

**LABORATORY MANUAL**

**OF**

**SIMULATION PRACTICE**

**ON**

**MATLAB**

**For**

**5<sup>th</sup> sem, Electrical Engg. (Diploma)**



**DEPARTMENT OF ELECTRICAL ENGINEERING**

**GOVERNMENT POLYTECHNIC, BARGARH**

**Prepared by:**

**Rashmita Gouda**

**Lecturer in Instrumentation & Control Engg.**

# LIST OF EXPERIMENTS IN MATLAB SIMULATION SOFTWARE

## **1. Introduction to MATLAB programming:**

1.1. Functions and operation using variables and arrays.

1.1.1. To learn algebraic, trigonometric and exponential manipulation.

1.1.2. To learn Arithmetic, Relational and Logic operator.

1.2. Matrix formation and its manipulation.

1.3. Vector manipulation:

1.3.1. Use of linspace to create vectors.

1.3.2. To create, add and multiply vectors.

1.3.3. Use of sin and sqrt functions with vector arguments.

1.4. Plotting:

1.4.1. Two dimensional Plots and sub plots

1.4.2. Label the plot and printing.

1.5. Write and execute a file to plot a circle, impulse, step, ramp, sine and cosine functions. .

## **2. Introduction to SIMULINK:**

2.1. Use of Commonly used blocks, Math operation block and Display block from SIMULINK library.

2.2. Use of logical and relational operator block.

2.3. Use of Sim-Power system block to use Electrical sources, elements and Power electronics devices.

2.4. SIMULATION:

2.4.1. Verification of Network theorems. (Any two)

2.4.2. Simulation of a half wave uncontrolled rectifier.

2.4.3. Simulation of 1-phase full bridge controlled rectifier.

2.4.4. Simulation of step-down chopper.

## INTRODUCTION TO MATLAB:

- The name MATLAB stands for **MA**TriX **LA**boratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.
- MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.
- MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide.
- It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.
- **Starting MATLAB:** you can enter MATLAB by double-clicking on the MATLAB shortcut icon on your Windows desktop. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains other windows. The major tools within or accessible from the desktop are:
  - The Command Window
  - The Command History
  - The Workspace
  - The Current Directory
  - The Help Browser
  - The Start button

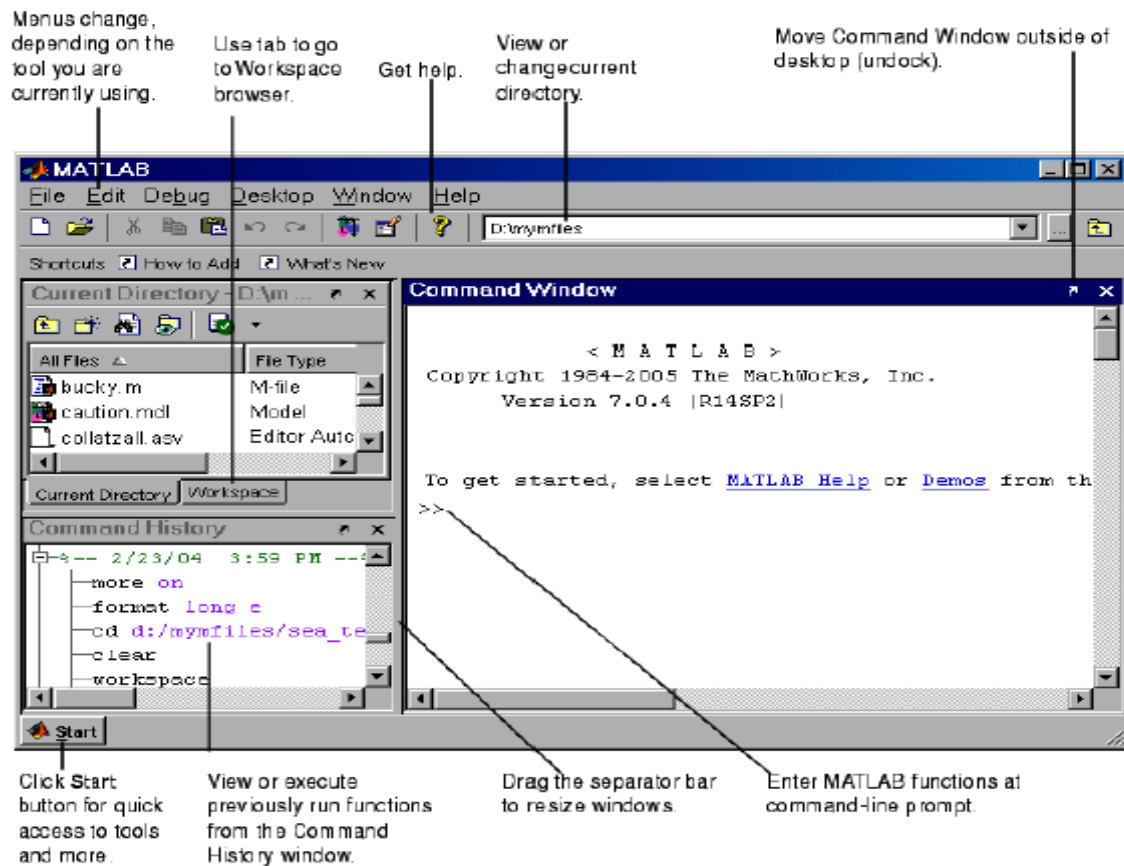


Fig:1 The graphical interface to the MATLAB workspace

When MATLAB is started for the first time, the screen looks like the one that shown in the Figure 1.1. This illustration also shows the default configuration of the MATLAB desktop. You can customize the arrangement of tools and documents to suit your needs. Now, we are interested in doing some simple calculations. We will assume that you have sufficient understanding of your computer under which MATLAB is being run. You are now faced with the MATLAB desktop on your computer, which contains the prompt (>>) in the Command Window.

### PROCEDURE:-

- Open MATLAB
- Open new M-file 21
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

## **EXPERIMENT NO.: 1(a)**

**AIM OF THE EXPERIMENT:** Write a MATLAB script for solving the algebraic equation ' $x^2 - 12x + 35 = 0$ '.

### **SOFTWARE REQUIRED:**

1. MATLAB R2018a.

### **THEORY:**

#### **Algebraic functions:**

`solve(eqn, x)` :

If eqn is an equation, `solve(eqn, x)` solves eqn for the symbolic variable x. Use the `==` operator to specify the familiar quadratic equation and solve it using `solve`.

#### **Syntax:**

`S = solve(eqn,var)`

### **SCRIPT :**

```
>> eqn = x^2-12*x+35==0 ;
```

```
>> s= solve(eqn);
```

```
>> disp(s(1));
```

```
>> disp(s(2));
```

### **Conclusion:**

## EXPERIMENT NO.: 1(b)

**AIM OF THE EXPERIMENT:** Write a MATLAB script to perform trigonometric manipulation by using trigonometric functions.

**SOFTWARE REQUIRED:-**

1. MATLAB R2018a.

**THEORY:**

**Trigonometric functions:**

The trigonometric functions in MATLAB calculate standard trigonometric values in radians or degrees.

Functions	uses
sin()	Sine of argument in radians
sind()	Sine of argument in degrees
cos()	Cosine of argument in radians
cosd()	Cosine of argument in degrees
tan()	Tangent of argument in radians
tand()	Tangent of argument in degrees
csc()	Cosecant of input angle in radians
cscd()	Cosecant of argument in degrees
sec()	Secant of angle in radians
secd()	Secant of argument in degrees
cot()	Cotangent of angle in radians
cotd()	Cotangent of argument in degrees
deg2rad()	Convert angle from degrees to radians
rad2deg()	Convert angle from radians to degrees

**SCRIPT:**

```
>> x=pi;  
>>a=sin(x)  
>>b=cos(x/2)  
>>c=tan(x/4)  
>>d=sin (x/2+x/6)
```

**Conclusion:**

## **EXPERIMENT NO.: 1(c)**

**AIM OF THE EXPERIMENT:** Write a MATLAB script to find the value of  $e^{10}$  using exponential function.

### **SOFTWARE REQUIRED:-**

1. MATLAB R2018a.

### **THEORY:**

#### **Exponential Functions:**

Syntax :

$$y = \exp(x)$$

This function returns the exponential  $e^x$  for each element in array x.

### **SCRIPT:**

```
>>x=10;
```

```
>>y=exp(x)
```

### **Conclusion:**

## EXPERIMENT NO.: 2(a)

**AIM OF THE EXPERIMENT:** Write a script to perform different arithmetic operations using MATLAB software.

### SOFTWARE REQUIRED:

1. MATLAB R2018a.

### THEORY:

The basic arithmetic operators used in MATLAB are listed below:

Operators	Uses
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Power

### SCRIPT:

```
a=5;
```

```
b=10;
```

```
r=a+b
```

```
s=a-b
```

```
t=a*b
```

```
u=a/b
```

```
v=a^3
```

### Conclusion:



## EXPERIMENT NO.: 2(b)

**AIM OF THE EXPERIMENT:** Write a script to compare two numbers using MATLAB software.

### SOFTWARE REQUIRED:

1. MATLAB R2018a.

### THEORY:

The relational operators used in MATLAB are listed below:

Operators	Uses
==	Determine equality
~=	Determine inequality
>	Determine greater than
>=	Determine greater than or equal to
<	Determine less than
<=	Determine less than or equal to

### SCRIPT:

```
a= input( 'Enter the 1st number' )  
b= input( 'Enter the 2nd number' )  
if (a>b)  
fprintf('%d is larger\n' ,a);  
else  
fprintf('%d is larger\n' ,b);  
end
```

### Conclusion:

## EXPERIMENT NO.: 2(c)

**AIM OF THE EXPERIMENT:** Write a script using logical operators in MATLAB software.

### SOFTWARE REQUIRED:

1. MATLAB R2018a.

### THEORY:

The basic logical operators used in MATLAB are listed below:

Operators	Uses
&&	Logical AND operations between two expressions / statements
	Logical OR operations between two expressions / statements
&	Find logical AND
~	Find logical NOT
	Find logical OR
xor	Find logical exclusive-OR
all	Determine if all array elements are nonzero or true
any	Determine if any array elements are nonzero
find	Find indices and values of nonzero elements
islogical	Determine if input is logical array
logical	Convert numeric values to logicals
true	Logical 1 (true)
false	Logical 0 (false)

### SCRIPT:

```
a= input( 'Enter the 1st number' )
b= input( 'Enter the 2nd number' )
if (a>0 && b>0)
disp('Both the numbers are positive integers');
else
disp('Both the numbers are not positive integers');
end
```

### Conclusion:

## EXPERIMENT NO.: 3(a)

**AIM OF THE EXPERIMENT:** Write a script for creation of matrix in MATLAB software.

### SOFTWARE REQUIRED:

1. MATLAB R2018a.

### THEORY:

All MATLAB variables are multidimensional arrays, no matter what type of data. A matrix is a two-dimensional array often used for linear algebra.

Array Creation:

- Specific set of data are arranged as elements in a matrix using a set of square brackets ([.....]).
- A single row of data has spaces or commas (,) in between the elements and semicolon (;) is used to separate the rows.
- Matrix of all zeros can be created by function zeros(m,n).
- Matrix of all ones can be created by function ones(m,n).
- Matrix of random numbers can be created by function rand(m,n).  
(Where  $m \times n$  is the dimension of matrix)

### SCRIPT:

```
a = [1,2,3;11,10,15;1,0,4]
```

```
b= zeros(2,3)
```

```
c= ones(2,2)
```

```
d= rand(3,1)
```

### Conclusion:

## EXPERIMENT NO.: 3(b)

**AIM OF THE EXPERIMENT:** Write a script to find transpose, inverse and

determinant of matrix  $a = \begin{bmatrix} 1 & 2 & 3 \\ 11 & 10 & 9 \\ 8 & 7 & 6 \end{bmatrix}$

**SOFTWARE REQUIRED:**

1. MATLAB R2018a.

**THEORY:**

- If A is a matrix then transpose of the matrix can be found by  $A'$ .
- `inv()` function is used to find inverse of a matrix.
- `det()` function is used to find determinant of a matrix.

**SCRIPT:**

```
a = [1,2,3;11,10,9;8,7,6]
```

```
b = a'
```

```
c = inv(a)
```

```
d = det(a)
```

**Conclusion:**

## **EXPERIMENT NO.: 3(c)**

**AIM OF THE EXPERIMENT:** Write a script for addition, subtraction, multiplication division of two matrixes using MATLAB software.

### **SOFTWARE REQUIRIED:**

1. MATLAB R2018a.

### **THEORY:**

- MATLAB allows to process matrixes using arithmetic operators ('+', '-', '\*', '/') or functions.

### **SCRIPT:**

```
a= [1, 2, 3; 4, 5, 6; 7, 8, 9];
```

```
b= [4, 6, 9; 3, 3, 1; 4, 11, 15];
```

```
w=a+b
```

```
x=a-b
```

```
y=a*b
```

```
z=a/b
```

### **Conclusion:**

## **EXPERIMENT NO.: 4(a)**

**AIM OF THE EXPERIMENT:** Write a MATLAB script for creating two vectors using linspace function and perform addition and multiplication operations of the two vectors.

