PRACTICAL FILE

|  |  |
| --- | --- |
| Student Name | Pardeep Kumar |
| UID | 19BCS1896 |
| Section & Group | ON\_19BCS-23 Group-B |
| Department | Computer Science & Engineering |
| Session | Aug-Dec 2022 |
| Course Name | Modelling and Simulation Lab |
| Course Code | CSP-443 |
| Semester | 7TH |

INDEX

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| S. N | Experiment | Date of Experiment | Conduct  (12 marks) | Viva  (10 marks) | Record  (08 marks) | Total  (30 marks) | Teacher’s  Signature |
| 1. | Our aim in this practical is to generate random numbers using random number generator. | 10/08/2022 |  |  |  |  |  |
| 2. | Our aim in this practical is to implement the standard normal distribution. | 18/08/2022 |  |  |  |  |  |
| 3. | Our aim in this practical is to write a program to implement testing of Random numbers (Chi Square) | 24/08/2022 |  |  |  |  |  |
| 4. | To implement Monte Carlo Simulation. | 29/08/2022 |  |  |  |  |  |
| 5. | To implement the simulation of the LAG model. | 05/10/2022 |  |  |  |  |  |
| 6. | To implement simulation of Single Queue Server System. | 12/10/2022 |  |  |  |  |  |
| 7. | Write a program to implement simulation of Multiple Queue Server System. | 06/10/2022 |  |  |  |  |  |
| 8. | Write a program to implement simulation of telephonic system. | 06/10/2022 |  |  |  |  |  |
| 9. | Write a program to implement simulation of telephonic system. | 06/10/2022 |  |  |  |  |  |
| 10. | Write a program to implement COBWEB model. | 06/10/2022 |  |  |  |  |  |

Experiment - 1

**Aim:**

Our aim in this practical is to generate random numbers using random number generator.

**Tools/Software Required:**

MATLAB/Octave Online

**Theory:**

**Octave Online**: It is a web UI for GNU Octave, the open-source alternative to MATLAB.

**Types of random functions:**

1. **rand:** It will generate random numbers between 0 & 1.
2. **rand(argument):** It will create a random matrix of order of argument passed inside it between 0 & 1.
3. **rand(x,y):** It will create a random matrix with x rows and y columns between 0 & 1.
4. **randi**: It stands for random integer.
5. **randi(x):** It generators a random integer between 0 to x.
6. **randi(range,x,y):** It generators random integers within the range of 0 to the number specified in the range in a format of array(matrix) of size x,y(x-rows,y-columns).
7. **randperm(x):** It generators a random number between 1 to x.
8. **randn(x):** It will generate a matrix of order x, containing random numbers between 0 to 1. The mean of all the values in the matrix will be 0 and the standard deviation will be 1.

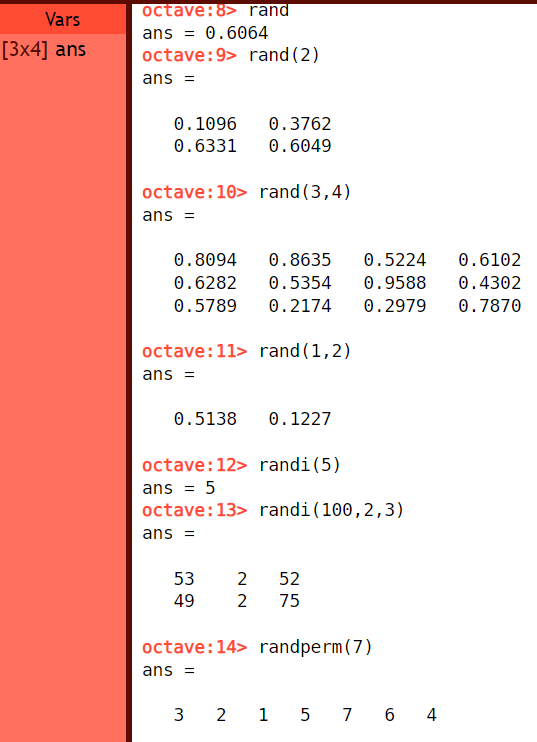
**Formula:**

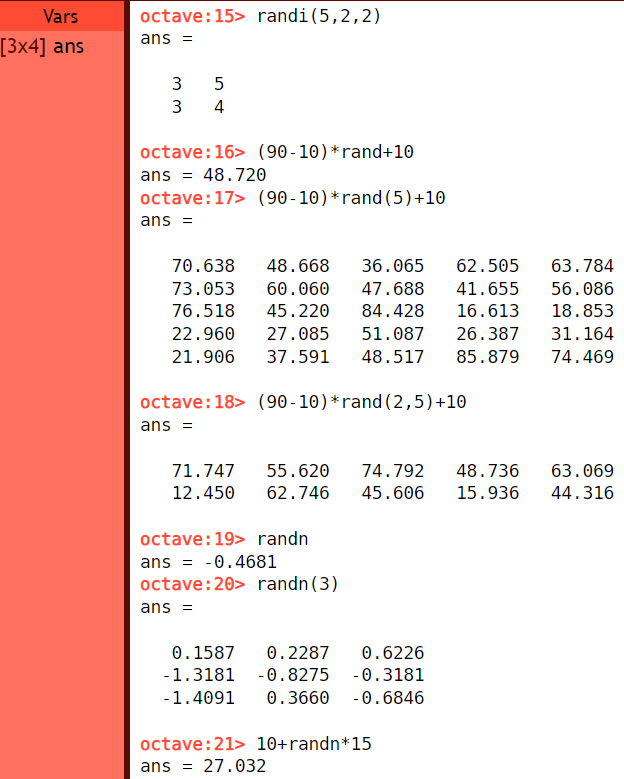
1. (b-a)\*rand+a : Range is from a to b. The formula is used for generating numbers between a specified range.
2. (b-a)\*rand(x)+a: Creates a matrix containing random numbers between the range of (a,b) of order x.
3. (b-a)\*rand(x,y)+a: Creates a matrix containing random numbers between the range of (a,b) with x rows and y columns.
4. X+randn(m,n)\*S: In the formula X is for mean & S is for standard deviation . It will a matrix containing random numbers whose mean will be X and standard deviation will be S with m rows & n columns.

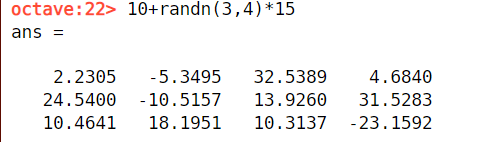
**Algorithm/Procedure:**

**Step:** Execute the above commands with proper syntax. For syntax related help use help command.

**Program Code with Output:**

****





**Result:** The program code is successfully executed, and output is verified.

**Learning Outcomes:**

1. Learnt about random numbers.
2. How to generate random numbers using different functions.

Experiment - 2

**Aim:**

Our aim in this practical is to implement the standard normal distribution.

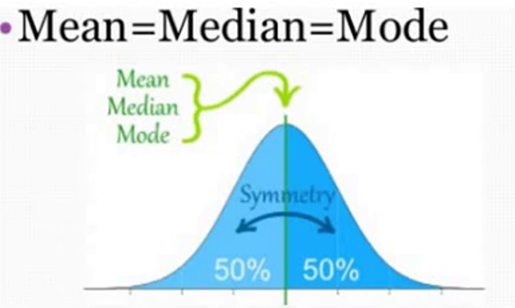
**Tools/Software Required:**

MATLAB/Octave Online

**Theory:**

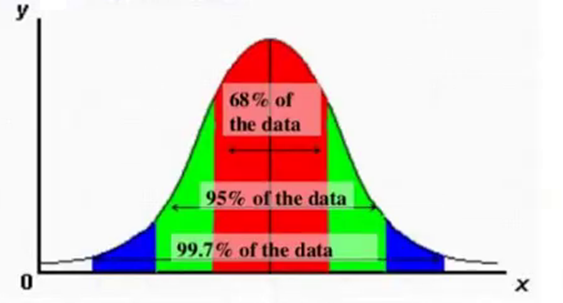
**Octave Online**: It is a web UI for GNU Octave, the open-source alternative to MATLAB.

**Standard Normal Distribution:**

****

* It is defined as a continuous frequency distribution of infinite range. It is a descriptive model that describes real world situations.
* Many dependent variables are commonly assumed to be normally distributed in the population.
* If a variable is approximately normally distributed, we can make inferences about values of that variable.

