

*MODULATION AND SIMULATION (SE-207)*

****

**Mr. Rahul**

**Shivam Bharti (2K19/SE/115) & Pawan (2K19/SE/083)**

**QUEUEING MODEL FOR CNG PUMP STATION**

MTE PROJECT REPORT

SUBMITTED BY:

**ACKNOWLEDGEMENT**

We pay our deep sense of gratitude to **Mr. Rahul** of Department of Computer Science, **Delhi Technological University**, to provide us this opportunity to prepare this Project. We are immensely obliged to **our friends** for their elevating each other inspiration, encouraging guidance, and kind supervision in completion of this Project.

**CONTENTS**

* Abstract
* Introduction
* Literature Review
* Aim & Objective
* Proposed Methodology
* System Requirements
* Case Study
* Design
* Code Implementation
* Module wise explanation
* Output Screenshots
* Model Validation
* Result & Discussion
* Conclusion
* References

**ABSTRACT**

In our modern society, critical decision making has become a serious problem to top management in different sections of industry including service industry, due to many limitations in time, cost, labour and different sources of energy. Regarding the situation described above, using different tools to optimize processes and reduce costs through monitoring the system’s behaviour, forecasting changes and evaluating various types of critical decisions before their implementation or taking any physical action are becoming more important. Here is the place where simulating a process or behaviour of a system can widely help managers come to the best solutions. This project mainly aims at simulating the behaviour of a CNG PUMP station as one of the most important sections of service industry to which everyone has to deal with every now and then. Simulation would provide numerous opportunities for managers to take the advantages of such a tool to come to the best solutions to promote their businesses with the lowest time and cost.

# INTRODUCTION

A model is presented in this project for CNG pumps. The model was developed under the assumption that there are many customers having requirement of fuel for uninterrupted working of their vehicles and limited number of gas outlets in working conditions at every CNG pump (service providers) which provide gas to fill the cylinders of vehicles. The heavy rush at CNG pumps motivates the people to move for other fuels like petrol, diesel etc. which increases the level of pollution in the atmosphere in India. Studying this situation in many CNG pumps we realized that people are ready to switch from other pollution increasing fuels to CNG but the long queues at CNG pumps demotivates them. As per the survey of Times of India, it was observed that long queues of vehicles waiting outside CNG pump stations led to traffic congestion on service lanes or even main roads. It was also observed that haphazardly parked cars and autos (Three wheelers) can also lead to accidents.

There is regular heavy traffic block outside the CNG filling station in every area on the nearby roads. The whole road is occupied by vehicles waiting for filling fuel. They queue up in 2-3 rows which take away a major portion of the road. Besides, there is other traffic also near the CNG station and it is a nightmare to drive through these roads many times. Heavy rush at CNG stations is a major problem in our country in current scenario as this is the best fuel for having pollution free transportation. The demand of fuel at CNG pumps is increasing day by day but the capacity of CNG pump is not in the same ratio. Delays in the availability of fuels may cause drastic outcomes for customers as well as for surroundings. CNG Pump’s performance in terms of customers flow and of the available resources can be studied using the Queuing Theory.

CNG stations can be regarded as a network of queues and servers where customers with their requirements of fuel in respective vehicle arrive, wait for service, get the fuel tank i.e. gas cylinder filled and then leave the station. The length of queue, utilization of pumps and waiting time evaluations are effective tools to support management decisions about capacity planning of their CNG stations as per the arrival and service rate of customers. The results of this study can help us to understand the broader problem, the relationship between resources and waiting times, and to provide a method for understanding and provides a better solution to face the daily crisis at CNG pumps. These problems could be solved from the results of the proposed model. To use queuing models with simulation modelling at CNG pump stations to provide an accurate evaluations of the system’s performance in present scenario, to make appropriate capacity plan in order to maintain optimal uses of resources.