### **SOFTWARE REQUIRED:**

1. MATLAB R2018a.

### **THEORY:**

- Row vector can be created by using linspace() function. This function takes 3 arguments i.e. initial limit, final limit and no. of elements. If no of elements is not specified then by default 100 elements are created.
- MATLAB allows to process vectors using arithmetic operators ('+', '-', '\*', '/') or functions.
- For element to element multiplication '.'\*' is used between two vectors.

### **SCRIPT:**

```
a= linspace(1,10,5)
```

```
b=linspace(2,25,5)
```

```
x=a+b
```

```
y=a.*b
```

```
z=a*b'
```

### **Conclusion:**

## **EXPERIMENT NO.: 4(b)**

**AIM OF THE EXPERIMENT:** Write a MATLAB script for using sin and sqrt functions with vector arguments.

### **SOFTWARE REQUIRED:**

1. MATLAB R2018a.

### **THEORY:**

- sqrt(X) returns the square root of each element of the array X. For the elements of X that are negative or complex, sqrt(X) produces complex results.
- sin(X) returns the sin of each element of the array X in radian.

### **SCRIPT:**

```
a= [1, 4, 9, 16, 25, 28]
```

```
b= sqrt(a)
```

```
x= [0, pi/2, pi/4, pi]
```

```
y=sin(x)
```

## **EXPERIMENT NO.: 5(a)**

**AIM OF THE EXPERIMENT:** Write a MATLAB script to plot sine and cosine functions and label it.

### **SOFTWARE REQUIRED:**

1. MATLAB R2018a.

### **THEORY:**

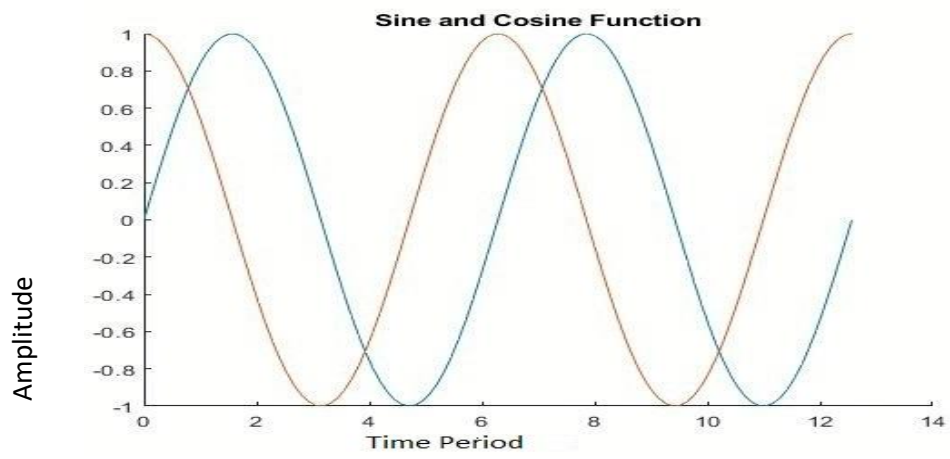
- `plot(x,y)` function is used to plot 2-D graphs. Where x is the x-axis variable and y is the y-axis variable.
- `xlabel('text')` and `ylabel('text')` are used to label the x-axis and y-axis respectively.
- `Title('text')` function is used to label the title of 2-D graph.

### **SCRIPT:**

```
a=linspace(0,pi*4);  
b=sin(a);  
c=cos(a);  
plot(a,b);  
plot(a,c);  
hold off  
xlabel('Time Period');  
ylabel('Amplitude');  
title('sine and cosine function');
```



## SIMULATION OUTPUT:



## Conclusion:

## EXPERIMENT NO.: 5(b)

**AIM OF THE EXPERIMENT:** Write a MATLAB script to subplot sine, cosine and tangent functions.

**SOFTWARE REQUIRED:**

1. MATLAB R2018a.

**THEORY:**

- subplot(m,n,p) function is used to divide the graph into  $m \times n$  grids.

Where, m= no. of rows

n= no of columns

p= position of the grid

**SCRIPT:**

```
a= linspace(0,2*pi,50);
```

```
x=sin(a);
```

```
y=cos(a);
```

```
z=tan(a);
```

```
subplot(2,3,1);
```

```
plot(a,x);
```

```
xlabel('t');
```

```
ylabel('sint');
```

```
label('sine plot');
```

```
subplot(2,3,3);
```

```
plot(a,y);
```

```
xlabel('t');
```

```
ylabel('cost');
```

```
label('cosine plot');
```

```
subplot(2,3,5);
```

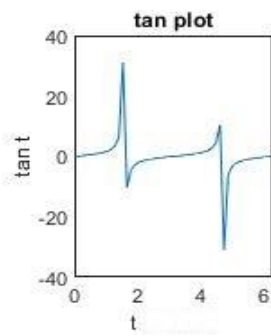
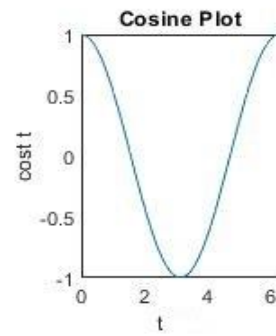
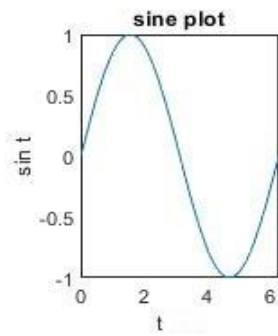
```
plot(a,z);
```

```
xlabel('t');
```

```
ylabel('tant');
```

```
label('tan plot');
```

### **SIMULATION OUTPUT:**



### **Conclusion:**

## EXPERIMENT NO.: 6(a)

**AIM OF THE EXPERIMENT:** Write a script to plot a circle with centre at point (5,7) and radius 2 ,by using MATLAB software.

### SOFTWARE REQUIRED:

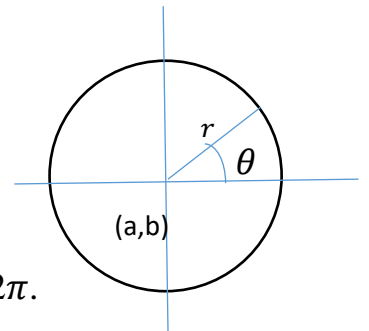
1. MATLAB R2018a.

### THEORY:

For any point (x,y) on the circle with centre at (a,b)

$$x = r\cos\theta + a$$

$$y = r\sin\theta + b$$

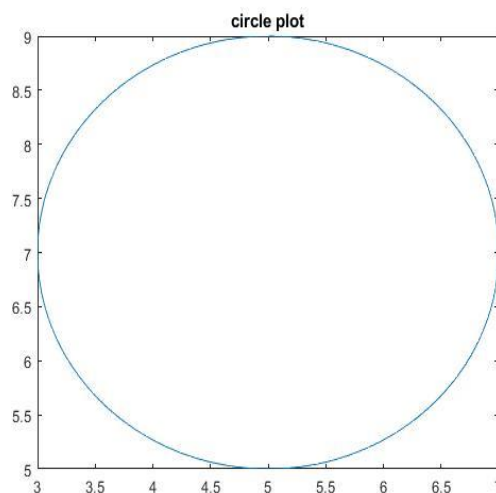


Where, r is the radius of the circle and  $\theta$  angle varies from 0 to  $2\pi$ .

### SCRIPT:

```
r=2;
theta= linspace(0,2*pi);
x= r * cos(theta) + 5;
y= r * sin(theta) + 7;
plot(x,y);
title('circle plot');
```

### SIMULATION RESULT:



### Conclusion:

## EXPERIMENT NO.: 6(b)

**AIM OF THE EXPERIMENT:** Write a script to plot an unit Ramp function by using MATLAB software.

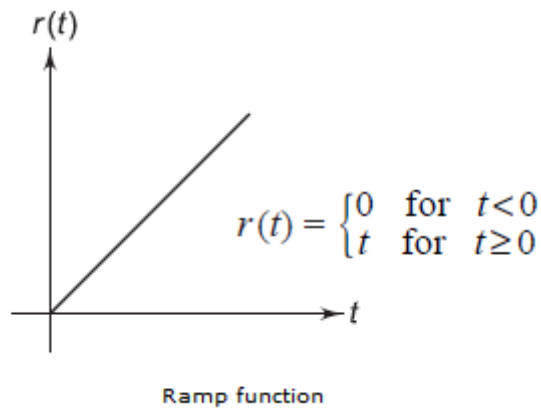
**SOFTWARE REQUIRED:**

1. MATLAB R2018a.

**THEORY:**

The unit Ramp signal is denoted by  $r(t)$

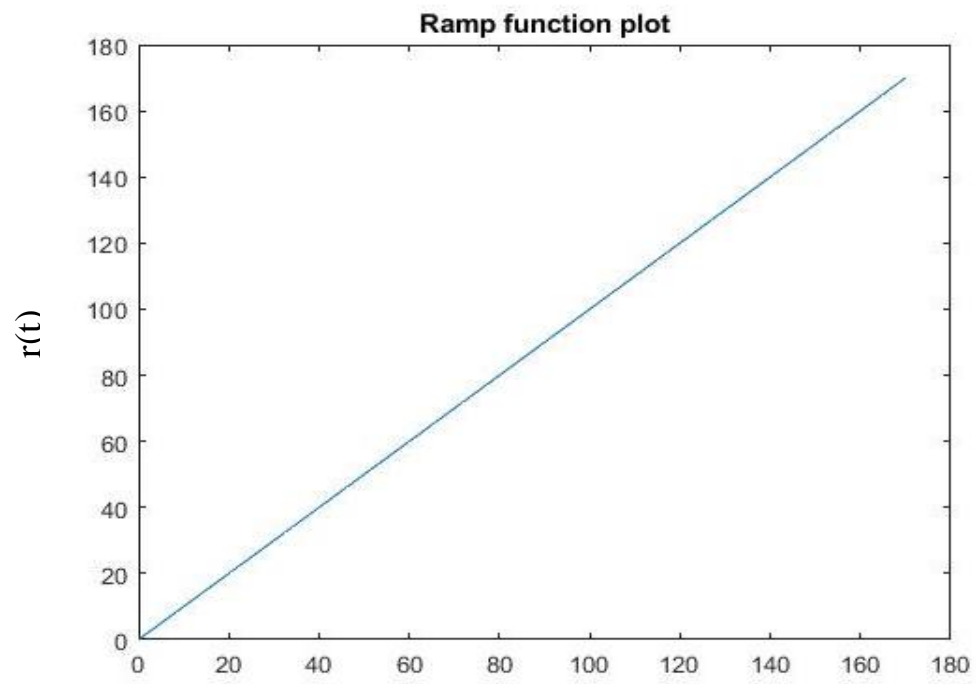
- It is defined as  $r(t) = \{t, t \geq 0 \text{ and } 0, t < 0$
- Area under unit ramp is unity.



**SCRIPT:**

```
x=linspace(0,100)
y=1*x;
plot(x,y);
xlabel('t');
ylabel('r(t)');
title('Ramp function plot');
```

## SIMULATION RESULT:



## Conclusion:

## EXPERIMENT NO.: 6(c)

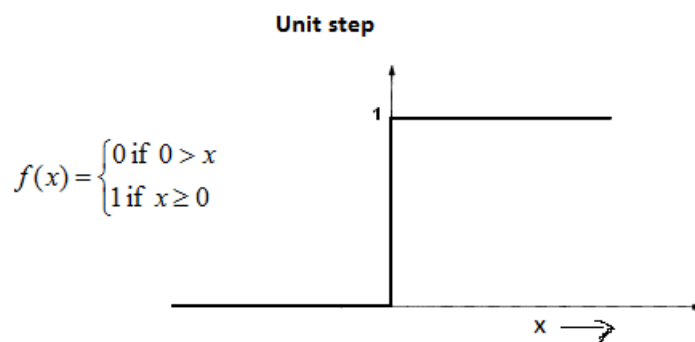
**AIM OF THE EXPERIMENT:** Write a script to plot a unit Step function by using MATLAB software.

### SOFTWARE REQUIRED:

1. MATLAB R2018a.

### THEORY:

- A function that increases or decreases abruptly from one constant value to another is called as Step function.



