2022

Single Channel Queuing Simulation

BY STUDENT: AMR RAAFAT MOHAMMED MOHAMMED ELBARDINI

MODELING AND SIMULATION 2022

Single Channel Queuing Application

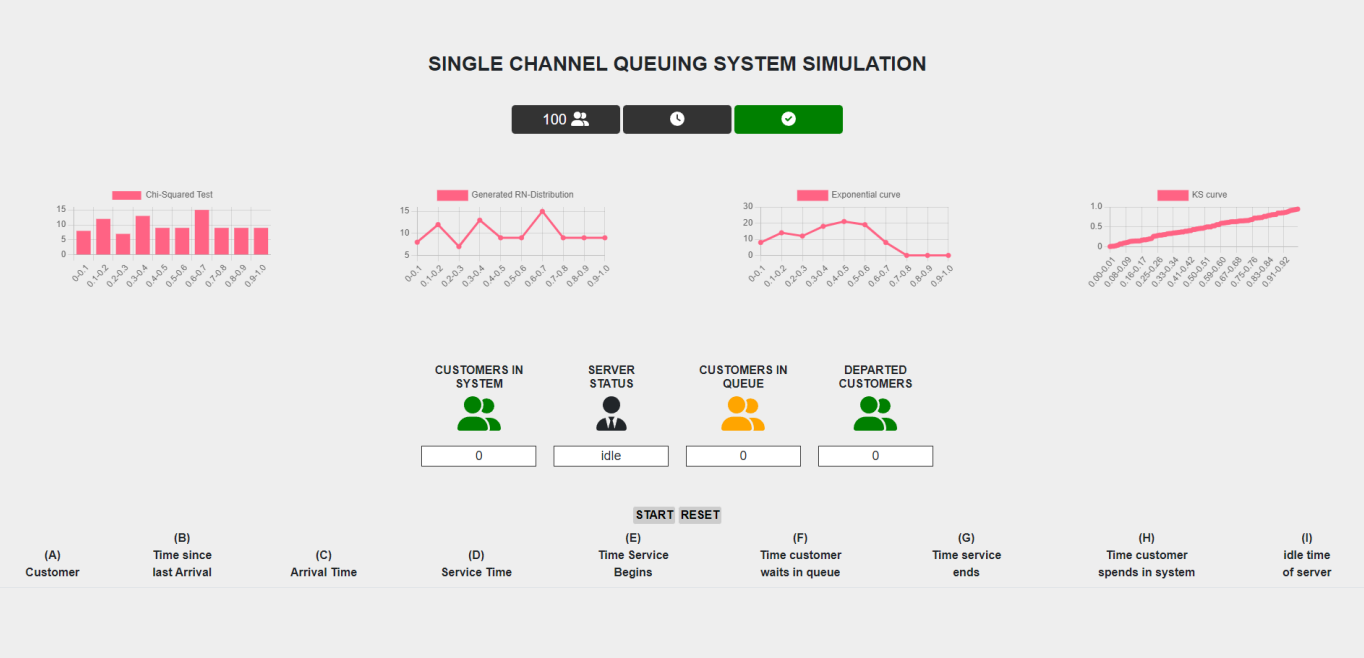

Figure-1 Single channel Application Dash Board

Introduction

This application aims to solve a single channel queuing system problem, using generated random numbers, the numbers have to go through certain tests to check their distribution and dependency, the green check mark in the dashboard shown in figure-1 represents the passing of the generated random numbers patch, the patch is accepted and it proceeds through the application phases.

If the check mark turns red this indicates a failure to pass at least one test, which in turn the application is forced to stop, and the user has to reset to generate a new patch and get a green check mark to proceed.

Application Phases

1. Generating Random numbers

Using the Math.Random() function in JavaScript, we were able to generate 100 random numbers, the Math.Random() function output is uniformly distributed.

2. Testing Random numbers

Chi-Squared Test:

The generated random numbers are divided into classes or segments each represents the weight of each range (the number of times the random numbers fall into a specific range), this forms our observed values, the Expected values are the expected weight of repetition for each range.