**LITERATURE REVIEW**

Simulation modeling and analysis is defined as a process aims to create and experiment with a physical model of system which is computerized using programming and mathematical logics. A system is identified as a set of related interacting sections that receive input and supply output for many goals. This field contains traditional simulation and training simulators. Generally, the difference is as follows. Traditional simulation is applicable to analyse systems and make operating decisions. Training simulators are used to train users making better decisions or improving the performance of individual processes. Modeling a complete system with particular attention to detail in the specific component intended for analysis enables concepts relative to the component to be analyzed as well as their influence on the rest of the system. The purpose of modeling and simulating dynamic systems is to generate a set of algebraic and differential equations or a mathematical model. Many different industries can benefit from the advantages of computer simulation modeling; manufacturing, assembly, aerospace, automobile and telecommunication industries are such industries that are broadly using computer simulation modeling to promote their quality and efficiency while lowering their costs. Among all the above-mentioned industries, service sector is not an exception. Banks, Rail Stations, Airports, Healthcare, pilgrim movement and many others are the clear instances where computer simulation modelling has had a great impact on their performance. One of these service-oriented sectors that is going to be focused in this project is CNG PUMP station, where a few effective efforts have been done, from the computer simulation modelling point of view. In many industries, many decisions are being made to which there are no clear outcomes. Large amounts of money, time and energy would be put into action but in many cases, the desired result would not be met. Regarding to the fact that a CNG Pump station is a complex combination of different parameters such as various resources and machinery, facility location, location population, staff and customers, simulating the behaviour of such a system before physically implementing it would play a great role in how successful the business will perform. Nowadays, competitive strategies for serving customers with the highest quality, lowest waste and costs have become a key factor to perform a successful business.

We have created a C++ program for multi-server queuing system. We have also taken advantage of C++’s facility to give variables and functions fairly long names, which thus should be self-explanatory. (For instance, the current value of simulated time is in a variable called “sim\_time”).

# Aim & Objective

For each CNG centre it is a challenge to decrease the waiting time for environment friendly fuel and to improve the customer’s satisfaction. Long waiting times is the most important problem in customer’s satisfaction. Most frequent issues are about the waiting time which is too long and the number of less outlets. To manage these situations, we will use multi-server queuing models which can provide reasonably accurate evaluations of our system’s performance.

# Proposed Methodology

* *All customers have two choices to select from depending upon their vehicle -->* We are going to divide the customers/vehicles in 4 queues (i.e. two outlets for each three-wheelers /auto’s and two outlets for each four-wheelers). *Payment can be done using either method (cash or debit/credit card, depending on the customer choice).*
* *No customer leaves the system after entering the queue: Balking or reneging in system is not allowed (i.e. if queue line is going slowly then customer must wait for service, leaving the system after entering the queue/system/gas station is prohibited).*
* *There is no jockeying in the system:* Basically, jockeying is a queue behaviour where no customer/vehicle moves from one queue to another queue if they think they have chosen a slow line. As queue had separated earlier on the basis of type of vehicle, hence there should be no jockeying in the system.
* *We are also assuming that if customer enters the system, he/she should have to refuel his/her vehicle fuel tank before leaving the system.*
* The observation process (data collection) was completed in different hours of the day.
* **Hardware and Software Requirements:**

• Processor: Intel Pentium Dual-Core or any better one.

• RAM: 2 GB at least

• Hard Disk: 10 GB free will be more than sufficient.

• OS: Windows XP/Vista/ 7/ 8 /8.1/ 10.

• ***Language used***: C++

# Case Study

The case study is about simulating the behaviour of a Gas station named INDERPRASTHA GAS LIMITED CNG Pump located in the Block I, G-Block, govindpuram, Ghaziabad, 201013. This station consists of two main platforms; one of them contains two outlet – generally for three wheelers and another one contains two CNG generally for four-wheelers. Four cashiers are located in this station two for each platform.

As the first step of simulating the behaviour of a CNG pump station, different model’s elements and their interactions should be mentioned; how customers enter the model, move through different stages of the model and how they exit the model.

Process map is made for this reason to illustrate the exact relations and sequences of different sections and activities while customers enter and exit the model.

The second step to construct the simulation model would be collecting data for each activity’s duration that occurs in the model. Therefore, based on the flow charts, related data should be recorded and gathered.