- heaviside( x ) function in MATLAB evaluates the Heaviside step function (also known as the unit step function) at x . The Heaviside function is a discontinuous function that returns 0 for x < 0 ,

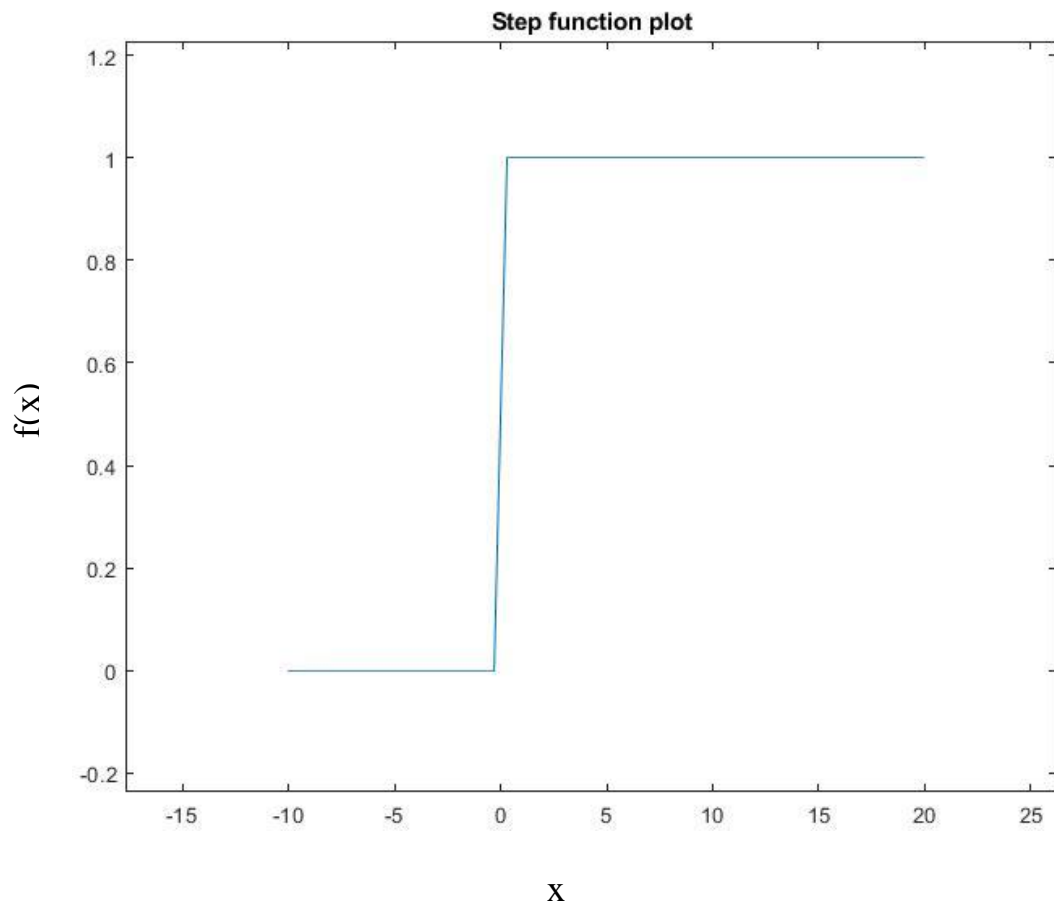
1/2 for x = 0 ,

and 1 for x > 0 .

### SCRIPT:

```
x=linspace(-10,20);  
y=heaviside(x);  
plot(x,y);  
xlabel('t');  
ylabel('r(t)');  
title('Step function plot');
```

## SIMULATION RESULT:



**Conclusion:**



## EXPERIMENT NO.: 6(d)

**AIM OF THE EXPERIMENT:** Write a script to plot an Impulse, unit step and ramp functions by using MATLAB software.

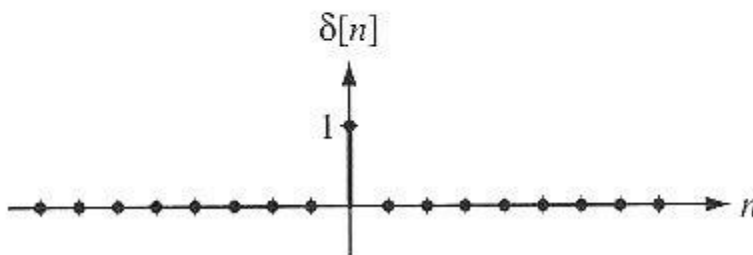
**SOFTWARE REQUIRED:**

1. MATLAB R2018a.

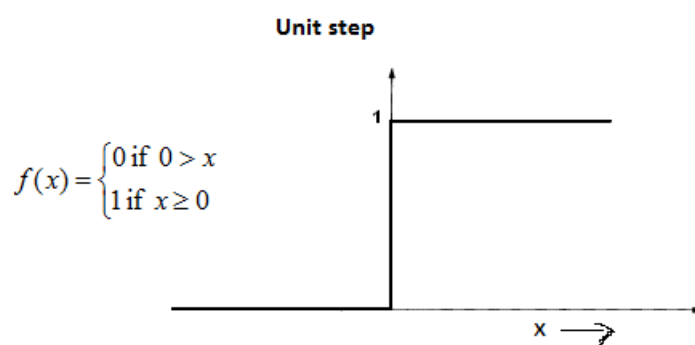
**THEORY:**

An ideal impulse function is a function that is zero everywhere but at the origin the amplitude is infinite.

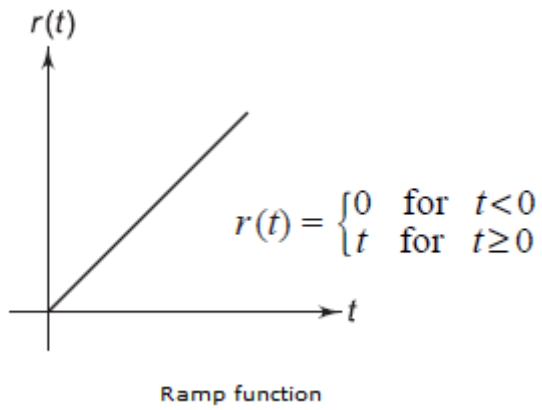
$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$



- A function that increases or decreases abruptly from one constant value to another is called as Step function.



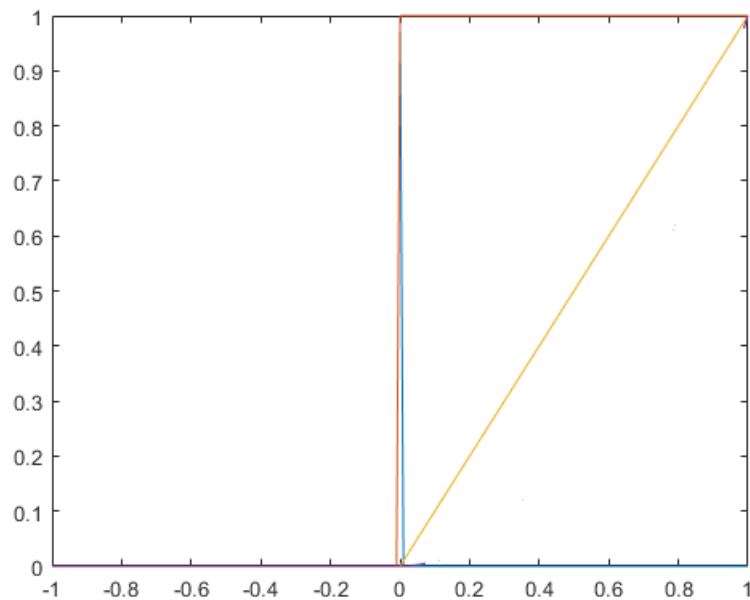
- The unit Ramp signal is denoted by r(t)
  - It is defined as  $r(t) = \{t, t \geq 0 \text{ and } 0, t < 0$
  - Area under unit ramp is unity.



**SCRIPT:**

```
t = (-1:0.01:1)';
impulse = t==0;
unitstep = t>=0;
ramp = t.*unitstep;
plot(t,[impulse unitstep ramp])
```

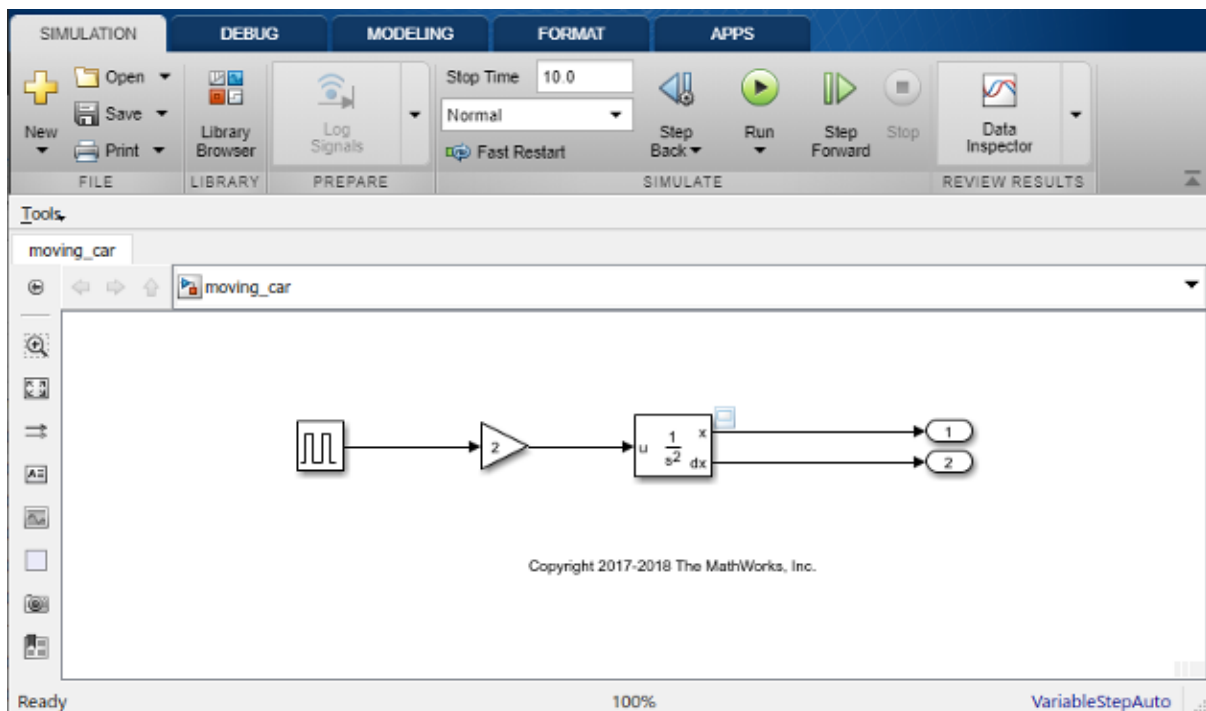
**SIMULATION RESULT:**



**Conclusion:**

# INTRODUCTION TO SIMULINK

Simulink is a block diagram environment for multi-domain simulation and Model-Based Design. It supports system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems. Simulink provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB, enabling you to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis.



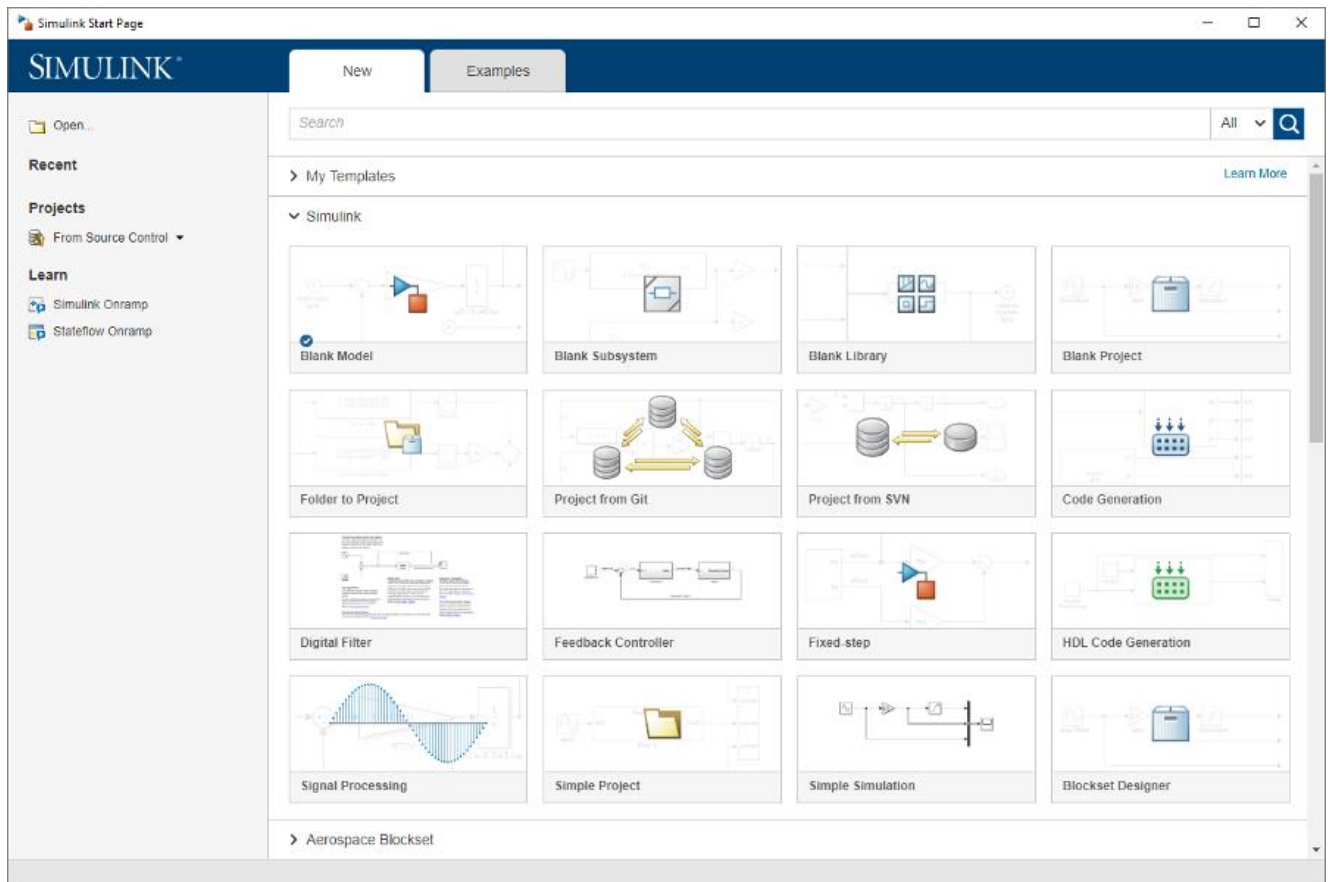
## PROCEDURE:

Step-1:

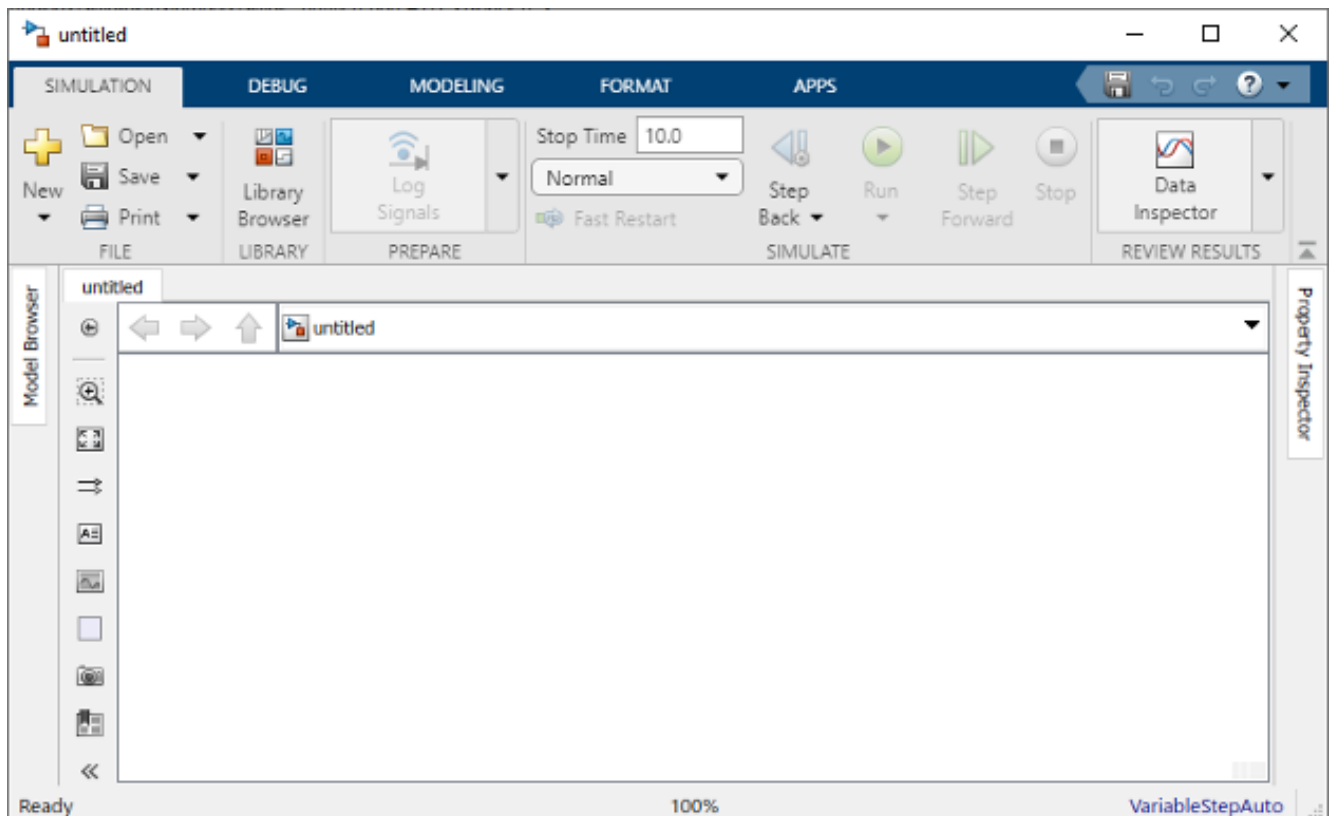
Start MATLAB from the MATLAB toolstrip, click the **Simulink** button  to open the Simulink editor.

Step-2:

Click the **Blank Model** template to open the Simulink editor.



Step: 3



From the Simulation tab, select Save > Save as. In the File name text box, enter a name for your model. For example, simple model. Click Save. The model is saved with the file extension .slx.

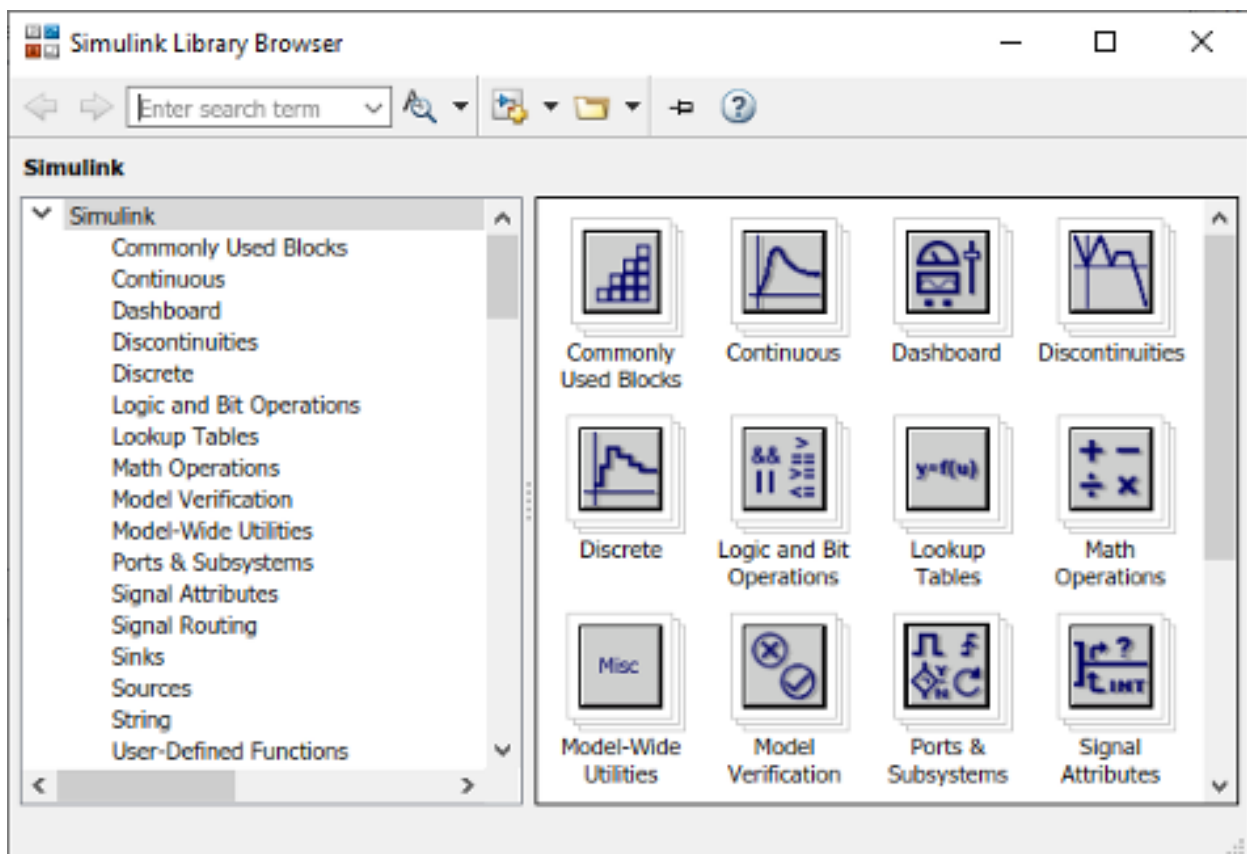
Step: 4


### Open Simulink Library Browser

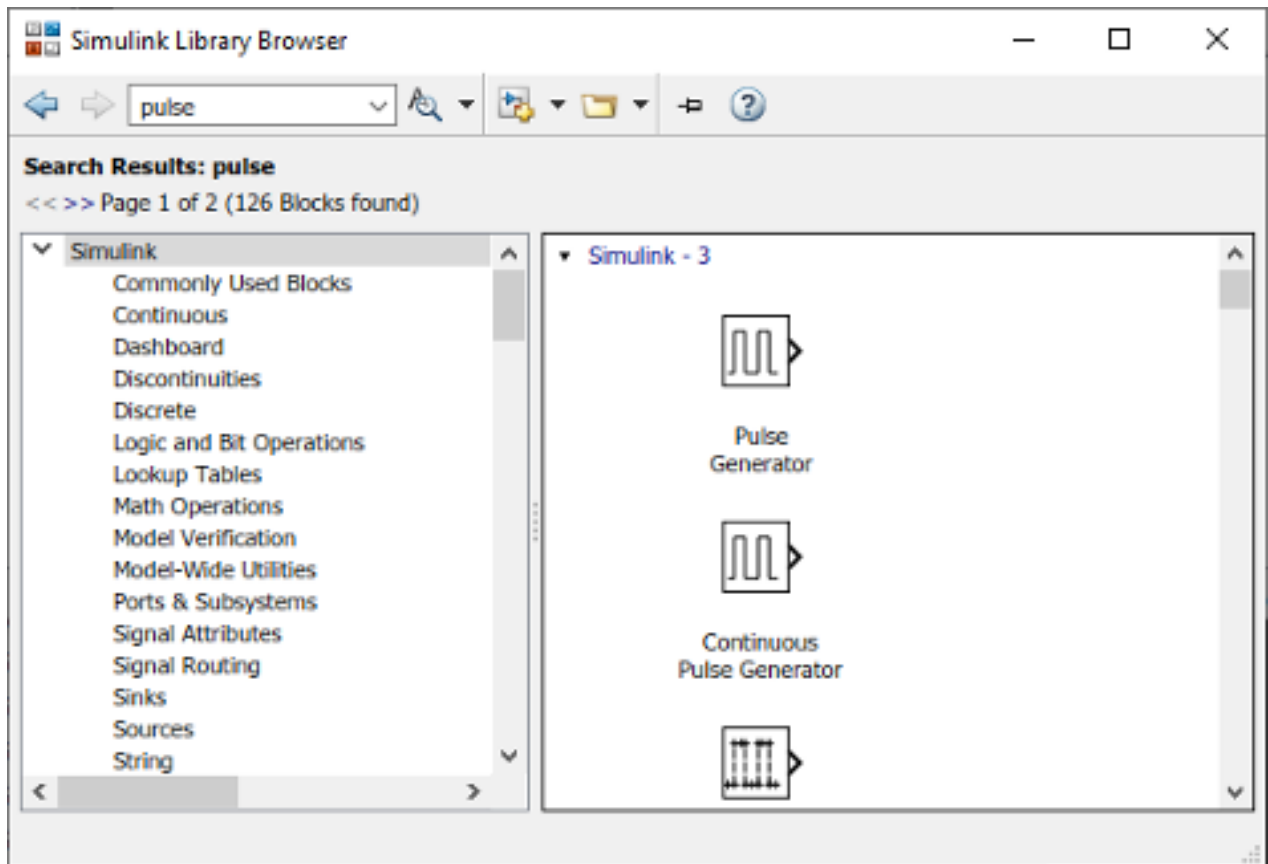
Simulink provides a set of block libraries, organized by functionality in the Library Browser. The following libraries are common to most workflows:

- Continuous — Blocks for systems with continuous states
- Discrete — Blocks for systems with discrete states
- Math Operations — Blocks that implement algebraic and logical equations
- Sinks — Blocks that store and show the signals that connect to them
- Sources — Blocks that generate the signal values that drive the model

From the **Simulation** tab, click the **Library Browser** button .



1. Set the Library Browser to stay on top of the other desktop windows. On the Simulink Library Browser toolbar, select the **Stay on top** button . To browse through the block libraries, select a category and then a functional area in the left pane. To search all of the available block libraries, enter a search term.



#### **Step: 4**

##### **Add Blocks to a Model**

To start building the model, browse the library and add the blocks.

#### **Step: 5**

##### **Connect Blocks**

Connect the blocks by creating lines between output ports and input ports.

#### **Step: 6**

##### **Add Signal Viewer**

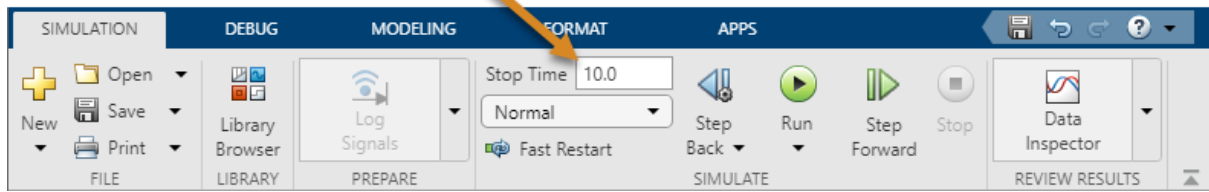
To view simulation results, connect the first output to a Signal Viewer.


