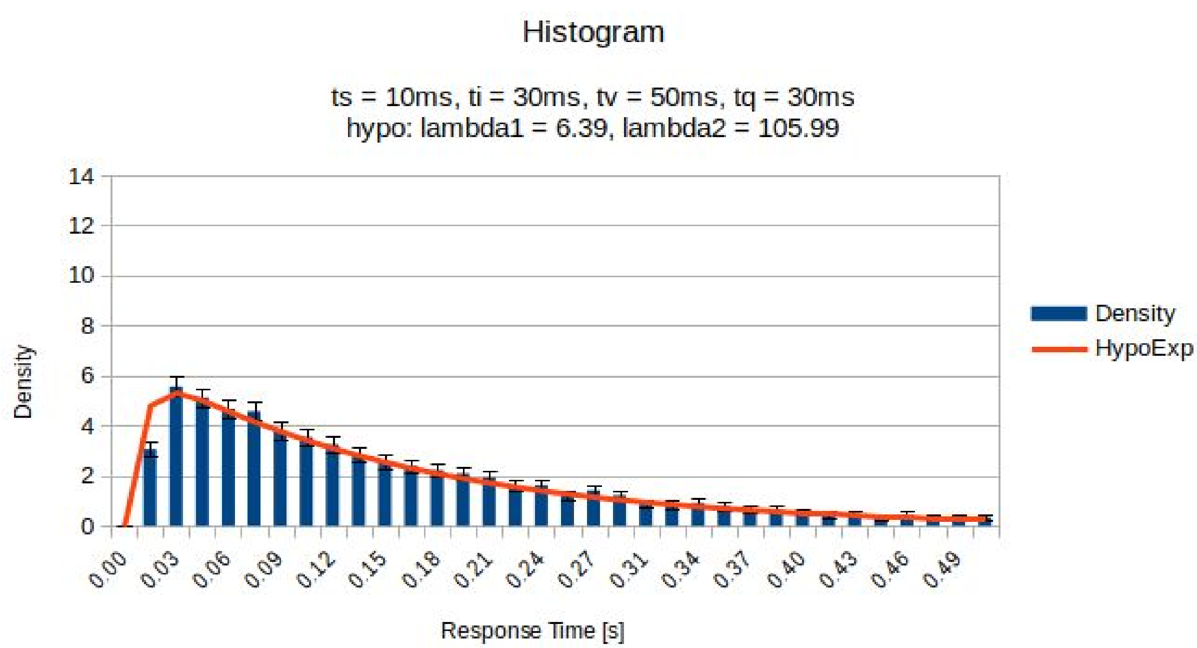
We used the MLE method for estimating the parameters λ_1 and λ_2 of the distribution:

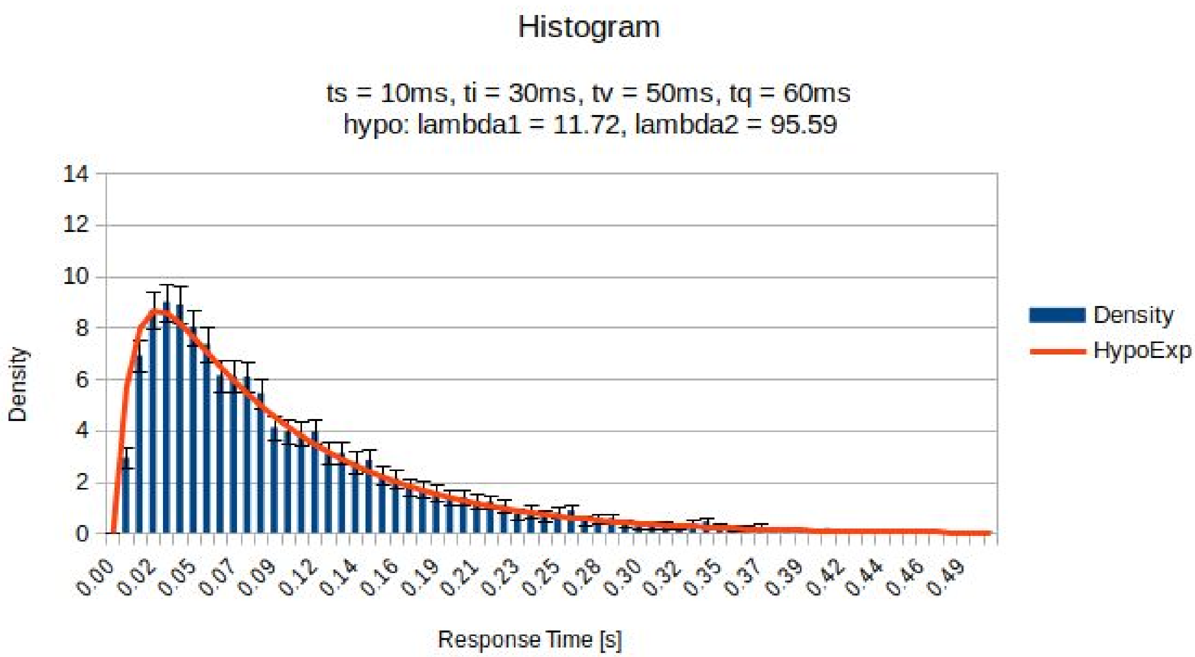
$$\lambda_1 = \frac{2}{\bar{x}}\left[1 + \sqrt{1 + 2(c^2 - 1)}\right]^{-1}$$