**DESIGN**

* **Process Map**

**Is vehicle three -wheeler?**

**Pay the price by either cash or card**

**Refuel the vehicle**

**YES**

**NOYES**

**Pay the price by either cash or card**

**Refuel the vehicle**

* **Flow chart for Arrival routine**

**Schedule the next Arrival event for customer**

**Is both the outlet are idle**

**NOYES**

**NOYES**

**YES**

**Move to the idle server**

**YES**

**NOYES**

**YES**

**Is queue length of both the Outlet is same?**

**Is both the Outlet No. 1 /3 and 2 /4 Busy?**

**Choose either Outlet No. 1 /3 or 2 /4**

**Is queue length of Outlet 1/3 is less than of 2/4?**

**Store time of arrival of this customer**

**Add 1 to the no. of in queue of Outlet No. 2 /4**

**NOYES**

**Move to the Chosen Outlet**

**Set delay=0 for this customer & gather Statics**

**YES**

**Store time of arrival of this customer**

**NOYES**

**Add 1 to the no. of in queue of Outlet No. 1 /3**

**Choose Outlet No.2 /4?**

**Add 1 to the No. of Customer delayed for this Outlet**

**Move to the Chosen Outlet**

**YES**

* **Flowchart for departure routine**

**Make this Outlet idle**

**Move each customer in queue (if any for this outlet) Up one place**

**Schedule the Departure event for this customer**

**Make this Outlet busy**

**Add 1 to the No. of customer delayed for this Outlet**

**Subtract 1 to the no. in queue of this Outlet**

**Compute delay of Next customer entering the service and gather statics**

**Eliminate the departure event from consideration from this outlet till the next event is arrival**

**YES**

**NOYES**

**Is Queue Empty of this Outlet?**

**Code Implementation:**

#include <iostream>

#include <time.h>

#include <conio.h>

using namespace std;

#define Q\_LIMIT 100

#define BUSY 1

#define IDLE 0

float D1, D2;

int Count = 0, ch = 1;

class Server

{

int next\_event\_type, num\_cust\_delayed, num\_delays\_required, num\_events, num\_in\_q\_1, num\_in\_q\_2, server\_status\_1, server\_status\_2;

float area\_num\_in\_q\_1, area\_num\_in\_q\_2, area\_server\_status\_1, area\_server\_status\_2, mean\_interarrival, mean\_service, sim\_time;

float time\_arrival[Q\_LIMIT + 1], time\_last\_event, time\_next\_event[3], total\_delay, delay\_S1, delay\_S2;

public:

void main\_program();

void initialize();

void timing();

void arrival();

void departure();

void report();

void event();

float library(float mean);

float expon(float mean\_1);

int choose();

float fuel\_consumption(int N);

};

void Server::main\_program()

{

num\_events = 2;

cout << "\n\t\t\*\*\* Multi-server queueing system \*\*\*\n";

cout << " Please provide the given details: " << endl;

getch();

cout << "\n Mean inter-arrival time= ";

cin >> mean\_interarrival;

cout << "\n Mean service time= ";

cin >> mean\_service;

cout << "\n Number of customers= ";

cin >> num\_delays\_required;

initialize();

while (num\_cust\_delayed < num\_delays\_required)

{

timing();

event();

switch (next\_event\_type)

{

case 1:arrival();

break;

case 2:departure();

break;

}

}

report();

cout << "\n Press any key to continue!! ";

getch();

}

void Server::initialize()

{

sim\_time = 0.0;

server\_status\_1 = IDLE;

server\_status\_2 = IDLE;

num\_in\_q\_1 = 0;

num\_in\_q\_2 = 0;

time\_last\_event = 0.0;

num\_cust\_delayed = 0;

total\_delay = 0.0;

delay\_S1 = 0.0;

delay\_S2 = 0.0;

area\_num\_in\_q\_1 = 0.0;

area\_num\_in\_q\_2 = 0.0;

area\_server\_status\_1 = 0.0;

area\_server\_status\_2 = 0.0;

time\_next\_event[1] = sim\_time + library(mean\_interarrival);

time\_next\_event[2] = 1.0e+40;

}

void Server::timing()

{

int i;

float min\_time\_next\_event = 1.0e+39;

next\_event\_type = 0;

for (i = 1; i <= num\_events; ++i)

{

if (time\_next\_event[i] < min\_time\_next\_event)

{

min\_time\_next\_event = time\_next\_event[i];

next\_event\_type = i;

}

}

if (next\_event\_type == 0)

{

cout << "\n Event list empty at time " << sim\_time;

exit(1);

}

sim\_time = min\_time\_next\_event;

}

void Server::event()

{

float time\_lag;

time\_lag = sim\_time - time\_last\_event;

time\_last\_event = sim\_time;

area\_num\_in\_q\_1 += num\_in\_q\_1 \* time\_lag;

area\_num\_in\_q\_2 += num\_in\_q\_2 \* time\_lag;

area\_server\_status\_1 += server\_status\_1 \* time\_lag;

area\_server\_status\_2 += server\_status\_2 \* time\_lag;

}

void Server::arrival()

{

time\_next\_event[1] = sim\_time + library(mean\_interarrival);

if (server\_status\_1 == BUSY && server\_status\_2 == BUSY)