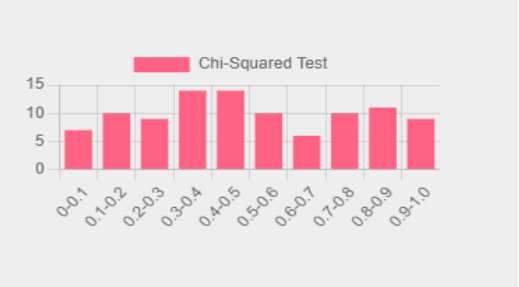

Figure-2 Chi-Square graph

Choosing angle α=0.05 we get critical value=16.919

If the outcome of the test is less than or equal critical value, the random number patch is considered uniform and H0: hypothesis is concluded.

Otherwise the H1: hypothesis is concluded.

Kolmogorov-Smirnov Test:

The generated random numbers are sorted and input through the KS test function D_{max} value is generated and then compared to the critical value from the tables at chosen angle α.

Choosing angle α=0.05 we get critical value=0.136 by substituting

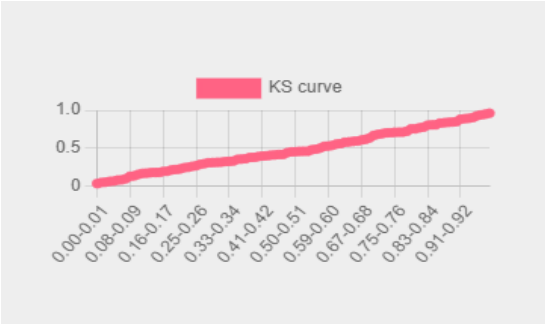

Figure-3 KS-TEST Graph

In the equation for N>40: $\frac{1.36}{\sqrt{N}}$

H0: hypothesis is concluded if KS-Test outcome for random numbers is less than or equal critical value then the random numbers are uniformly distributed.

Otherwise the H1: hypothesis is concluded.

Auto-Correlation Test:

The generated random numbers are input through the auto correlation test function, the purpose of that test is to eliminate the possibility of dependency between the generated random numbers, in order to make sure that they're truly uniform.

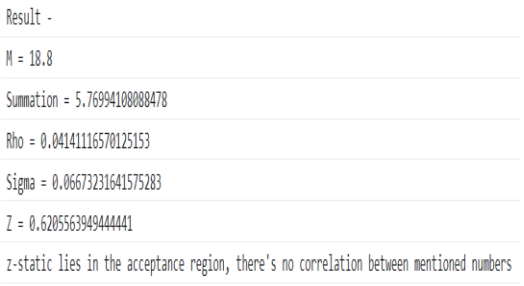

Figure-4 Auto-Correlation TEST Results

Taking angle α=0.05 and m(step)=5 we get the value of $Z(\frac{\alpha}{2})$=0.025

We get critical value range of -1.96 to +1.96

If the z-static lies in the acceptance region then there is no dependency between the random numbers at the chosen m.

3. Exponential Distribution

Since inter-arrival time is exponentially distributed our simulation application is required to simulate this distribution and thus after the random numbers generated passed the previous tests we know that they are uniformly distributed.

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & \text{elsewhere} \end{cases}$$

Figure-5 exponential distribution

Converting the uniformly distributed random numbers into exponential distribution using the equation in figure-5.

Getting the value of λ=1 according to figure-6, since our random numbers are generated from 0 to 1.

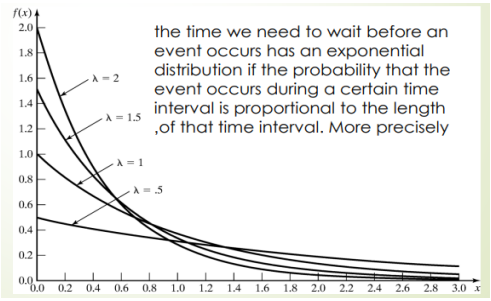

Figure-6 Exponential distribution graph

The output of this function is an exponentially distributed random number array.

Which is then multiplied by a factor and used to generate inter-arrival time.

The application then calculates the average waiting time for the given inputs and the rest of the required simulation values shown in figure-7.

```
totalCustomerWait 28740
average waiting time  287.4
probability wait 0.99
probability of idle server 0
average service time 17.75
average time between arrivals 11.81
average waiting time of those who wait  290.3030303030303
average time customer spends in the system  305.15
```

Figure-7 Simulation output

Conclusion:

Since the value of the Average waiting time is 287.4 minutes, we conclude that our system is not suitable to handle 100 customers.

Long wait will only result in customers being unsatisfied, therefore another server maybe hired to prevent such outcome.