**Empirical Rule:**

****

* The Empirical Rule states that 99.7% of data observed following a normal distribution lies within 3 standard deviations of the mean.
* Under this rule, 68% of the data falls within one standard deviation, 95% percent within two standard deviations, and 99.7% within three standard deviations from the mean.
* Three-sigma limits that follow the empirical rule are used to set the upper and lower control limits in statistical quality control charts and in risk analysis such as VaR.

**Example used for this experiment:** DGP University conducts placement examination to all incoming freshmen. The examination scores of the 1000 examinees last semester were approximately normally distributed with mean score of 80 and standard deviation of 5. What is the probability that randomly chosen student got a score below 70? Above 82? Between 75 and 90?

**Formula used in this experiment:**

* X = mean + sqrt(var)\*randn(r,c)
* Var 🡪 Variance
* Randn 🡪 The function is used to generate a matrix of order x, containing random numbers between 0 to 1. The mean of all the values in the matrix will be 0 and the standard deviation will be 1.

**Builtin Function used in this experiment:**

* **Linspace:** It gives direct control over the number of points and always includes the endpoints. “lin” in the name “linspace” refers to generating linearly spaced values.
* y = linspace(x1,x2) returns a row vector of 100 evenly spaced points between x1 and x2.
* y = linspace(x1,x2,n) generates n points. The spacing between the points is (x2-x1)/(n-1).
* M = max(A) returns the maximum elements of an array.
* M = min(A) returns the minimum elements of an array.

**Algorithm/Procedure:**

**Step 1:** Load the sample data. Create a vector containing the first column of the students' exam grades data.

**Step 2:** Test the null hypothesis that the data comes from a normal distribution with a mean of 75 and a standard deviation of 10.

**Step 3:** Use these parameters to center and scale each element of the data vector since, by default, kstest tests for a standard normal distribution.

x = (test1-75)/10;

h = kstest(x)

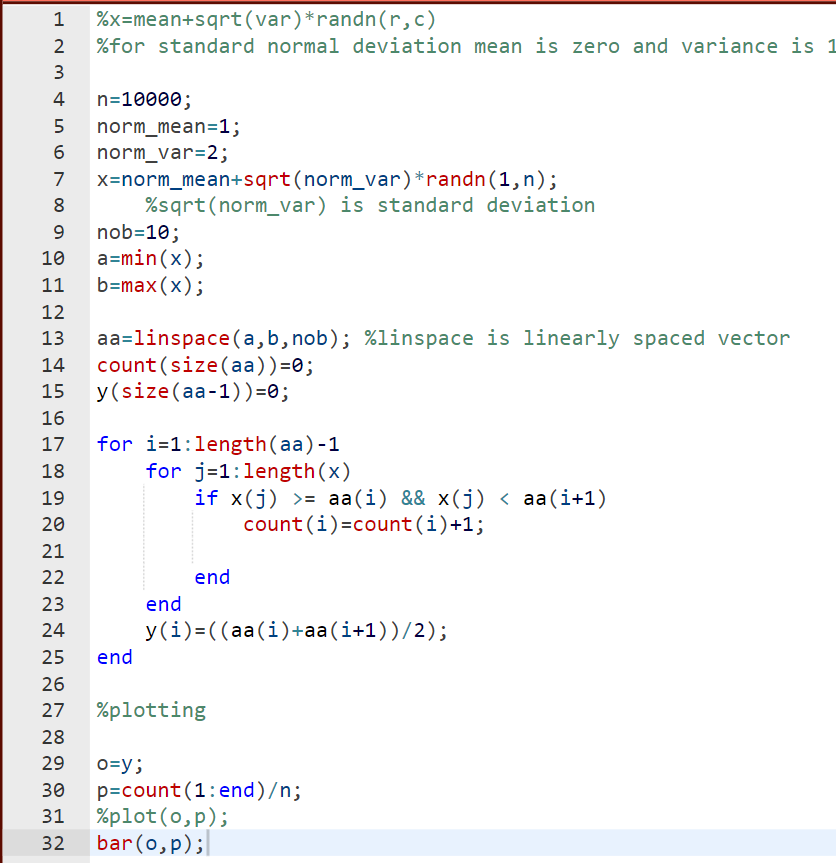
h=0

**Step 4:** The returned value of h = 0 indicates that kstest fails to reject the null hypothesis at the default 5% significance level.

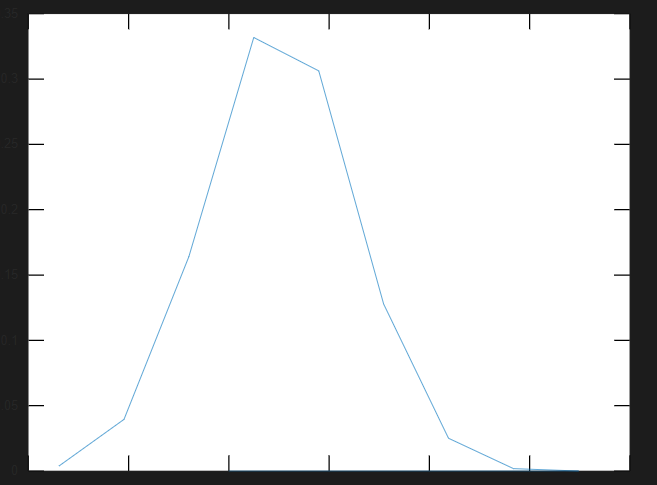
**Step 5:** Plot the empirical cumulative distribution function (cdf) and the standard normal cdf for a visual comparison.

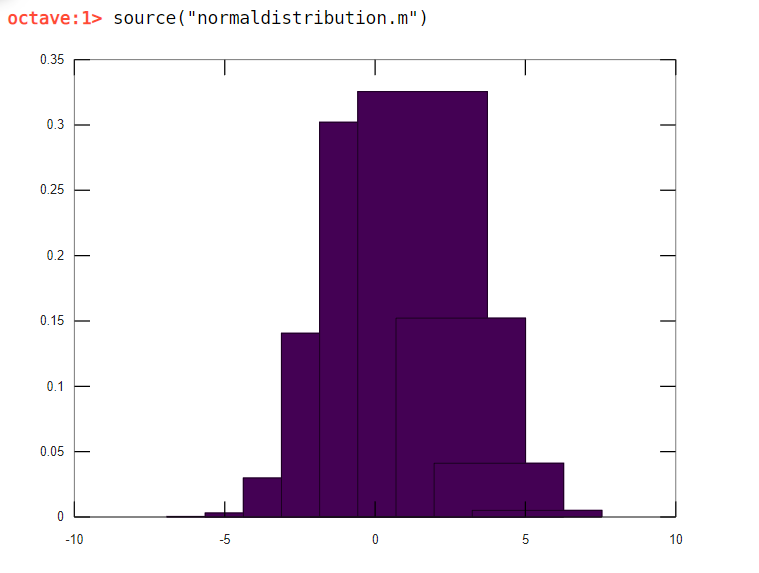
**Implementation:**

**Program Code (Executed as a script):**



**Output:**

****



**Result:** The program code is successfully executed, and output is verified.

**Learning Outcome:**

1. Learnt how to use Octave online.
2. Learnt how to run scripts in octave online.
3. Learnt about inbuilt functions of octave online.

Experiment - 3

**Aim:**

Our aim in this practical is to write a program to implement testing of Random numbers (Chi Square)

**Tools/Software Required:**

MATLAB ONLINE by MathWorks

**Theory:**

**Pearson’s Chi-Squared test:**

* It is method of testing the significance of difference between two proportions.
* Advantage: It can be used when more than two groups are to be compared.
* It is used to test categorical data.

**Steps of Chi-Squared Test:**

Step 1: Null hypothesis

Step 2: Calculation of expected values.

Step 3: Calculation of Chi-Square value.

Step 4: Calculation of degree of freedom.

**Kolmogorov-Smirov Test:**

* In essence, the test answers the question "What is the probability that this collection of samples could have been drawn from that probability distribution?" or, in the second case, "What is the probability that these two sets of samples were drawn from the same (but unknown) probability distribution?".

**Steps of Kolmogorov-Smirov Test:**

Step 1: Rank the data from smallest to largest.

Step2: Compute D+ max {i/N —Ri} and max {Ri —(i-1)/N}

Step 3: Find D = max {D+, D-}

Step 4: Determine the critical value, Du, from Table for the specified

value of a and the sample size N.