{

switch (ch)

{

case 1: ++num\_in\_q\_1;

if (num\_in\_q\_1 > Q\_LIMIT)

{

cout << "\n Overflow!! of the array time-arrival at " << sim\_time;

exit(2);

}

time\_arrival[num\_in\_q\_1] = sim\_time;

ch = 2;

break;

case 2: ++num\_in\_q\_2;

if (num\_in\_q\_2 > Q\_LIMIT)

{

cout << "\n Overflow!! of the array time-arrival at " << sim\_time;

exit(2);

}

time\_arrival[num\_in\_q\_2] = sim\_time;

ch = 1;

break;

default: break;

}

}

else if (server\_status\_1 == IDLE && server\_status\_2 == BUSY)

{

++num\_cust\_delayed;

server\_status\_1 = BUSY;

time\_next\_event[2] = sim\_time + library(mean\_service);

D1 = time\_next\_event[2];

}

else if (server\_status\_2 == IDLE && server\_status\_1 == BUSY)

{

++num\_cust\_delayed;

server\_status\_2 = BUSY;

time\_next\_event[2] = sim\_time + library(mean\_service);

D2 = time\_next\_event[2];

}

else

{

++num\_cust\_delayed;

server\_status\_1 = BUSY;

time\_next\_event[2] = sim\_time + library(mean\_service);

D1 = time\_next\_event[2];

}

}

void Server::departure()

{

int i;

float delay;

if (num\_in\_q\_1 == 0 && num\_in\_q\_2 == 0)

{

server\_status\_1 = IDLE;

server\_status\_2 = IDLE;

time\_next\_event[2] = 1.0e+40;

}

if (num\_in\_q\_1 > 0)

{

--num\_in\_q\_1;

if (D1 == time\_next\_event[2])

{

delay = sim\_time - time\_arrival[1];

delay\_S1 += delay;

++num\_cust\_delayed;

time\_next\_event[2] = sim\_time + library(mean\_service);

for (i = 1; i <= num\_in\_q\_1; ++i)

time\_arrival[i] = time\_arrival[i + 1];

}

}

else if (num\_in\_q\_2 > 0)

{

--num\_in\_q\_2;

if (D2 == time\_next\_event[2])

{

delay = sim\_time - time\_arrival[1];

delay\_S2 += delay;

++num\_cust\_delayed;

time\_next\_event[2] = sim\_time + library(mean\_service);

for (i = 1; i <= num\_in\_q\_1; ++i)

time\_arrival[i] = time\_arrival[i + 1];

}

}

}

void Server::report()

{

cout << "\n Simulation time= " << sim\_time;

total\_delay = area\_num\_in\_q\_1 + area\_num\_in\_q\_2;

cout << "\n total delay= " << total\_delay;

cout << "\n Average delay in queue= " << total\_delay / num\_cust\_delayed;

cout << "\n Waiting time of customers in Queue 1= " << area\_num\_in\_q\_1;

cout << "\n Waiting time of customers in Queue 2= " << area\_num\_in\_q\_2;

cout << "\n Server 1 utilization= " << area\_server\_status\_1 / sim\_time;

cout << "\n Server 2 utilization= " << area\_server\_status\_2 / sim\_time;

float Fuel;

Fuel = fuel\_consumption(num\_delays\_required);

cout << "\n Total fuel consumption= " << Fuel;

}

float Server::library(float mean)

{

float M;

M = expon(mean);

Count++;

return (M);

}

float Server::expon(float mean\_1)

{

srand((unsigned)time(NULL));

float m, Array[100];

for (int i = 0; i <= 100; i++)

{

m = ((float)rand() / RAND\_MAX);

m \*= mean\_1;

Array[i] = m;

}

return Array[Count];

}

float Server::fuel\_consumption(int N)

{

srand((unsigned)time(NULL));

int Array[N], m;

int sum = 0;

cout << endl;

for (int i = 0; i < N; i++)

{

m = rand() / 1000;

if (m == 0)

Array[i] = m + 10;

else

Array[i] = m;

sum += Array[i];

}

return sum;

}

int main()

{

Server s;

s.main\_program();

return 0;

}

# Module-wise Explanation

We have run our C++ program on several different computers and compilers. The numerical results differed in some cases due to inaccuracies in floating point operations. This can matter if, e.g., at some point in the simulation two events are scheduled very close together in time, and round-off error results in a different sequencing of the event’s occurrences.