#### **Step: 7**

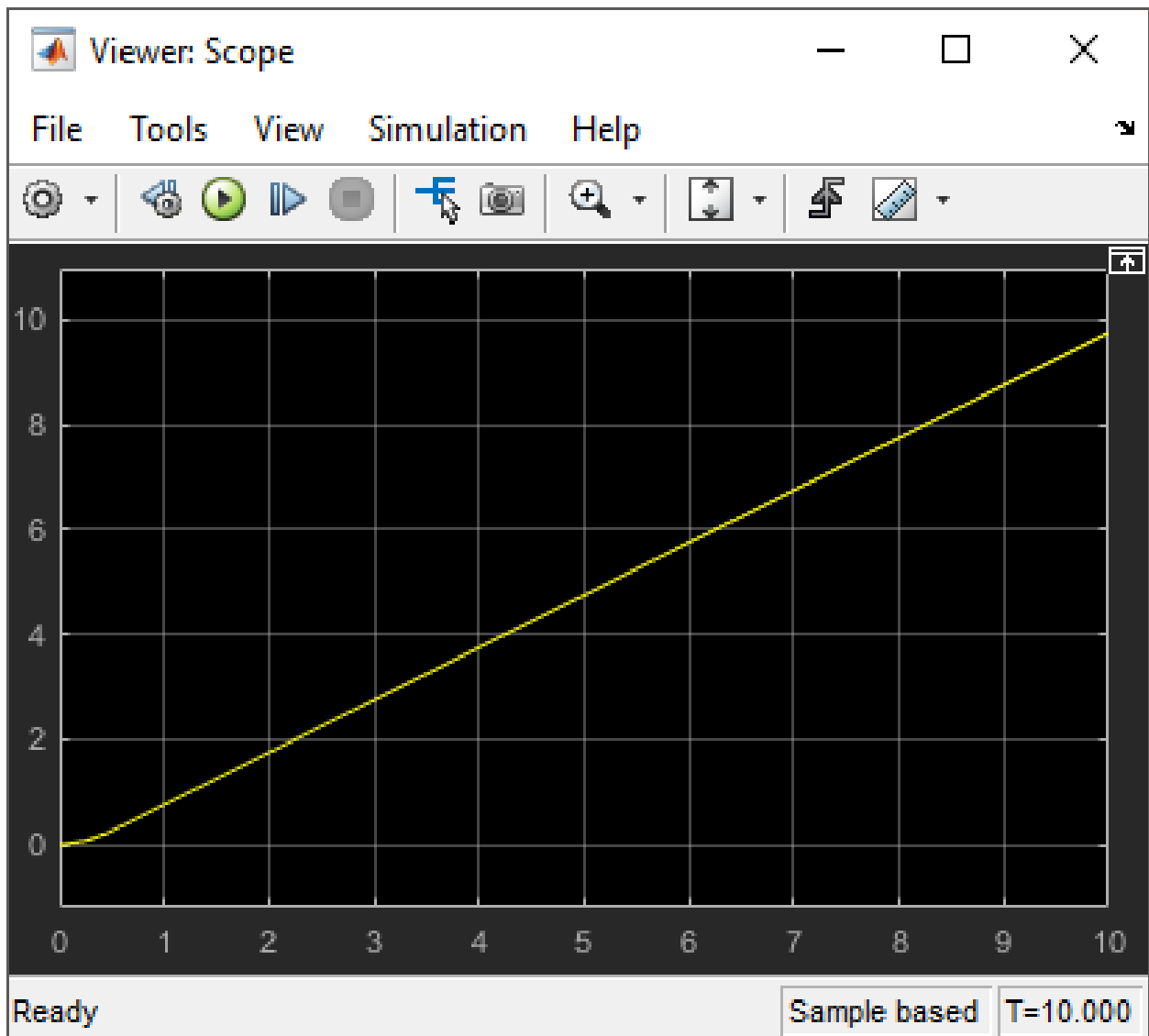
##### **Run Simulation**

After you define the configuration parameters, you are ready to simulate your model.

In the **Simulation** tab, set the simulation stop time by changing the value in the toolbar.



- The default stop time of 10.0 is appropriate for this model. This time value has no unit. The time unit in Simulink depends on how the equations are constructed. This example simulates the simplified motion of a car for 10 seconds — other models could have time units in milliseconds or years.
- To run the simulation, click the **Run** button .
- The simulation runs and produces the output in the viewer.



# EXPERIMENT NO.: 7(a)

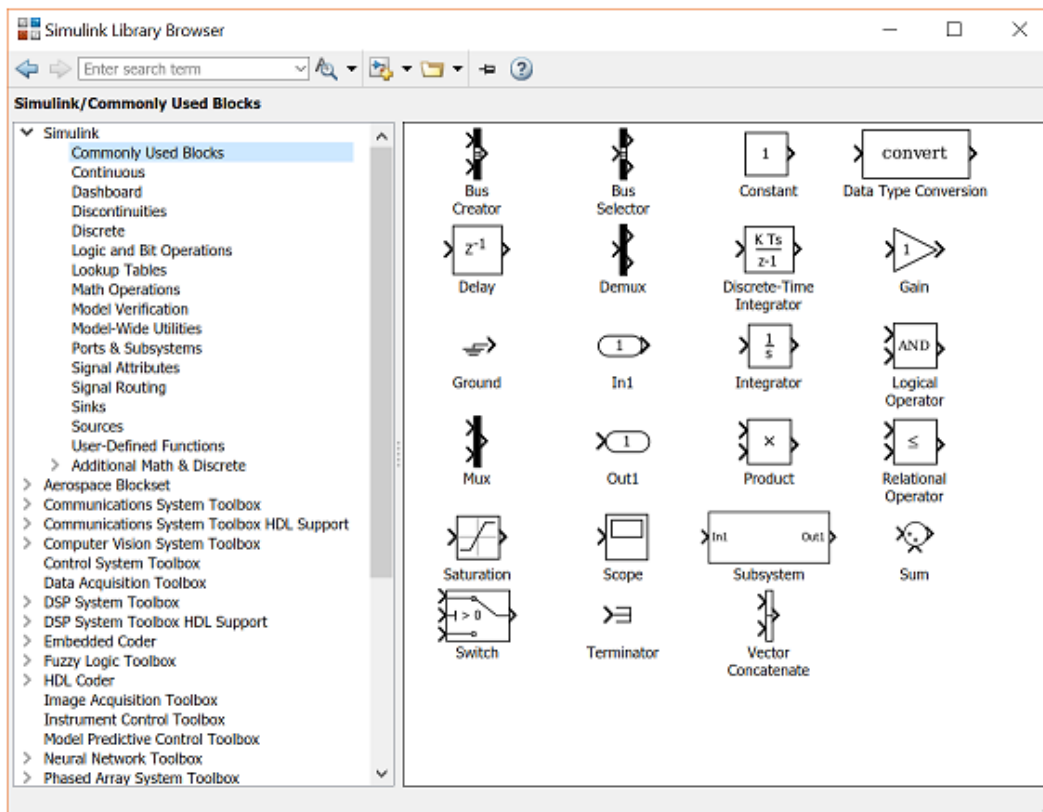
**AIM OF THE EXPERIMENT:** Study Commonly used blocks, Math operation block, Relational operator and Display block from SIMULINK library.

## SOFTWARE REQUIRED:

1. MATLAB R2018a.
2. Simulink Editor Toolbox

## THEORY:

**Commonly Used Blocks** are used to list a lot of blocks which are usually used. Double-click on the **Commonly Used Blocks** icon in the main Simulink window to bring up the Commonly Used window.



### Bus Creator

The Bus Creator block combines a set of signals into a bus.

### Bus Selector

The Bus Selector block outputs a specified subset of the elements of the bus at its input. The block can output the specified elements as separate signals or as a new bus.

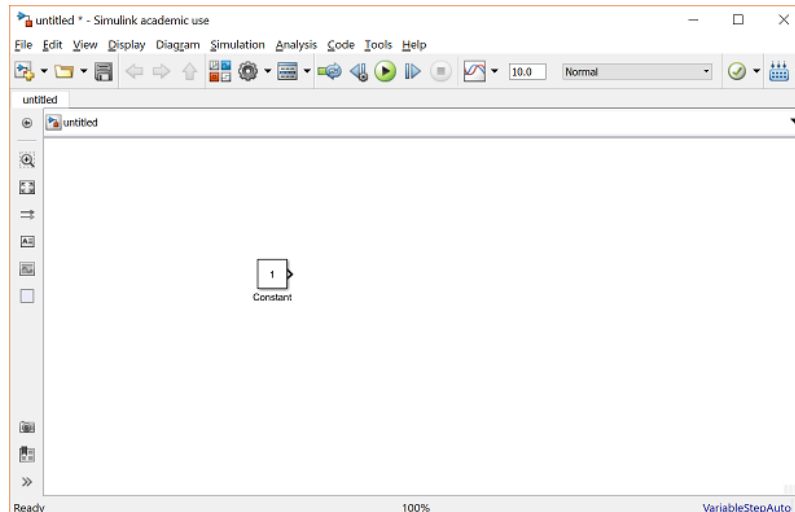


## Constant

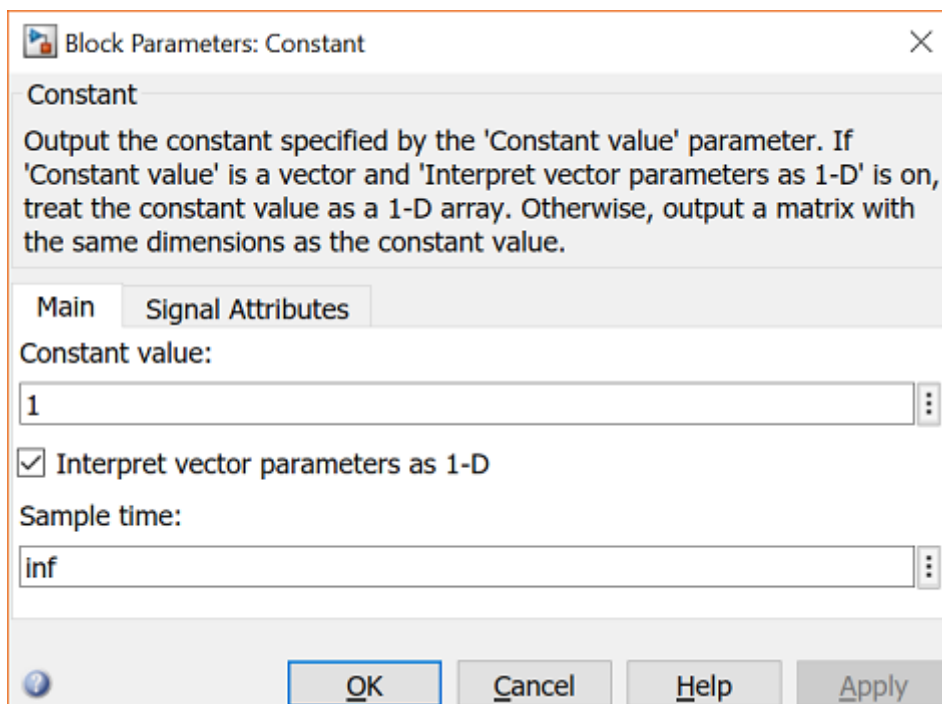
The **Constant** block generates a real or complex constant value. The constant output value is displayed in the middle of the block, with a default value of 1.

In order to examine these blocks, create a new model window (select **New** from the **File** menu in the Simulink window or hit **Ctrl+N**).

To use this block, drag it from the Commonly Used Blocks window into your new model window.



To change the constant output value, double-click on the block in your model window to bring up the following dialog box.



Change the constant value field from 1 to some other value, say, 5, and close the dialog box. Your model window will reflect the update by displaying a 5 in the middle of the constant block.

## **Data Type Conversion**

The Data Type Conversion block converts an input signal of any Simulink data type to the data type you specify for the Output data type parameter. The input can be any real- or complex-valued signal.

## **Delay**

The Delay block delays an input  $u$  according to the Delay length parameter, which you specify on the dialog box, or a delay length that a signal supplies to the input port. This block is equivalent to the  $z^{-1}$  discrete-time operator.

## **Demux, Mux**

The Mux (Multiplexer) block is used to combine two or more scalar signals into a single vector signal. Similarly, a Demux (Demultiplexer) block breaks a vector signal into scalar signal components. The number of vector components must be specified in each case.

## **Discrete-Time Integrator**

This is the discrete time approximation of a continuous-time integrator. The approximation method can be specified as well as the initial condition and saturation limits.

## **Gain**

The Gain block multiplies the input by a constant value (gain). The input and the gain can each be a scalar, vector, or matrix.

## **Ground**

The Ground block connects to blocks whose input ports do not connect to other blocks.

## **In1**

Inport blocks are the links from outside a system into the system.

## **Integrator**

The output of the Integrator is the integral of the input. An initial condition can be specified, as well as saturation limits.

## **Logical Operator**

The Logical Operator block performs the specified logical operation on its inputs. An input value is TRUE (1) if it is nonzero and FALSE (0) if it is zero.

## **Out1**

Outport blocks are the links from a system to a destination outside the system.

## **Product**

By default, the **Product** block outputs the result of multiplying two inputs: two scalars, a scalar and a nonscalar, or two nonscalars that have the same dimensions.

## **Relational Operator**

By default, the **Relational Operator** block compares two inputs using the **Relational operator** parameter that you specify. The first input corresponds to the top input port and the second input to the bottom input port.

### **Saturation**

The Saturation block imposes upper and lower limits on an input signal.

### **Scope**

The Scope block displays inputs signals with respect to simulation time.

### **Subsystem**

A Subsystem block represents a subsystem of the system that contains it. The **Subsystem** block can represent a virtual subsystem or a nonvirtual subsystem.

### **Sum**

The Sum block performs addition or subtraction on its inputs. This block can add or subtract scalar, vector, or matrix inputs. It can also collapse the elements of a signal.