Step 5: If the sample statistic D is greater than the tabulated value of Du, the null hypothesis that the data are a sample from uniform distribution is rejected.

**Built-in Function used in this experiment:**

**h = kstest(x):** It returns a test decision for the null hypothesis that the data in vector x comes from a standard normal distribution, against the alternative that it does not come from such a distribution, using the one-sample Kolmogorov-Smirnov test. The result h is 1 if the test rejects the null hypothesis at the 5% significance level, or 0 otherwise.

**h = chi2gof(x):** It returns a test decision for the null hypothesis that the data in vector x comes from a normal distribution with a mean and variance estimated from x, using the chi-square goodness-of-fit test. The alternative hypothesis is that the data does not come from such a distribution. The result h is 1 if the test rejects the null hypothesis at the 5% significance level, and 0 otherwise.

**X = randn(sz1,...,szN):** It returns an sz1-by-...-by-szN array of random numbers where sz1,...,szN indicate the size of each dimension. For example, randn (3,4) returns a 3-by-4 matrix.

**Algorithm/Procedure:**

**Step 1:** Generate an array of random numbers & store it in a variable.



**Step 2:** Perform KS test on the array of random number stored in the variable “x” by using inbuilt MATLAB function,kstest(). Store a result in a variable lets say h.



**Step 3:** Perform Chi-Sqaure test on the array of random number stored in the variable “x” by using inbuilt MATLAB function,chi2gof(). Store a result in a variable lets say h1.



**Step 4:** Check whether the NULL hypothesis is accepted by checking value h & h1 in above steps.

Graphical user interface, text, application

Description automatically generated with medium confidence

**Step 5:** If h = 0 🡪 Null Hypothesis is accepted. If h = 1 🡪 Null Hypothesis is rejected.

Conclusion: As h = 0 in both tests, Null Hypothesis is accepted.

**Program Code with Output:**

**Graphical user interface, text, application

Description automatically generated**

**Result:** The program code is successfully executed, and output is verified.

**Learning Outcomes:**

* Learned about Chi-Square Test.
* Learned about Kolmogorov-Smirov Test.
* Learned how to implement both the above tests on MATLAB.

Experiment - 4

**Aim:**

To implement Monte Carlo Simulation.

**Tools/Software Required:**

MATLAB

**Theory:**

* Monte Carlo simulation is a method for exploring the sensitivity of a complex system by varying parameters within statistical constraints. These systems can include financial, physical, and mathematical models that are simulated in a loop, with statistical uncertainty between simulations. The results from the simulation are analyzed to determine the characteristics of the system.

1. Common tasks for performing Monte Carlo analysis include:
2. Varying uncertain parameters for your model
3. Creating dynamic simulations and alter parameters with statistical uncertainty
4. Creating a Monte Carlo simulation to model a complex dynamic system
5. Distributing simulations between processor cores and individual PCs to speed analysis
6. Analyzing data through robust plotting and advanced statistical methods

* In this experiment we will be implementing Monte Carlo Simulation by using the example problem of Buffon needle problem. Buffon's needle problem asks to find the probability that a needle of length l will land on a line, given a floor with equally spaced parallel lines a distance d apart.

**Pseudo code/Algorithms/Flowchart/Steps:**

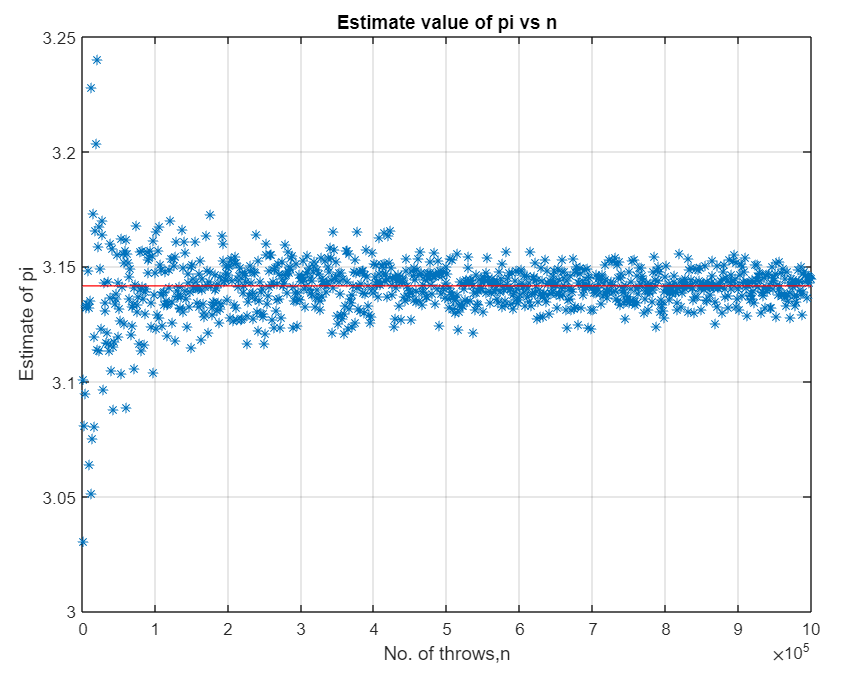
1. Make a variable for length of the needle and distance between the needles. Make sure length < distance.
2. Create a for loop for running trials for 10000 times.
3. Now inside the for loop, use rand function to generate n random numbers within the range of 0 & half the distance variable value for x.
4. Also generate n random numbers with the range of 0 & pie for theta
5. Create a loop for calculating the number of crossings
6. Now calculate pie and pie estimate using the following formulas
7. Plot the graph of pie estimate versus the number of throws.
8. End

**Implementation:**

Graphical user interface, text, application

Description automatically generated

**Output:**

****

**Learning Outcome:**

1. Learnt about Monte Carlo Simulation.
2. Learnt about Buffon needle problem.
3. Learnt how to simulate the Buffon needle problem on MATLAB.

Experiment - 5

**Aim:**

To implement the simulation of the LAG model.

**Tools/Software Required:**

MATLAB ONLINE by MathWorks

**Theory:**

In statistics and econometrics, a distributed lag model is a model for time series data in which a regression equation is used to predict current values of a dependent variable based on both the current values of an explanatory variable and the lagged (past period) values of this explanatory variable. In this experiment we will simulate a simple Lag model by initializing a matrix and a lag matrix using built in function of MATLAB.

**Pseudo code/Algorithms/Flowchart/Steps:**

Start

**Step1:** Initialized a matrix Y = [1 -1; 2 -2 ;3 -3 ;4 -4 ;5 -5] and a lag matrix: lags = [0 1 2].

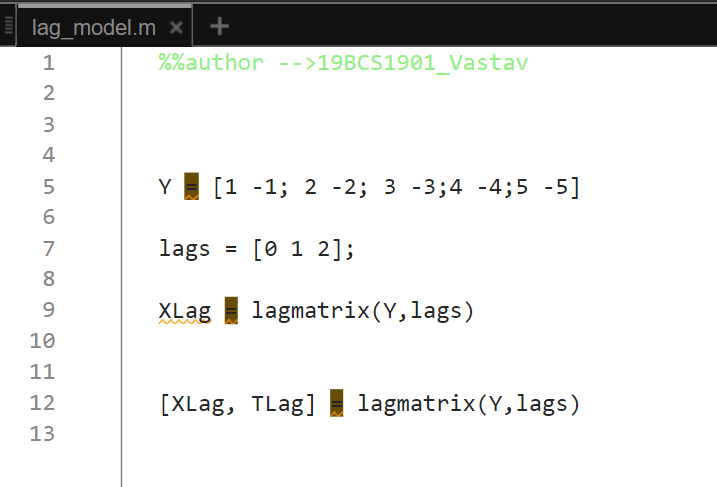
**Step2:** Performed the lagmatrix function in matlab: XLag = lagmatrix(Y,lags).