* **#include<time.h>:** This header file is included to declare the functions for the random-number generator.
* **#define BUSY 1** 🡪 server is busy

**#define IDLE 0** 🡪server is idle

* **next\_event\_type, num\_cust\_delayed, num\_delays\_required, num\_events, num\_in\_q\_1, num\_in\_q\_2, server\_status\_1,server\_status\_2** 🡪 All are state variables and all are self-explanatory ( *eg, num\_in\_q\_1 used to store the number of customers in queue 1).*
* **area\_num\_in\_q\_1, area\_num\_in\_q\_2, area\_server\_status\_1, area\_server\_status\_2, mean\_interarrival, mean\_service, sim\_time, time\_arrival [Q\_LIMIT+1], time\_last\_event, time\_next\_event[3], total\_delay,delay\_S1,delay\_S2🡪**Some of them are statistical counters( area\_num\_in\_q\_1{area under Q(t)1 }, area\_server\_status\_1 {area under B(t)1 } ).
* **void main\_program()🡪** This function control the whole program. Firstly call the initialize function and then take the user input as mean arrival time, mean service time and number of customers, then timing and event functions will repeatedly call until a stopping condition reached (i.e. number of customers delayed extend the value of number of customers).
* **void initialize()->** This function initialize all the state variables and statistical counters with 0 and initialize the arrival for customer1 with the help of library function which create the random numbers in between 0-1, departure as 1.0e+40(because we have to execute arrival first).
* **void timing()->** This function checks the type of the next event type either arrival or departure and according to this advance the simulation clock to this event time.
* **void event()->** This function update the statistical counters as:

-time lag = simulation clock time-last event time.

-area under Q(t)1 = Previous value of area under -Q(t)1 + (number in queue1)\*(time lag).

-area under Q(t)2 = Previous value of area under -Q(t)2 + (number in queue2)\*(time lag).

-area under B(t)1 = Previous value of area under -B(t)1 + (server status1)\*(time lag).

-area under B(t)1 = Previous value of area under -B(t)1 + (server status2)\*(time lag).

* **float library(float mean)->** This function of float return type which create random numbers with the help of rand() function within 0 & 1, and after multiplying it with given mean return the random number.
* **float expon(float mean\_1)->** This function helps library function to get rid of same random number problem.
* **void arrival()->** This function first schedule the arrival for next customer, then it checks:-

if both the servers are busy if so then this up-to customer that it may stand in the Queue1 or Queue2. If he choose 1 then increase 1 in number of queue1 and store this time in time of arrival array otherwise increase 1 in number of queue2 and store this time in time of arrival array.

Else if server1 is idle and server2 is busy then increase number of customers delayed by 1, set server1 to busy and schedule the departure time for this customer with the help of library function.

Else if server2 is idle and server1 is busy then increase number of customers delayed by 1, set server2 to busy and schedule the departure time for this customer with the help of library function.

Else if both the servers are idle, let give priority to server1 then increase number of customers delayed by 1, set server1 to busy and schedule the departure time for this customer with the help of library function.

* **void departure()->** This function firstly checks if both the queues are empty then set both the servers as idle.

Else if the number of queue1 containing at least 1 customer then decrease the number of queue1 by 1, calculate delay for server1 by taking difference of simulation clock time with time of arrival for this customer, increase the number of customer delayed by 1 and schedule the departure for next customer with the help of library function, and pop out the time of arrival for this customer from the array.

Else if the number of queue2 containing at least 1 customer then decrease the number of queue2 by 1, calculate delay for server2 by taking difference of simulation clock time with time of arrival for this customer, increase the number of customer delayed by 1 and schedule the departure for next customer with the help of library function, and pop out the time of arrival for this customer from the array.

* **float fuel\_consumption(int N)->** This function picks up the random fuel consumption with the help of random number generator and return the sum of the fuel.
* **void report()->** This function generate a report of all the given parameters:-