### **Switch**

The Switch block passes through the first input or the third input based on the value of the second input. The first and third inputs are called data inputs. The second input is called the control input.

### **Terminator**

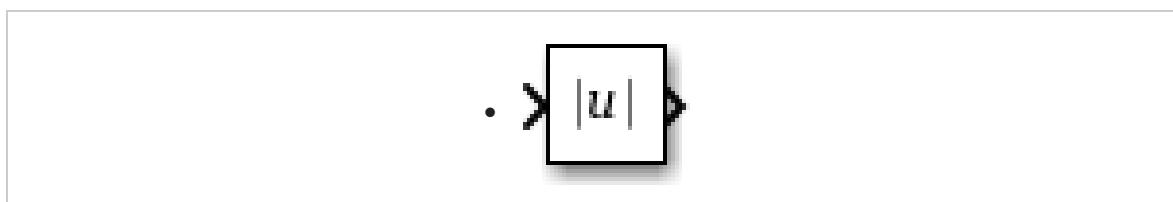
Use the Terminator block to cap blocks whose output ports do not connect to other blocks.

### **Vector Concatenate**

The Concatenate block concatenates the signals at its inputs to create an output signal whose elements reside in contiguous locations in memory.

### **MATH OPERATOR BLOCK**

1. Output absolute value of input



2. Add or Subtract the inputs

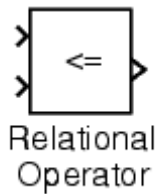


### 3. Multiply or Divide



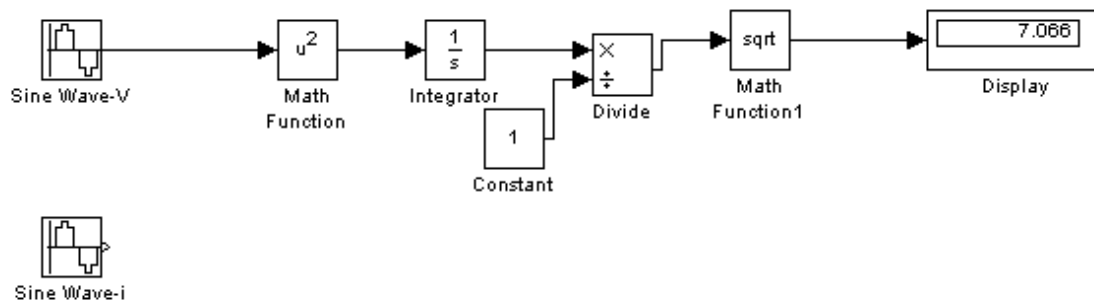
### Relational Operator:

By default, the **Relational Operator** block compares two inputs using the **Relational operator** parameter that you specify. The first input corresponds to the top input port and the second input to the bottom input port.

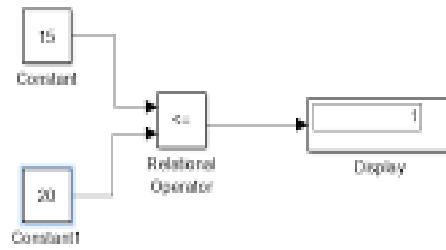


### SIMULATION:

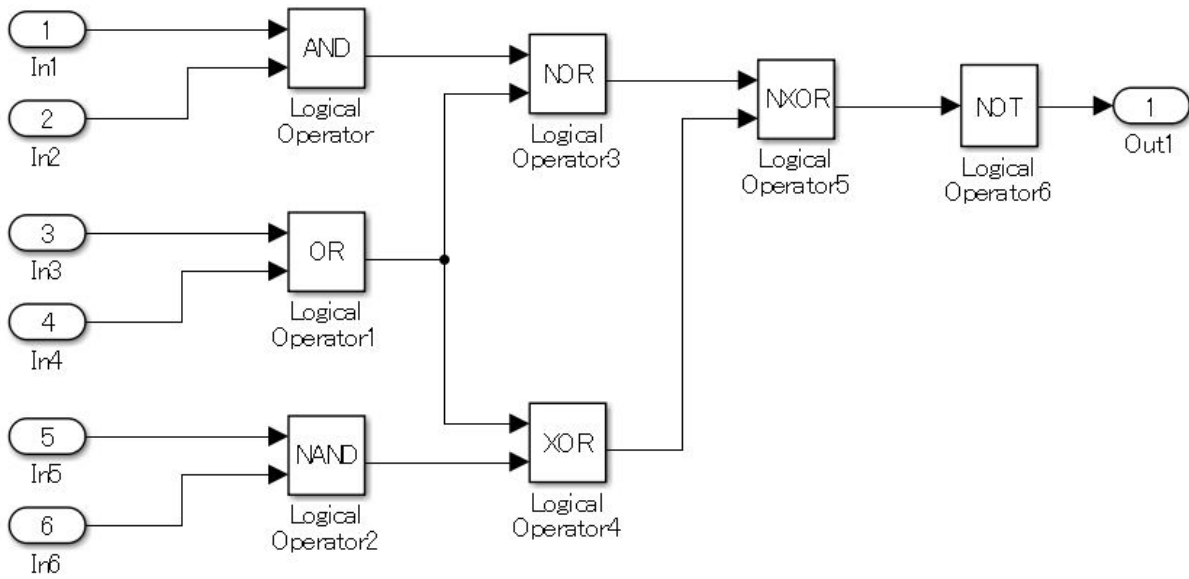
### Mathematical Operation:



### Relational Operation



## Logical Operation



**CONCLUSION:**

# EXPERIMENT NO.: 8

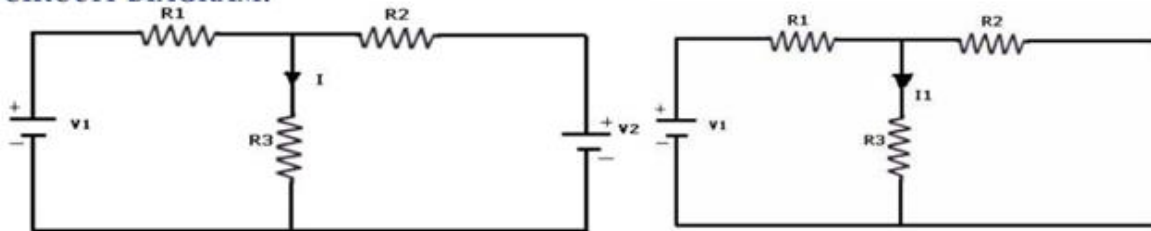
**AIM OF THE EXPERIMENT:** Design & Verify Superposition theorem using Simulink Editor toolbox.

**SOFTWARE REQUIRED:**

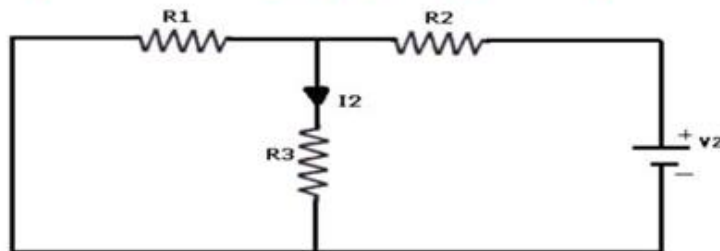
1. MATLAB R2018a.
2. Simulink Editor

**THEORY:**

**CIRCUIT DIAGRAM:**

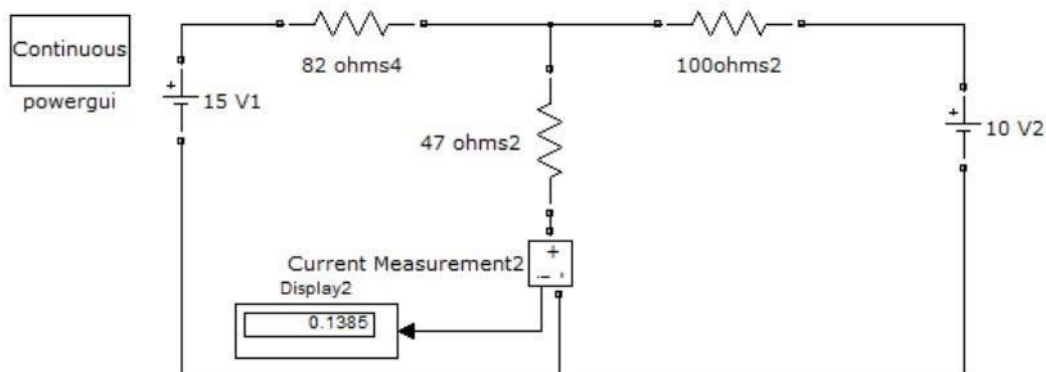


**Fig- 5.1 Both Voltage Sources are acting ( $V_1$ & $V_2$ )      Fig - 5.2 Voltage Source  $V_1$  is acting alone**

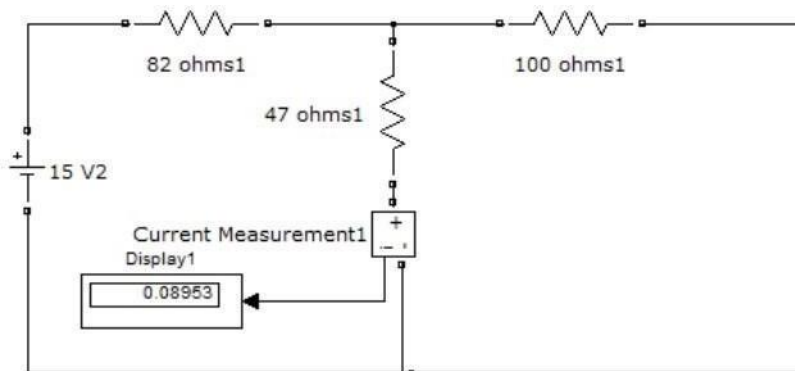


**Fig - 5.3 Voltage Source  $V_2$  is acting alone**

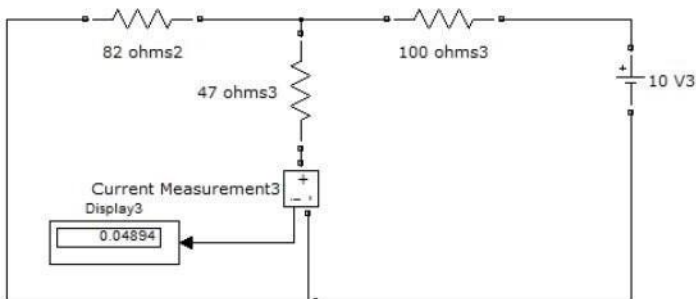
**SIMULATION:**



**Fig.1: Circuit diagram when  $V_1 \neq 0$  &  $V_2 \neq 0$**



**Fig.2: Circuit diagram when  $V_1 \neq 0$  &  $V_2 = 0$**



**Fig.3: Circuit diagram when  $V_1 = 0$  &  $V_2 \neq 0$**

**OBSERVATION:**

<b>PARAMETERS</b>	<b>WHEN BOTH <math>V_1 \neq 0</math> &amp; <math>V_2 \neq 0</math> (I)</b>	<b>WHEN <math>V_1 \neq 0</math> &amp; <math>V_2 = 0</math> (I<sub>1</sub>)</b>	<b>WHEN <math>V_1 = 0</math> &amp; <math>V_2 \neq 0</math> (I<sub>2</sub>)</b>
Current through R <sub>3</sub> (Theoretical Values)			
Current through R <sub>3</sub> (Practical Values)			

**CONCLUSION:**

# EXPERIMENT NO.: 9

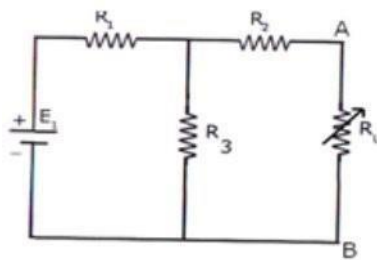
**AIM OF THE EXPERIMENT:** Design & Verify Thevenin's' theorem using Simulink Editor toolbox.

**SOFTWARE REQUIRED:**

1. MATLAB R2018a.
2. Simulink Editor

**THEORY:**

**Thevenin's Theorem:** Any linear, bilateral network having a number of voltage, current sources and resistances can be replaced by a simple equivalent circuit consisting of a single voltage source in series with a resistance, where the value of the voltage source is equal to the open circuit voltage and the resistance is the equivalent resistance measured between the open circuit terminals with all energy sources replaced by their ideal internal resistances.