**Step3:** Create a shifted matrix, which is composed of the original X and its first two lags:

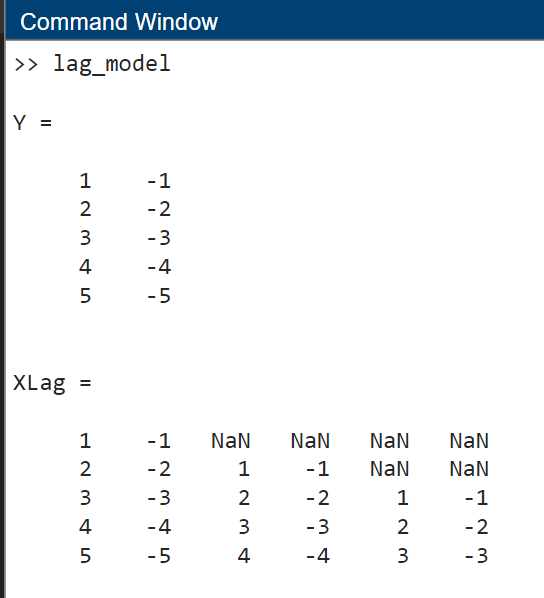
[XLag,TLag] = lagmatrix(Y,lags).

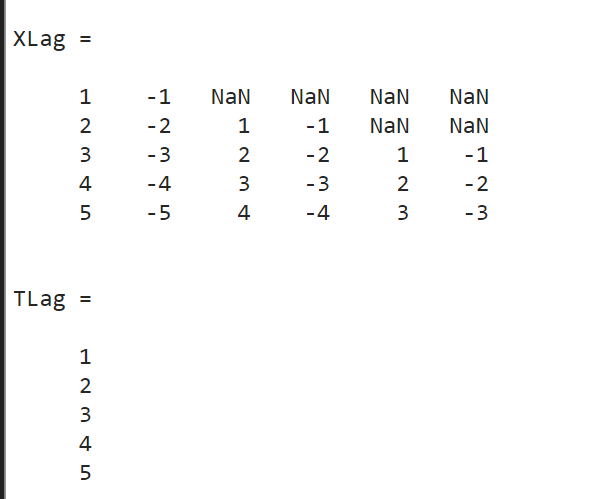
End

**Implementation/Program Code:**



**Output:**





**Result:** The program code is executed successfully, and the output is verified.

**Learning Outcomes:**

1. Learnt about lag model.
2. Learnt about the inbuilt function of MATLAB called lagmatrix.
3. Learnt how to simulate Lag model on MATLAB.

Experiment - 6

**Aim:**

To implement simulation of Single Queue Server System.

**Tools/Software Required:**

MATLAB ONLINE by MathWorks

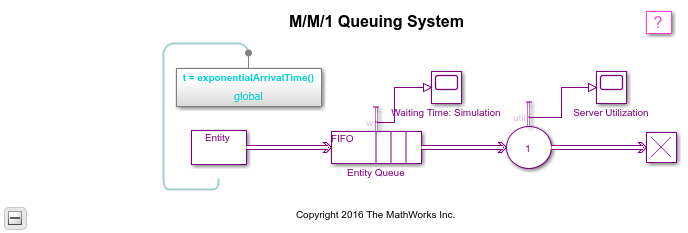
**Theory:**

This example shows how to model a single-queue single-server system with a single traffic source and an infinite storage capacity. In the notation, the M stands for Markovian; M/M/1 means that the system has a Poisson arrival process, an exponential service time distribution, and one server. Queuing theory provides exact theoretical results for some performance measures of an M/M/1 queuing system and this model makes it easy to compare empirical results with the corresponding theoretical results.

**Structure**

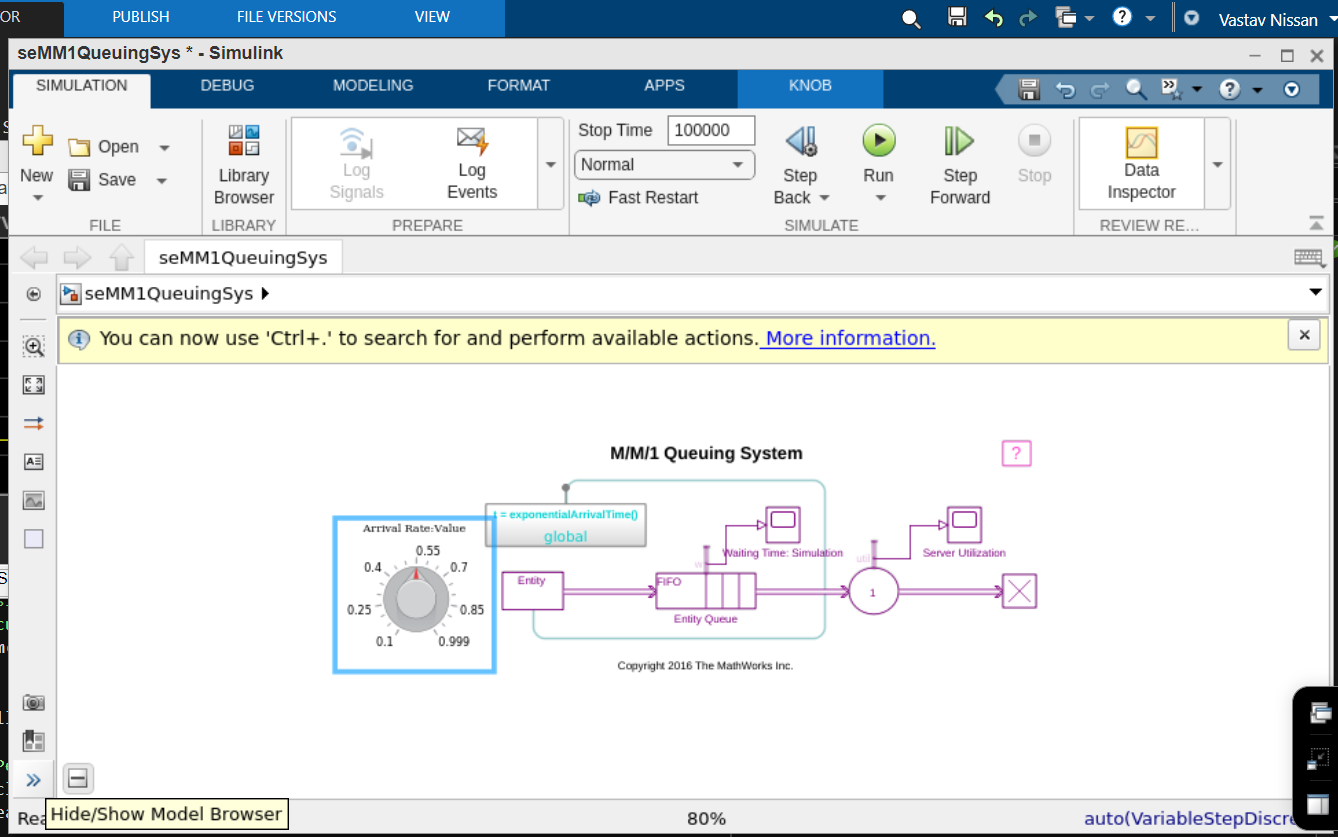
The model includes the components listed below:

1. **Time Based Entity Generator block**: It models a Poisson arrival process by generating entities (also known as "customers" in queuing theory).
2. **Exponential Interarrival Time Distribution subsystem**: It creates a signal representing the interarrival times for the generated entities. The interarrival time of a Poisson arrival process is an exponential random variable.
3. **FIFO Queue block**: It stores entities that have yet to be served.
4. **Single Server block**: It models a server whose service time has an exponential distribution.

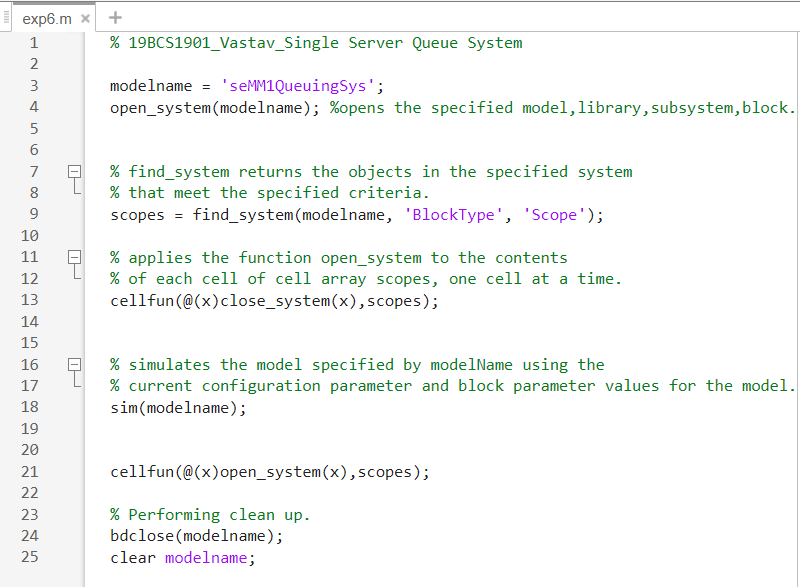


**Pseudo code/Algorithms/Flowchart/Steps:**

1. Give a model name to the queuing system by making a variable.
2. Use open\_system() with above variable as argument to open the model from the Mathworks library.
3. Now use find\_system() to return the objects in our specified system that meet simulation criteria.
4. Now, apply the function to the contents of each cell of cell array, one cell at a time.
5. Now perform simulation by using in-built function sim ().
6. Run the script and then run the Simulink model for simulation results.
7. Lastly perform clean up and clear the model by using clear command.