*-simulation time when the simulation ends.*

*-total delay.*

*-Average delay in queue.*

*-Waiting time for customers in queue1.*

*-Waiting time for customers in queue2.*

*-Server1 utilization*

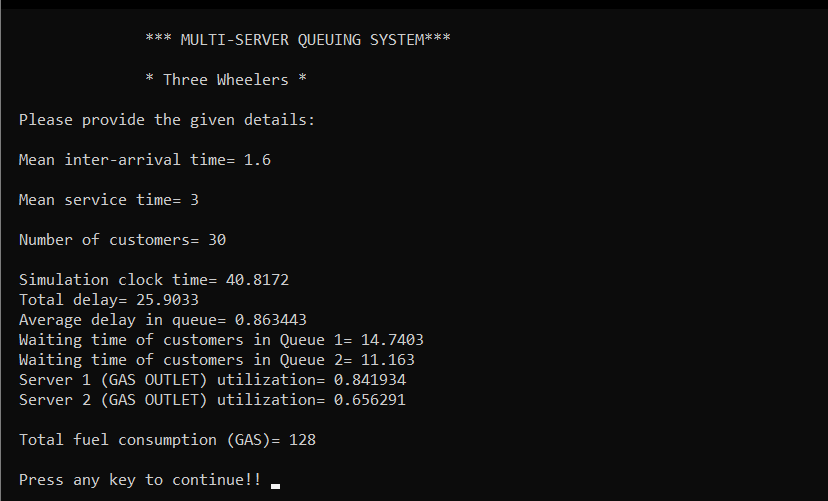
*-Server2 utilization*

*-Total fuel consumption*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**Output Screenshots**

**As the arrivals and departures of customers are random.**

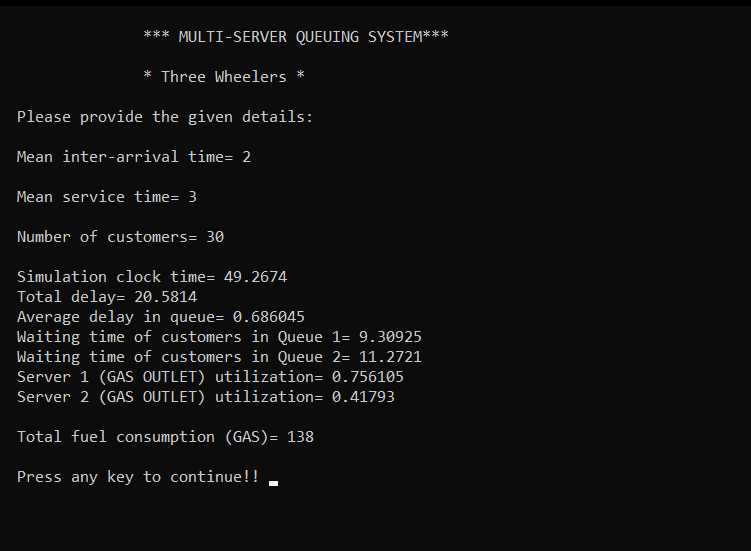
****We have run the above program three times for four wheelers and three wheelers by providing different parameters.**

***By our analysis, we provided mean inter-arrival time as 1.6 minutes and mean service-time as 3. Then we simulate our program for 30 customers for three wheelers.***