8.1 CIRCUIT DIAGRAM:

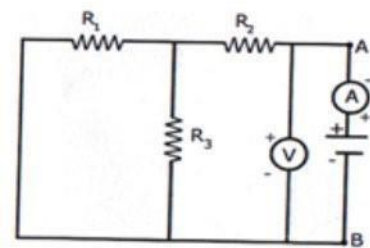


Fig-8.1 Measurement of  $V_{TH}$  or  $V_{OC}$

Fig – 8.2 Measurement of  $R_{TH}$

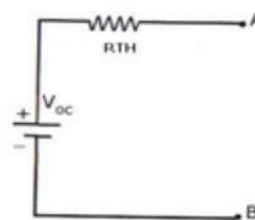
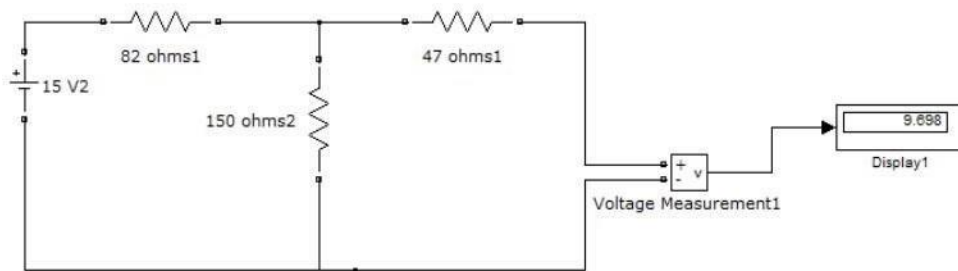


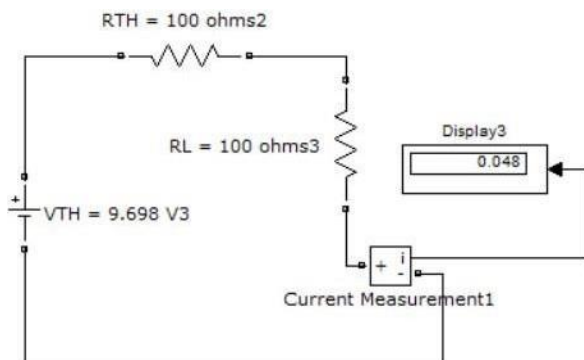
Fig – 8.3 Measurement of  $I_L$  ( $I_L = V_{TH}$  or  $V_{OC} / R_{TH} + R_L$ )

**SIMULATION:**





**Fig.1: Measurement of V**



**Fig.1: Measurement of  $I_L$**

**OBSERVATION:**

Parameters	Theoretical Values	Practical Values
$I_L$		

**CONCLUSION:**

# EXPERIMENT NO.: 10

**AIM OF THE EXPERIMENT:** Design and simulate uncontrolled half wave rectifier by using Simulink Editor.

## SOFTWARE REQUIRED:

3. MATLAB R2018a.
4. Simulink Editor

## THEORY:

- Half-wave rectifiers only allow one half-cycle (positive or negative half-cycle) of the AC voltage through and will block the other half-cycle on the DC side, as seen below.
- But the diode is only part of it – a complete half-wave rectifier circuit consists of 3 main parts:
  - A transformer
  - A resistive load
  - A diode

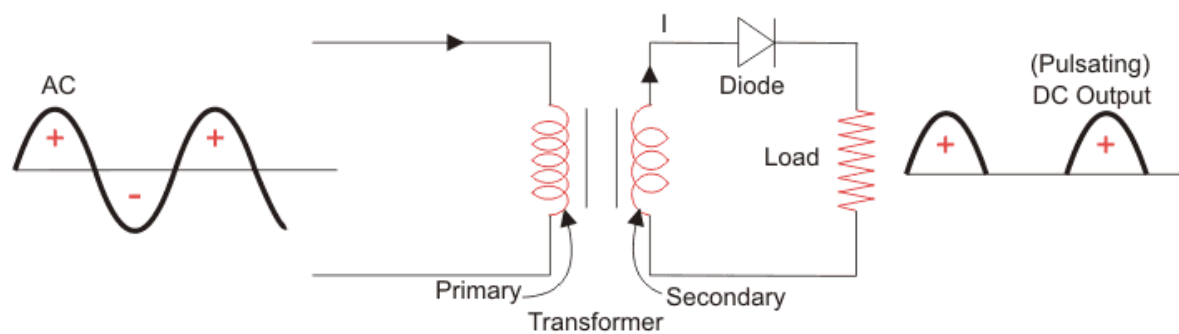
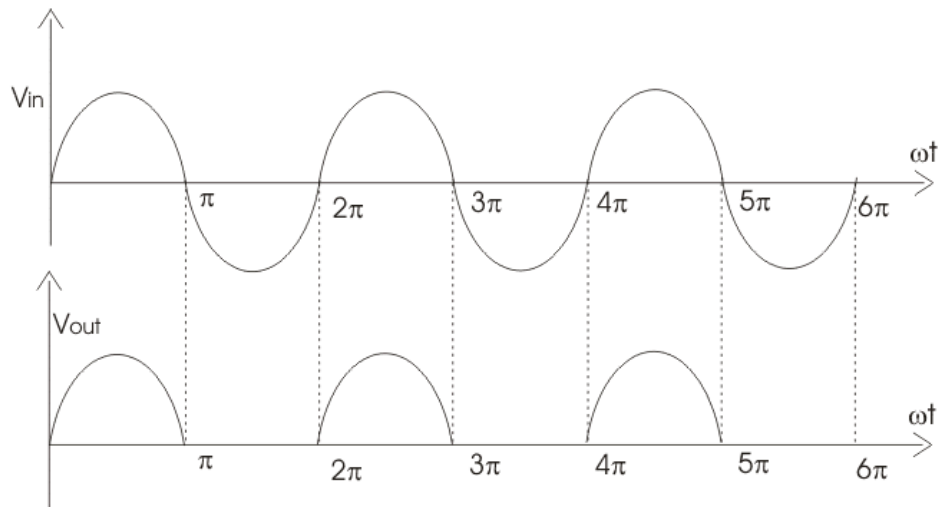


Figure - 3

- AC voltage is applied to the primary side of the step-down transformer and we will get a low voltage at the secondary winding which will be applied to the diode.
- During the positive half cycle of the AC voltage, the diode will be forward biased and the current flows through the diode. During the negative half cycle of the AC voltage, the diode will be reverse biased and the flow of

current will be blocked. The final output voltage waveform on the secondary side (DC) is shown in the below figure.

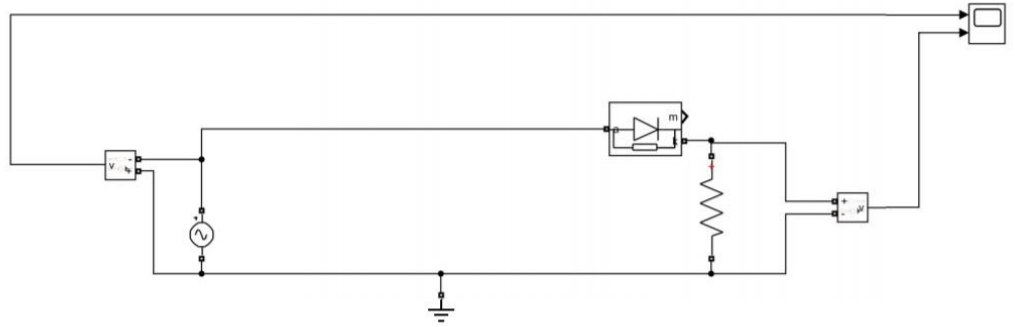


### PROCEDURE:

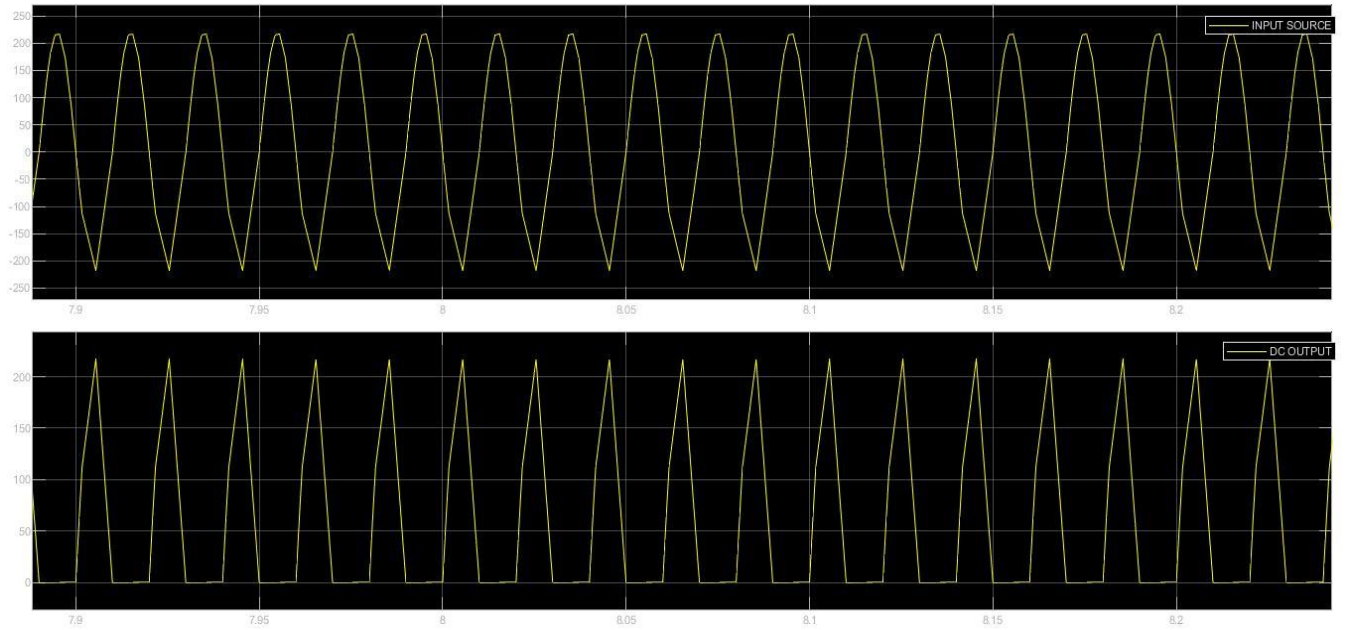
- Open MATLAB and then open Simulink using the simulink icon on MATLAB.
- Create a blank model.
- Click on the library browser icon on Simulink recently created model.
- From the library browsers, search all the components required to design the full wave rectifier circuit and add them to the model.

## SIMULATION:

Continuous  
powergui



## OUTPUT:



## CONCLUSION:

# EXPERIMENT NO.: 11

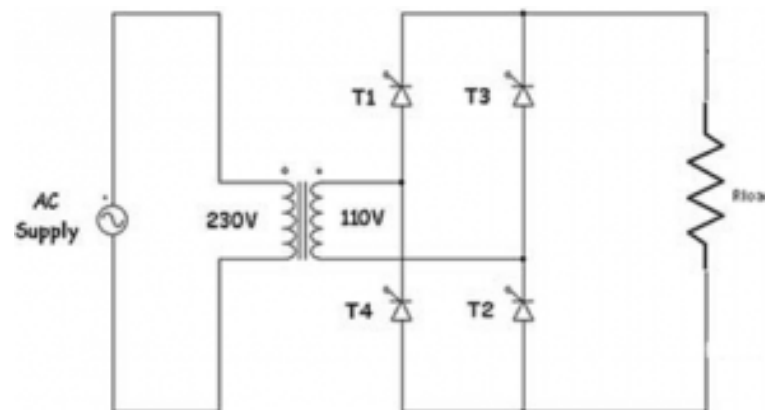
**AIM OF THE EXPERIMENT:** Design and simulate 1- $\phi$  controlled full wave rectifier by using Simulink Editor.

## SOFTWARE REQUIRED:

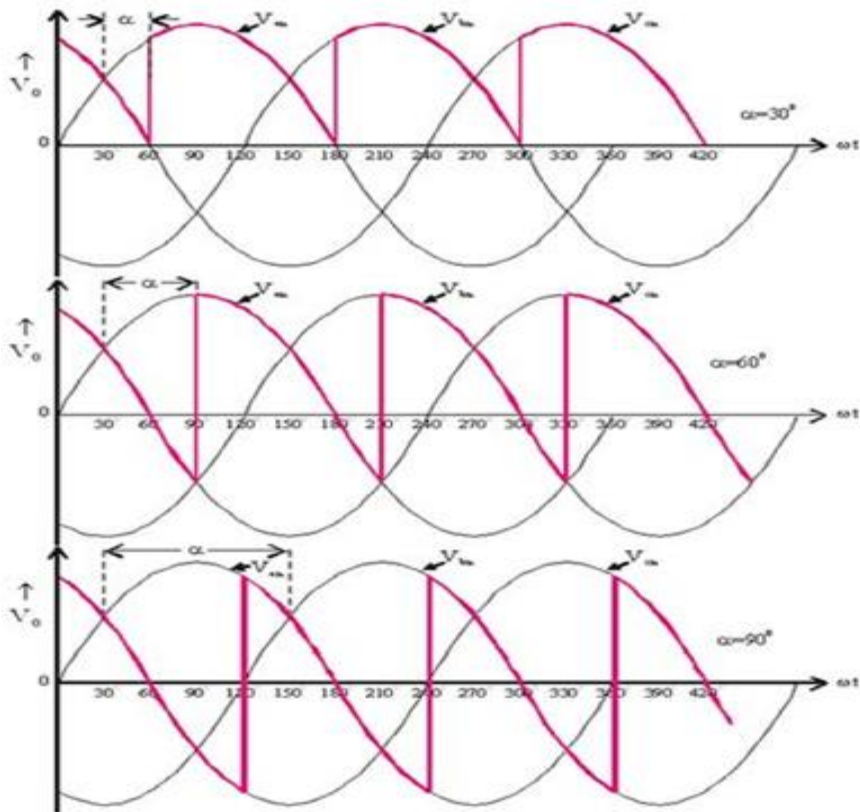
1. MATLAB R2018a.
2. Simulink Editor

## THEORY:

- A full wave rectifier however uses both the positive and negative parts of the AC wave to rectify. Again in this case we use switches with gate pulses to be used as the firing angles of the rectified output.
- All four devices used are thyristors. The turn-on instants of these devices are dependent on the firing signals that are given. Turn-off happens when the current through the device reaches zero and it is reverse biased at least for duration equal to the turn-off time of the device specified in the data sheet.