**Implementation/Program Code:**

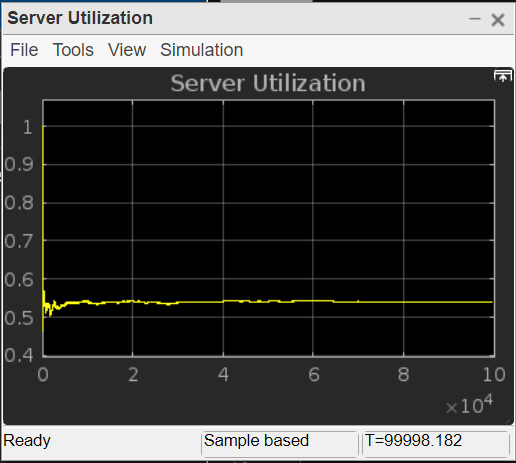


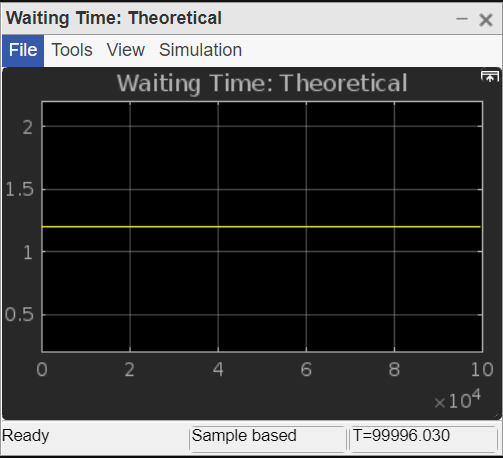
**Output:**

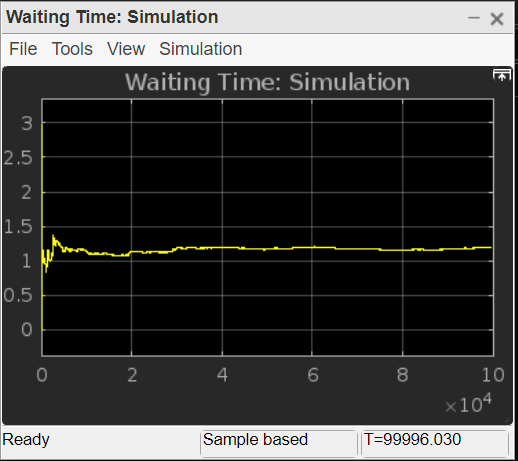
The model includes these visual ways to understand its performance:

Scopes labeled "Waiting Time: Theoretical" and "Waiting Time: Simulation" showing the theoretical and empirical values of the waiting time in the queue, on a single set of axes. We can use this plot to see how the empirical values evolve during the simulation and compare them with the theoretical value.

A scope labeled "Server Utilization" showing the utilization of the single server over the course of the simulation.







**Result:** The program code is executed successfully, and the output is verified.

**Learning Outcomes:**

1. Learnt about Single Queue server system.
2. Learnt about different in-built functions such as open\_system, cellfunc etc.
3. Learnt how to simulate single queue server systems on MATLAB.

Experiment - 7

**Aim:**

Write a program to implement simulation of Multiple Queue Server System.

**Tools/Software Required:**

MATLAB ONLINE by MathWorks

**Theory:**

In this experiment we will implement simulation of Multiple Queue Server System comprising of 2 separate infinite queues (FIFO-LIFO). The interarrival time will be distributed exponentially i.e., mean = 30 min. FIFO single server having a service time of 25min and LIFO single server service time of 20 min. We will compare these two queues in terms of waiting time, the number of entities in the queue and their server in terms of utilization. We will use two queue-server pairs connected in parallel, in which a copy of each entity arrives at both, represent a multicasting situation such as sending a message to multiple recipients. Note that copying entities might not make sense in some applications.

**Pseudo code/Algorithms/Flowchart/Steps:**

Start

Step 1. Open Simulink on MATLAB.

Step 2. Add various entity generators using Simulink library browser, sim events.

Step 3. Edit block parameters as per simulation requirements.

Step 4. Add FIFO and LIFO queue from Simulink library present under servers.

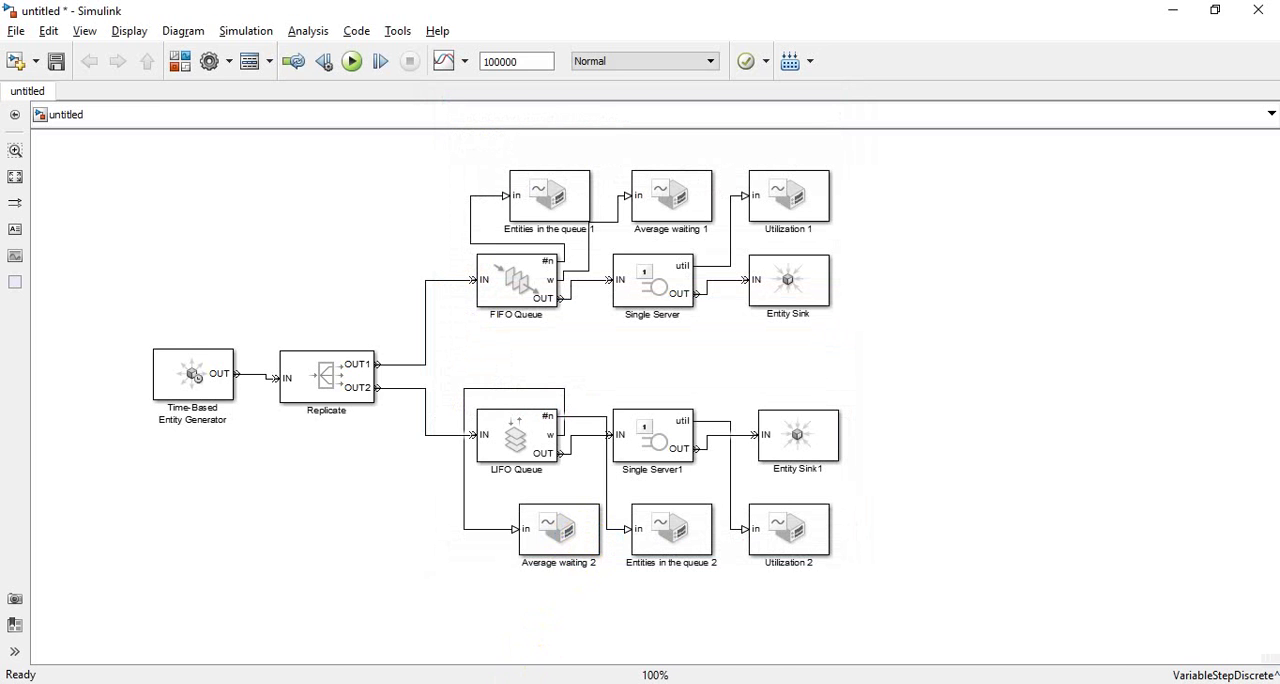
Step 5. Edit block parameters of FIFO and LIFO queues.

Step 6. Add relevant statistics according to the requirements and click on for simulation results.