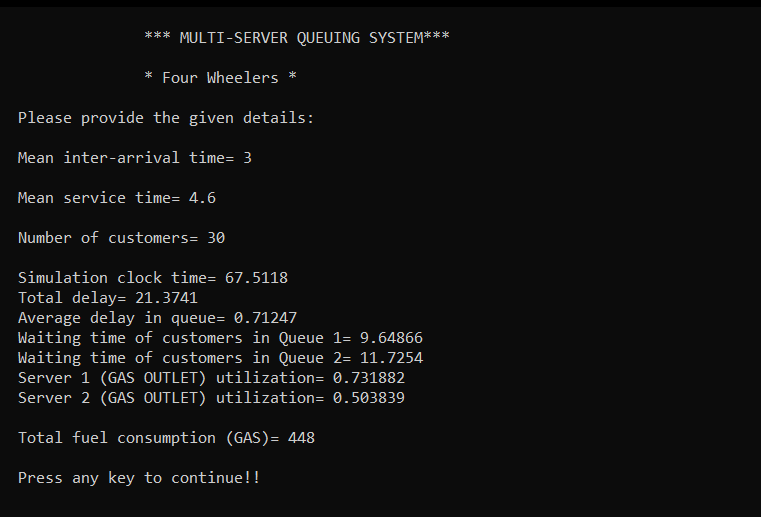
***And we get the above information…***

***Here, we provided mean inter-arrival time as 1.6 minutes and mean service-time as 2.4 for 30 customers.***

***And we get the above information…***

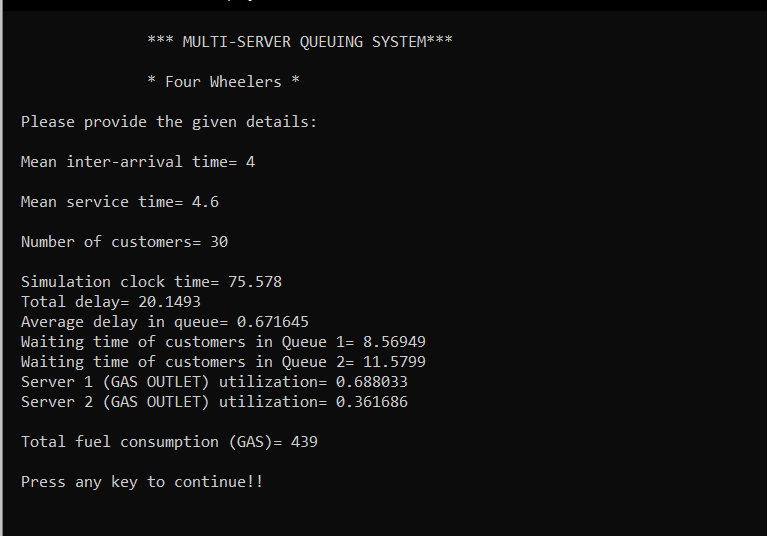
******

***Here, we provided mean inter-arrival time as 2 minutes and mean service-time as 3 for 30 customers.***

***And we get the above information…***

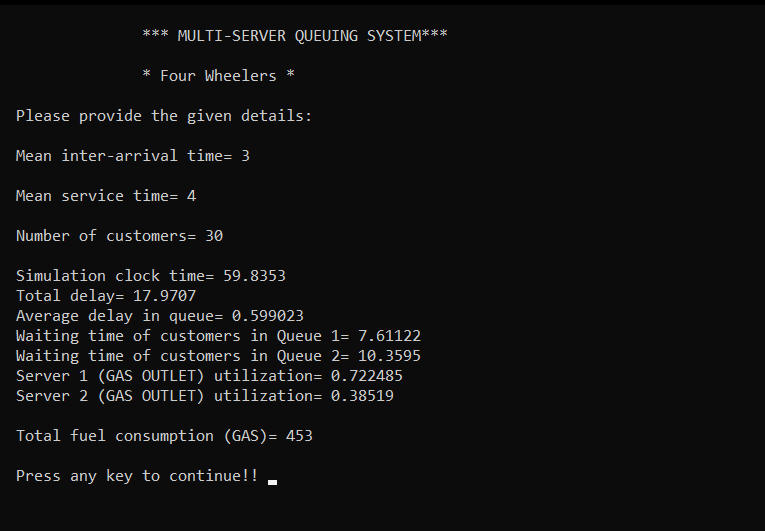
***By our analysis, we provided mean inter-arrival time as 3 minutes and mean service-time as 4.6. Then we simulate our program for 30 customers for four wheelers.***

***And we get the above information…***

******

***Here, we provided mean inter-arrival time as 4 minutes and mean service-time as 4.6 for 30 customers.***

***And we get the above information…***

******

***Here, we provided mean inter-arrival time as 3.0 minutes and mean service-time as 4.0 for 30 customers.***

***And we get the above information…***

Model Statistics after 3 runs

-**Four wheelers table of 30 customers for 1-hour simulation time.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No. of runs** | **Mean inter-arrival time in minutes** | **Mean service time in minutes** | **Total delay in minutes** | **Server1 utilization in percentage** | **Server2 utilization in percentage** |
| 1st run | 3.0 | 4.6 | 18.99 | 75.18 | 50.38 |
| 2nd run | 4.0 | 4.6 | 15.99 | 68.80 | 36.16 |
| 3rd run | 3.0 | 4.0 | 18.02 | 72.24 | 38.51 |

-**Three wheelers table of 30 customers for 1-hour simulation time.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No. of runs** | **Mean inter-arrival time in minutes** | **Mean service time in minutes** | **Total delay in minutes** | **Server1 utilization in percentage** | **Server2 utilization in percentage** |
| 1st run | 1.6 | 3.0 | 38.07 | 84.19 | 65.62 |
| 2nd run | 1.6 | 2.4 | 21.92 | 78.40 | 45.37 |
| 3rd run | 2.0 | 3.0 | 25.06 | 75.61 | 41.79 |

# Model Validation

Validation is a process aims to ensure that the simulation model perform as reality. The simulation model validation process contains of both face validation and statistical validation. Face validation is the constant process of guaranteeing that the model performs as reality. Statistical validation contains comparing the simulation model to the actual system. Using statistical validity, some performance measure outputs are collected. The similar system processing situations observed during the data collection phase must be reconstructed in the simulation model.