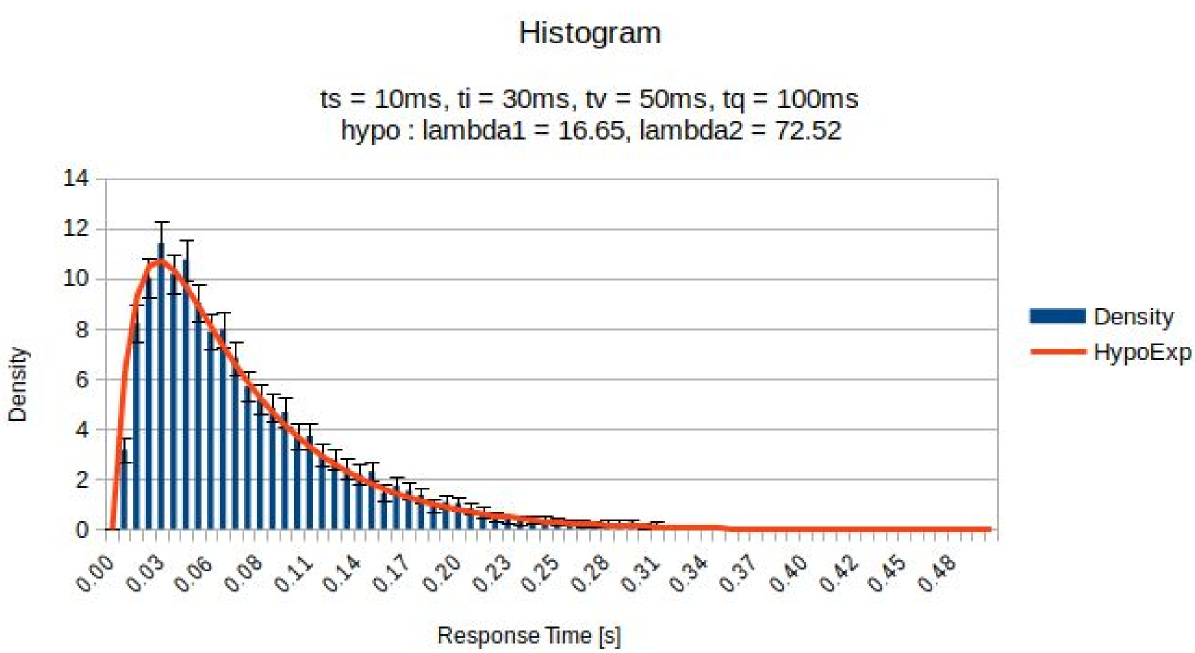
$$\lambda_2 = \frac{2}{\bar{x}}\left[1 - \sqrt{1 + 2(c^2 - 1)}\right]^{-1}$$

where $\bar{x}$ is the sample mean and c is the coefficient of variation.

In the following graphs we have plotted the empirical and theoretical densities of the response times for these scenarios: t_s = 10ms, t_i = 30ms, t_v = 50ms, t_q = {40, 60, 100} ms




Histogram
ts = 10ms, ti = 30ms, tv = 50ms, tq = 60ms
hypo: lambda1 = 11.72, lambda2 = 95.59
Density
Response Time [s]
Density
HypoExp


Histogram
ts = 10ms, ti = 30ms, tv = 50ms, tq = 100ms
hypo : lambda1 = 16.65, lambda2 = 72.52
Density
Response Time [s]
Density
HypoExp

Here is also an example of qq-plot showing the genuinity of our fitting.