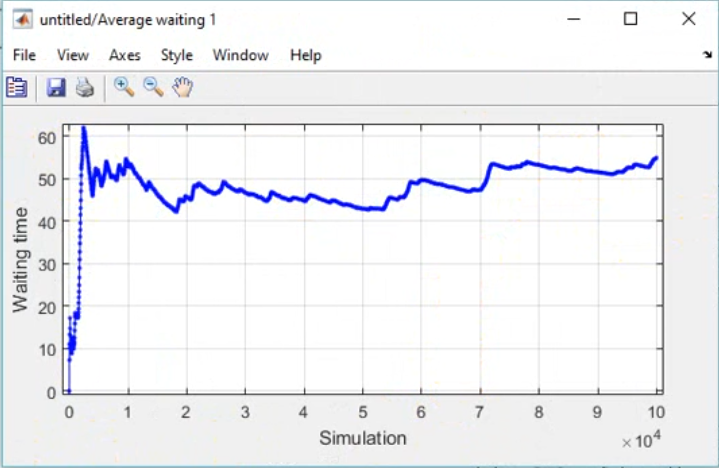
End

**Implementation:**

After adding all the required blocks from Simulink library.

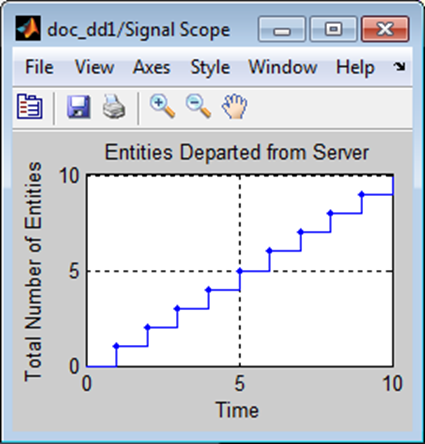


**Output:**



Chart, line chart

Description automatically generated



**Result**: The program code is executed successfully, and the output is verified.

**Learning Outcomes:**

1. Learnt about LIFO and FIFO servers.
2. Learnt about logical queues.
3. Learnt how to simulate using Simulink.

Experiment - 8

**Aim:**

Write a program to implement simulation of inventory system.

**Tools/Software Required:**

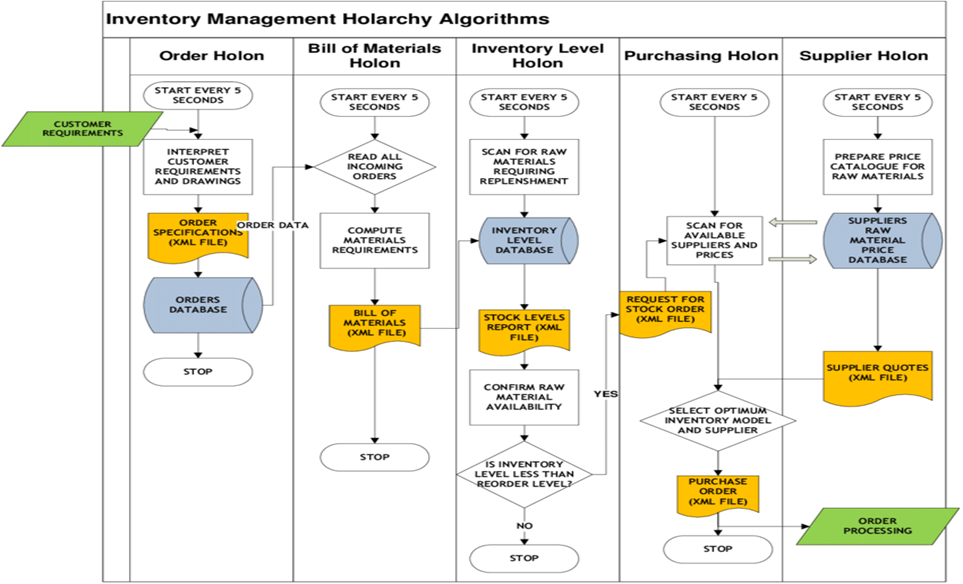
MATLAB ONLINE by MathWorks

**Theory:**

In this experiment we will demonstrate how to build a simple inventory management system for a retail store. This demonstration includes:

1. Random customer arrivals to the store with the number of products requested by each customer also randomly distributed.
2. Tracking available inventory at the end of the day.
3. Tracking and disposal of expired products.
4. Placing periodic orders for fresh products.
5. Store profitability analysis.

**Flowchart:**



**Pseudo code/Algorithms/Flowchart/Steps:**

Start

Step 1. Set the model’s name by initializing it to “seInventoryManagement”.

Step 2. Open the model using built-in MATLAB function called “open\_system”.

Step 3. Then use the function “open\_system” multiple times to open Warehouse, Transportation, Store Management Model at once.

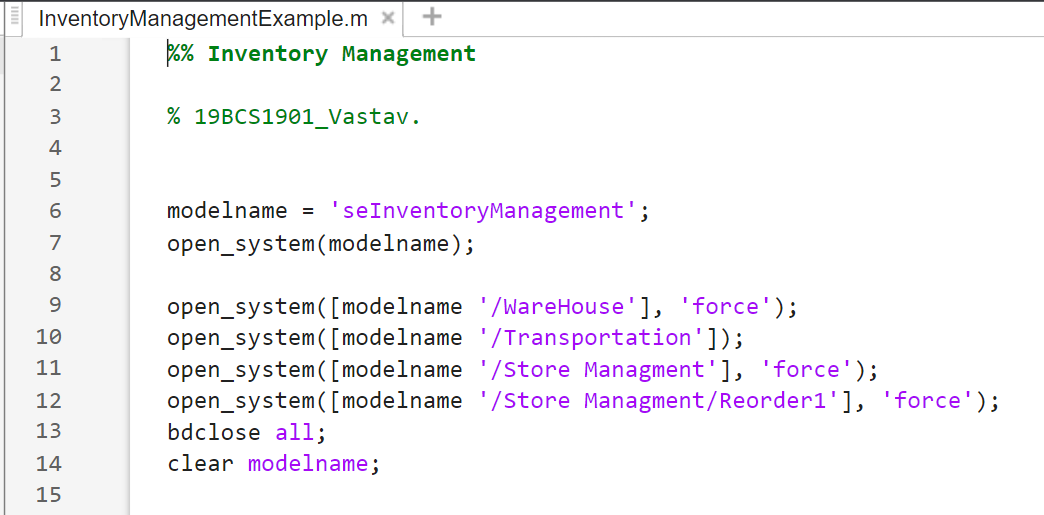
Step 4. Close the models and clear model name variable.

Step 5. Clear the model’s name variable.

Step 6. Run the simulation in Simulink for analysing results.

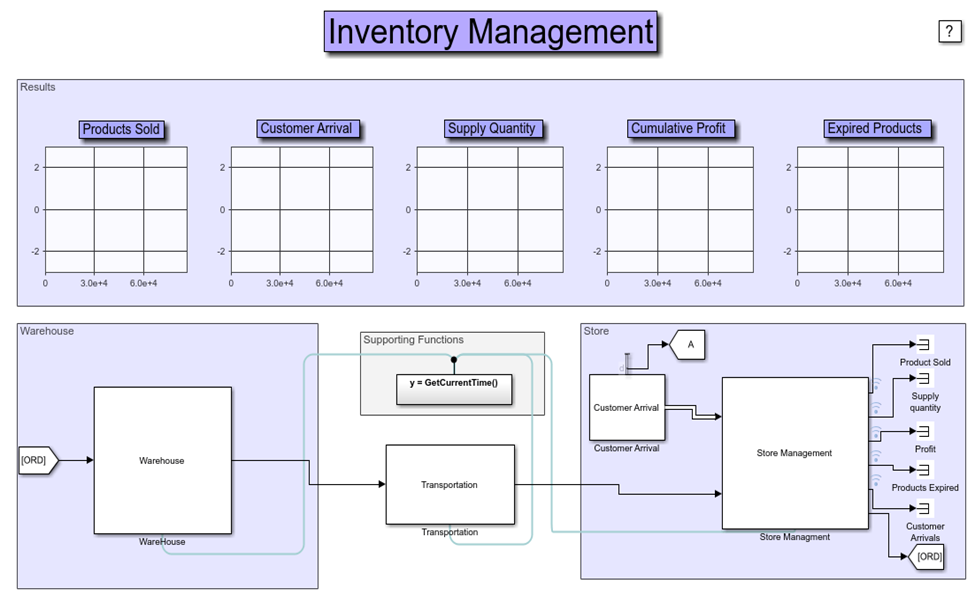
End

**Implementation/Program Code:**



**Output:**

Before Simulation:



After Simulation:

Graphical user interface, diagram

Description automatically generated

The model is simulated for 60 days. One unit of simulation time represents 1 minute of wall clock time. Based on the model parameters set, plots are generated showing the number of products sold, the number of customers who arrived at the store, the product order size, number of expired products in the store and the store's profitability. Observe that for each period, the optimal store order quantity is around 85 for the given customer arrival rate.