In order to validate the model, the total fuel consumption generated from the model was compared with the actual data obtained from the real situation five times.

|  |  |  |  |
| --- | --- | --- | --- |
| No. of Runs | Fuel consumption/hour (petrol & diesel) in litres | Total Fuel Consumption/  hour  Real Situation | Percentage of variation |
| 1st run | 448 | 457 | 3.282 |
| 2nd run | 452 | 457 | 1.094 |
| 3rdrun | 460 | 457 | 1.969 |
| 4th run | 439 | 457 | 1.750 |

**Model validation results for four wheelers**

|  |  |  |  |
| --- | --- | --- | --- |
| 5th run | 446 | 457 | 4.595 |

**Mean=** 449

Average variation= 2.538 < 5 %

As can be seen, the average variation percentage is less than 5%, which means the model is valid for four wheelers.

**Model validation results for three wheelers**

|  |  |  |  |
| --- | --- | --- | --- |
| **No. of runs** | **Fuel consumption / hour (Litres)** | **Total fuel consumption/hour Real situation** | **Percentage of variation** |
| 1st run | 155.38 | 160 | 2.880 |
| 2ndrun | 167.56 | 160 | 4.725 |
| 3rd run | 138.00 | 160 | 1.668 |
| 4th run | 168.79 | 160 | 4.118 |
| 5th run | 164.78 | 160 | 2.987 |

**Mean=** 159.69

Average variation= 3.275 < 5%

As can be seen, the average variation percentage is less than 5%, which means the model is valid for four wheelers.

# Result and discussion

In our project, two different types of reports are categorized in order to give a better view about what can be derived from these kinds of reports and how useful and helpful they can be for managers to make critical decisions.

Firstly, inter arrival times for the main system entries are discussed. As mentioned before, as customers enter the Gas station, they would have four choices to choose from; they are three-wheelers or four wheelers, they can refuel their cars with CNG. We generated two reports one for three-wheeler and another one for four wheelers, in which we run our program 3 times by varying mean inter-arrival time as well as mean service time and we get the adjacent values of total delay, server1 utilization and server2 utilization, which shows busy time for both fuel pumps. Here is where managers can benefit from the results of a simulation model; optimize their staffing and their working hours, redesign their tasks and jobs, facilitate their CNG station with more advanced devices and many other instances that all would result in better service quality with lower costs and waste.

Secondly, percentage variations of fuel consumption are discussed, in which we run our program 5 times for three wheelers as well as four-wheeler to get better results. A number of variables have been defined in order to give better visualization scenery on what is happening inside the model while it is running. Some of them are being used to verify and validate the model when they are compared with the real-world situation.

# Conclusion

There is regular heavy traffic block outside the CNG filling station in every area on the nearby roads. The whole road is occupied by vehicles waiting for filling fuel. They queue up in 2-3 rows which take away a major portion of the road. Delays in the availability of fuels may cause drastic outcomes for customers as well as for surroundings. Now after doing the analysis of system, we observed that there is long queue formation at all CNG filling points i.e. servers (4 servers working in parallel) due to the high demand of CNG by CNG vehicles like autos, cars, buses etc. Since, we want to provide quick service to customer, so we need to eliminate the waiting line of customers because it gives rise to traffic problems on the roads near CNG pump station and financial as well as goodwill loss, as after observing long waiting queue or long waiting time some customer moves to any other option, as dissatisfied customers. Hence, we should make this system balanced as it is necessary to turn dissatisfied customers into satisfied customers for maintaining goodwill in the market and gaining more revenue. In this regard we can increase number of servers under the financial constraints, so that no server will be idle for long duration and queue size will also be decreased with improved waiting time.

The results of this study can help us to understand the broader problem, the relationship between resources and waiting times, and to provide a method for understanding and providing a better solution to face the daily crisis at CNG pumps.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# References

* <http://www.macs.hw.ac.uk/~jphillips/DAS/DataAnalysisTopic8.pdf>
* <https://www.sciencedirect.com/>
* <https://www.tutorialspoint.com/modelling_and_simulation/modelling_and_simulation_discrete_system_simulation.htm>
* **Amritpal Kaur, Dr. Williamjeet Singh. Analysis of Queuing to Customers Management in Banking System using Simulation.** **Punjabi University Patiala, Punjab (India).**
* **Averill M Law M & S book**.