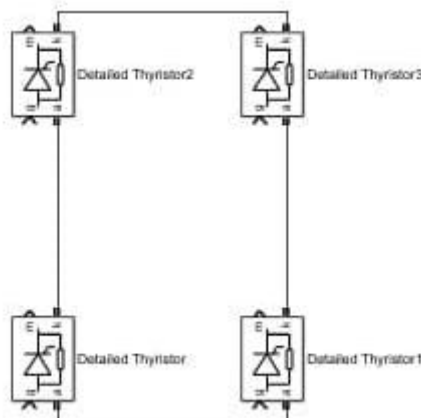


- In positive half cycle thyristors T1 & T2 are fired at an angle  $\alpha$  .  
When T1 & T2 conducts  $V_o = V_s$   
 $I_o = V_o/R = V_s/R$
- In negative half cycle of input voltage, SCR's T3 & T4 are triggered at an angle of  $(\pi + \alpha)$ . Here output current & supply current are in opposite direction. T3 & T4 becomes off at  $2\pi$ .



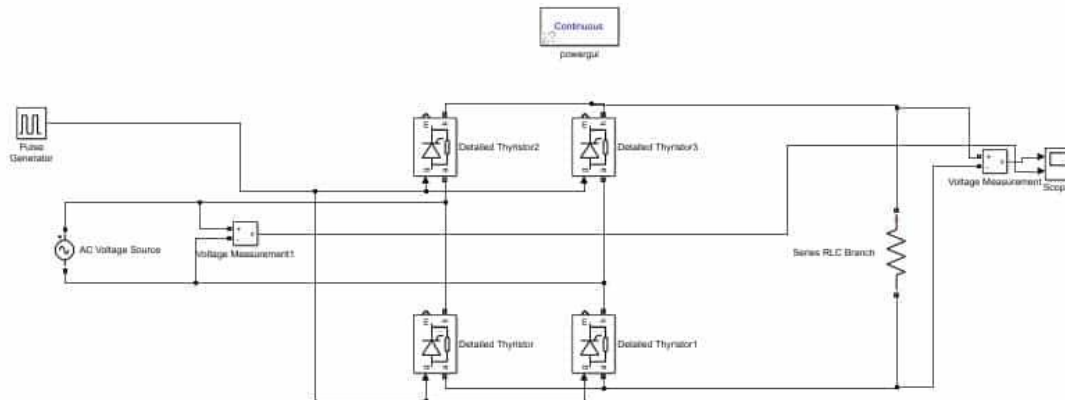
## PROCEDURE:

- Open MATLAB and then open Simulink using the simulink icon on MATLAB.
- Create a blank model.
- Click on the library browser icon on Simulink recently created model.
- From the library browsers, search all the components required to design the full wave rectifier circuit and add them to the model. The required components are: Pulse generator (1 no.), Detailed Thyristors (4 no.s), voltage measurement (2 no.s), series RLC load (1 no.), scope (1 no.), AC voltage source (1 no.), powegui (1 no.).
- Connect all the 4 thyristors in bridge configuration as shown in the figure.



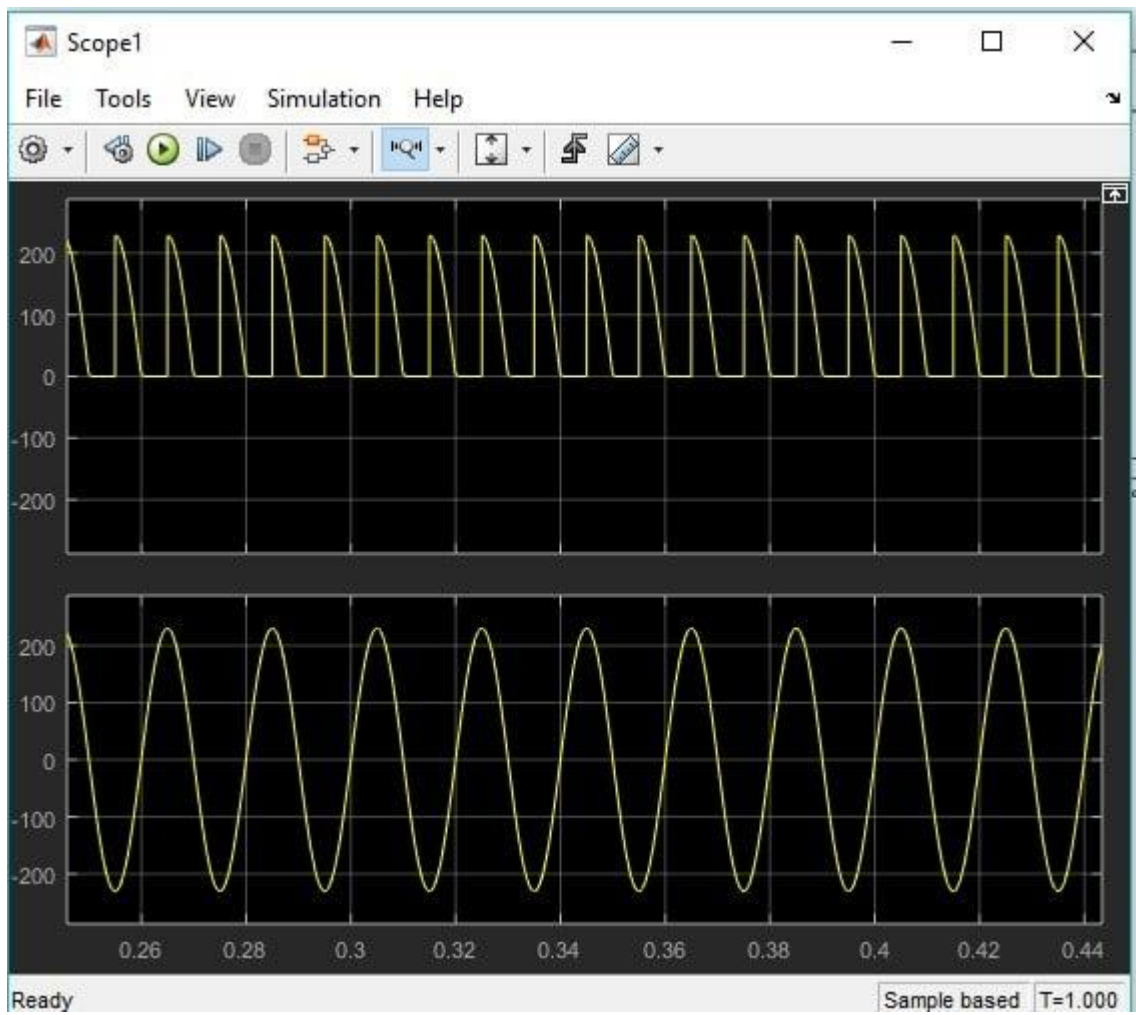
- Pulse generator is connected to the input pin **g** of the thyristors.
- Connect the positive side of the AC source between the two serially connected diodes and the negative side to other serially connected diodes as shown in the figure below.
- After the source is connected we also need a ground to complete the circuit.
- Double click on the RLC branch and from the parameters block select the resistor branch only and connect it as load in the circuit.
- powergui is a block which is the essentially required in Simulink to run an electronic circuit.
- Connect one of the voltage measurement block across the load resistor as are basically interested in measuring and viewing the voltage waveform across the load resistor.
- Another voltage measurement block is connected across the input source.
- These two voltage measurement block outputs are connected at the input ports of oscilloscope to visualize the waveform.
- Set the parameters of pulse generator.
- Run the model by pressing the run button.
- Double click on the scope to view the waveform.

## SIMULATION:



**Fig: Full wave bridge rectifier circuit designed in Simulink editor.**

## RESULT:



**Figure: Input and Output waveforms of full wave bridge rectifier.**

**CONCLUSION:**



# EXPERIMENT NO.: 12

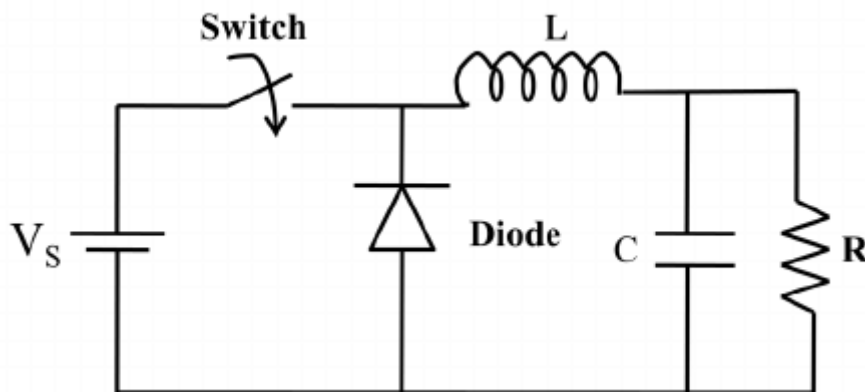
**AIM OF THE EXPERIMENT:** Design and simulate step-down chopper by using Simulink Editor.

**SOFTWARE REQUIRED:**

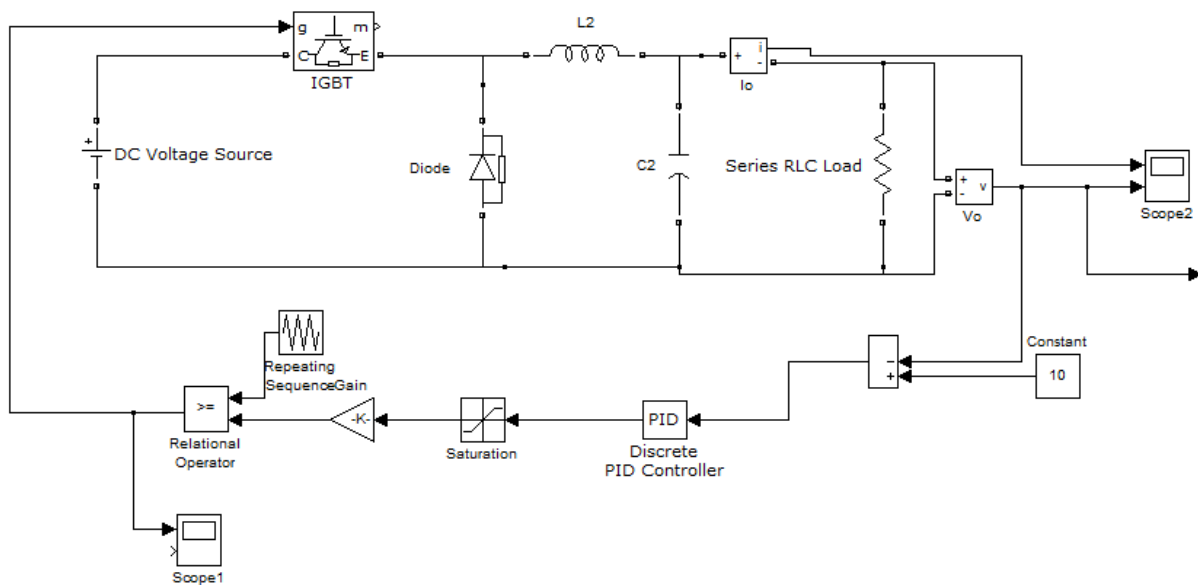
1. MATLAB R2018a.
2. Simulink Editor

**THEORY:**

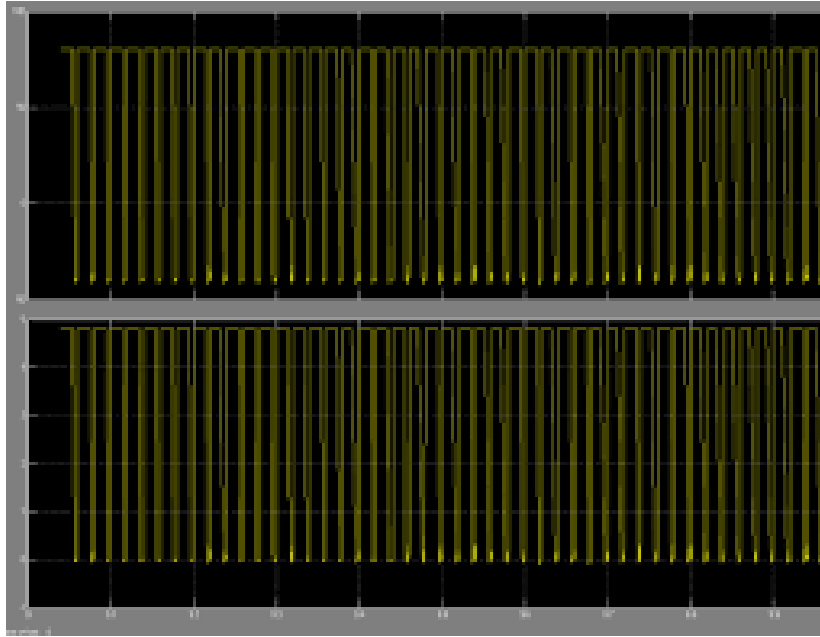
A Step-down chopper is a static device that step downs its DC input voltage. The value of average output DC voltage of this chopper is less than that of its fixed DC input source voltage. This type of chopper is more common.



**SIMULATION:**



**OUTPUT:**



**CONCLUSION:**