**Result**: The program code is executed successfully, and the output is verified.

**Learning Outcomes:**

1. Implemented and discussed various aspects of simulating an inventory management system.
2. Simulated store as well as customer demand using Simulink.
3. Simulated management of various aspects of a warehouse.

Experiment - 9

**Aim:**

Write a program to implement simulation of telephonic system.

**Tools/Software Required:**

MATLAB ONLINE by MathWorks

**Theory:**

Lost calls are any inbound calls that do not result in the caller being connected to either an advisor or an answering service. This happens when the customer hangs up or is disconnected by the centre. The centre may disconnect a call deliberately, because of timeout – the call was taking too long to reach an agent – or through error.

Categories of Lost Call:

• **Abandoned** – The customer terminates the call before it is answered. The industry standard suggests that an abandon rate of 2-5% is commonplace.

• **Missed** – The call is deliberately disconnected by the centre. This usually occurs when an incoming call reaches the maximum threshold for waiting time set by the ACD.

• **Dropped** – The call is accidentally disconnected due to a technical error. The dropped call rate for landline calls is below 0.01%. The rate for mobile phone calls is slightly higher, but a dropped call rate of even a few per cent would warrant further investigation.

**Pseudo code/Algorithms/Flowchart/Steps:**

Start

Step 1. Scan the potential event.

Step 2. Select the activity that is to cause the event.

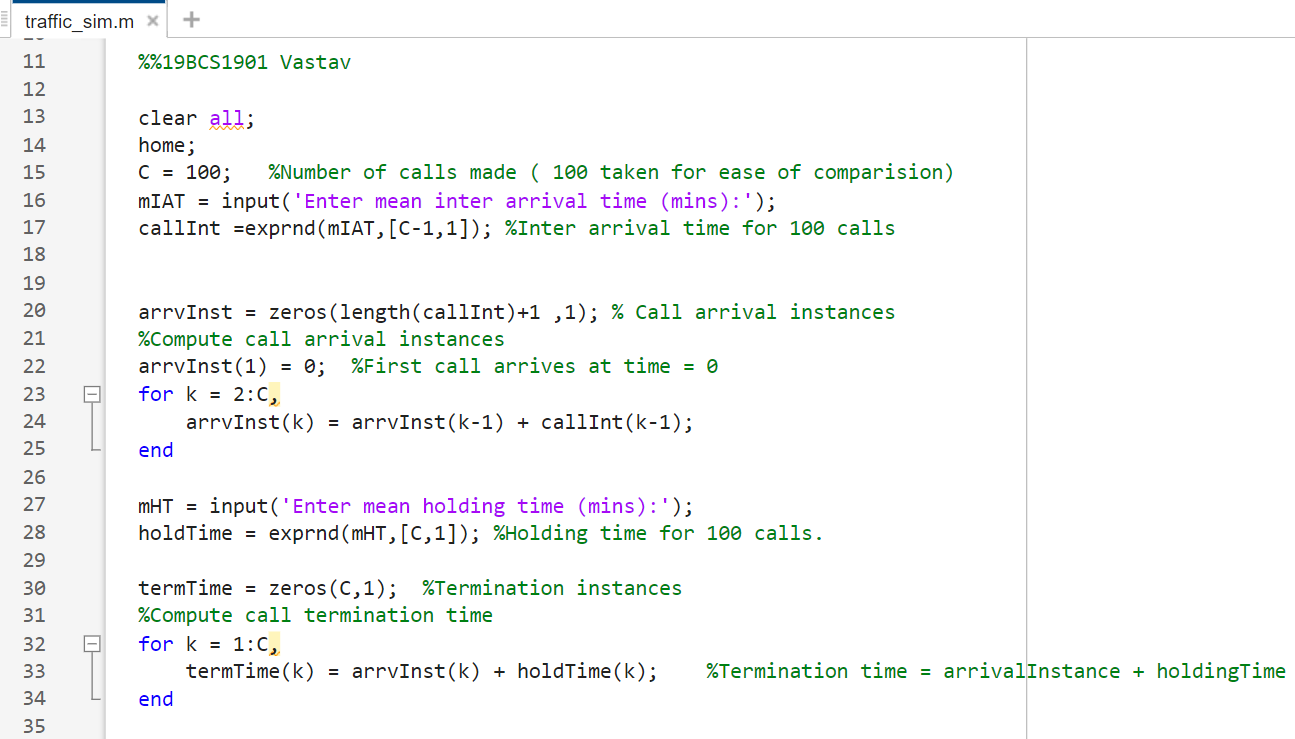
Step 3. Test whether the potential event that can be executed.

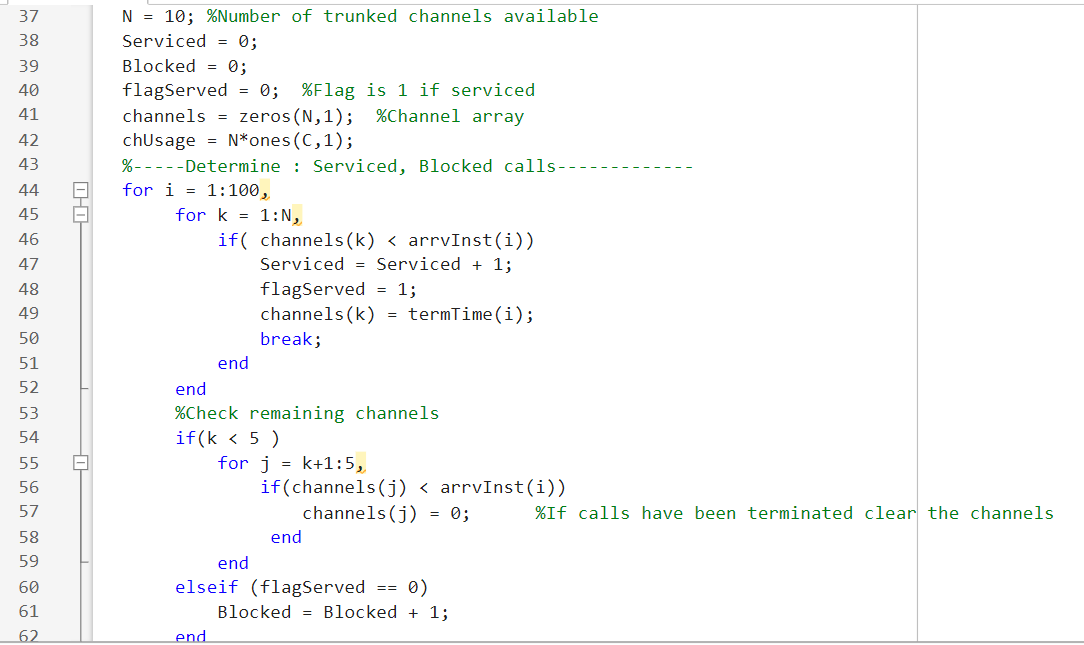
Step 4. Change records to reflect the event.

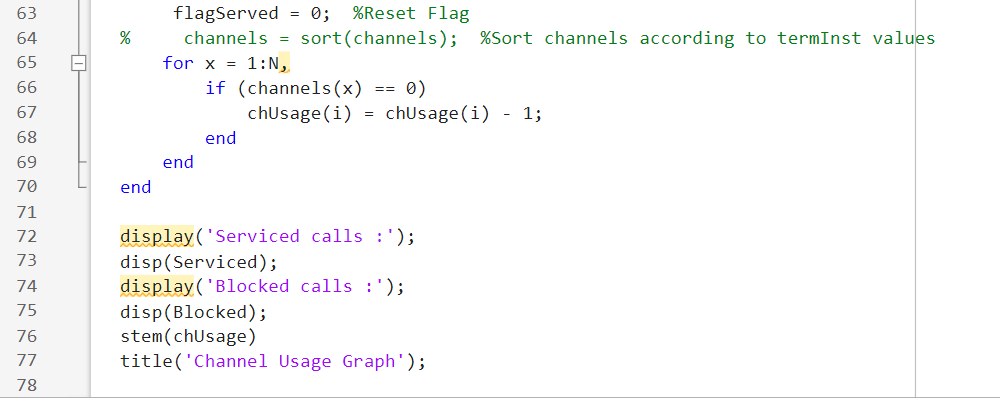
Step 5. Gather the statistics of simulation output.

End

**Implementation/Program Code:**

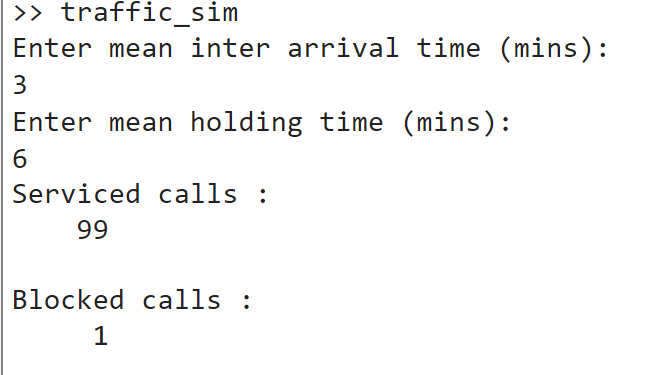






**Output:**

Simulation of a trunked communication network with N channels. Total of 100 calls are considered for simulation. The network is a Lost Calls Cleared type system.



Chart

Description automatically generated

**Result**: The program code is executed successfully, and the output is verified.

**Learning Outcomes:**

1. Learnt about implement telephone management system in MATLAB.
2. Learnt about differentiate between various categories of lost call log.
3. Learnt about simulate a lost call log telephonic system.

Experiment - 10

**Aim:**

Write a program to implement COBWEB model.

**Tools/Software Required:**

MATLAB ONLINE by MathWorks

**Theory:**

The cobweb model or cobweb theory is an economic model that explains why prices might be subject to periodic fluctuations in certain types of markets. It describes cyclical supply and demand in a market where the amount produced must be chosen before prices are observed. The cobweb model is generally based on a time lag between supply and demand decisions.

The cobweb model can have four types of outcomes:

1. If the supply curve is steeper than the demand curve, then the fluctuations decrease in magnitude with each cycle, so a plot of the prices and quantities over time would look like an inward spiral, as shown in the first diagram. This is called the stable or convergent case.
2. If the demand curve is steeper than the supply curve, then the fluctuations increase in magnitude with each cycle, so that prices and quantities spiral outwards. This is called the unstable or divergent case.
3. Fluctuations may also maintain a constant magnitude, so a plot of the outcomes would produce a simple rectangle. This happens in the linear case if the supply and demand curves have the same slope (in absolute value).
4. If the supply curve is less steep than the demand curve near the point where the two curves cross, but steeper when we move sufficiently far away, then prices and quantities will spiral away from the equilibrium price but will not diverge indefinitely; instead, they may converge to a limit cycle.

In this experiment we will generate the COWEB plot associated with the orbits x\_n+1=f(x\_n) over interval (a,b).

**Pseudo code/Algorithms/Flowchart/Steps:**

Start

Step 1. Create two function files called “cobweb” and “f”.

Step 2. In the coweb function file call the cobweb function.

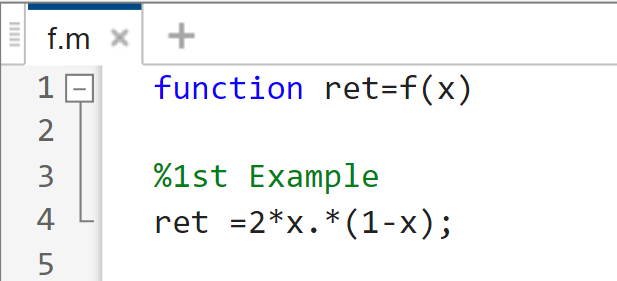
Step 3. Then generate N linearly spaced values on (a,b).

Step 4. Plot the values in step 3 using y=f(x) function.

Step 5. Then turn hold on to gather up all plots in one command window.

Step 6. In the function file called “f” that we created in step 1.

Write down the following code:

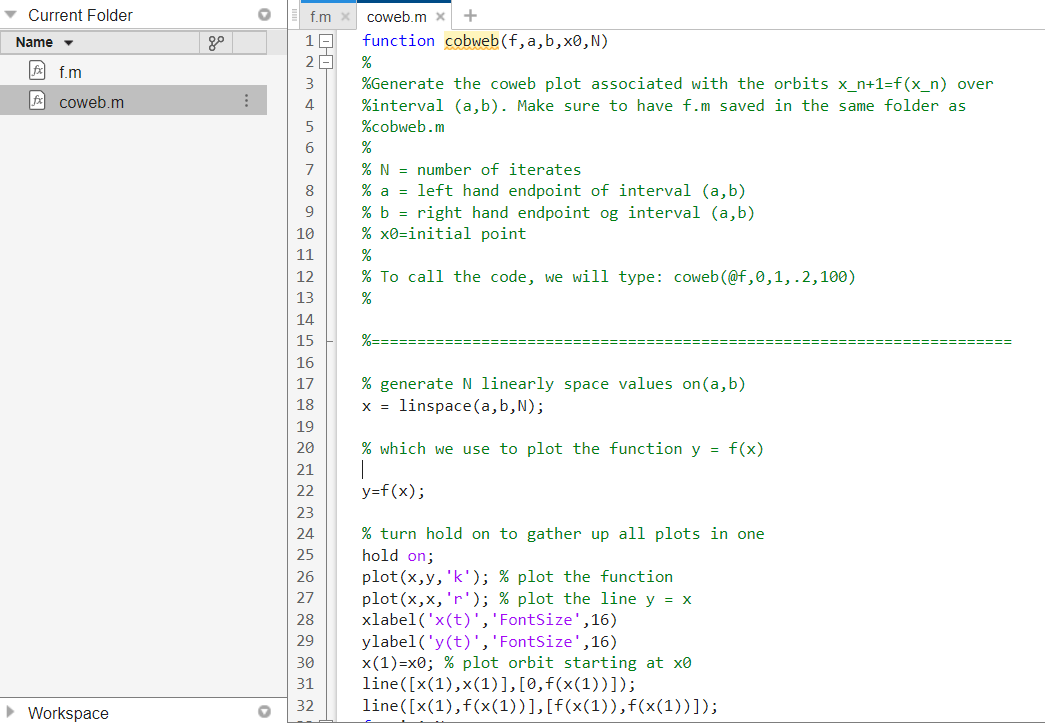


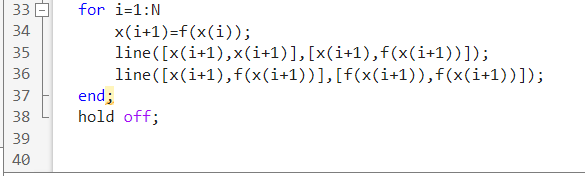
Lastly call the cobweb function in the command window.

End

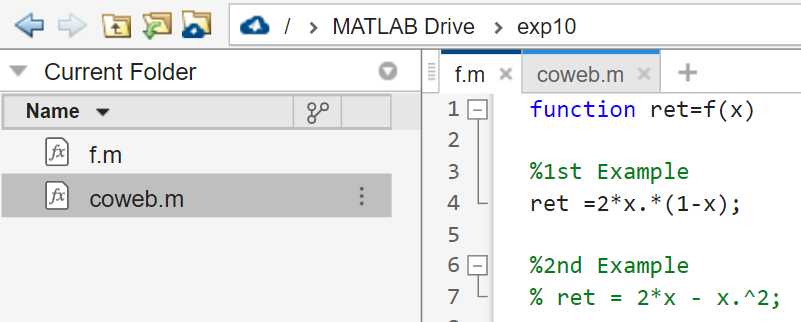
**Implementation/Program Code:**

**Cobweb.m function file:**





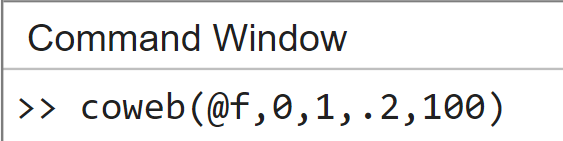
**f.m function file:**



**Output:**

The output graph will be COBWEB which will depend upon the input function and values that we chose.

Our Chose Values:



Chart, line chart

Description automatically generated

**Result**: The program code is executed successfully, and the output is verified.

**Learning Outcomes:**

1. Learnt about COBWEB graph.
2. Learnt how to interpret the curves of the COBWEB graph.
3. Learnt how to demonstrate the effect of input and time on the curves of the COBWEB graph.