INDEX

Sr No	Experiment	Date of submission	Marks obtained	Sign
1	Basic computations using Python programming.			
2	Introduction to Arduino Embedded platform.			
3	Introduction to MATLAB programming and SIMULINK			
4	Design of smart meter for recording the electricity consumption			
5	Design of Ultrasonic proximity sensors using Arduino			
6	3D printing of Airfoil through rapid prototyping 3D printer			
7	Dynamic simulation of drone (unmanned air vehicle) through MATLAB/SIMULINK			
8	Design of line follower autonomous vehicle.			
9	ANSYS simulation of bending of a beam in an earthquake resist-building			
10	Use simulations to understand the performance/behavior of a system, generating or loading data from sources, and testing the system.			

Experiment No:-03
Introduction to MATLAB programming and SIMULINK

Aim: To understand the basics about MATLAB software and learn basic programming and simulation.

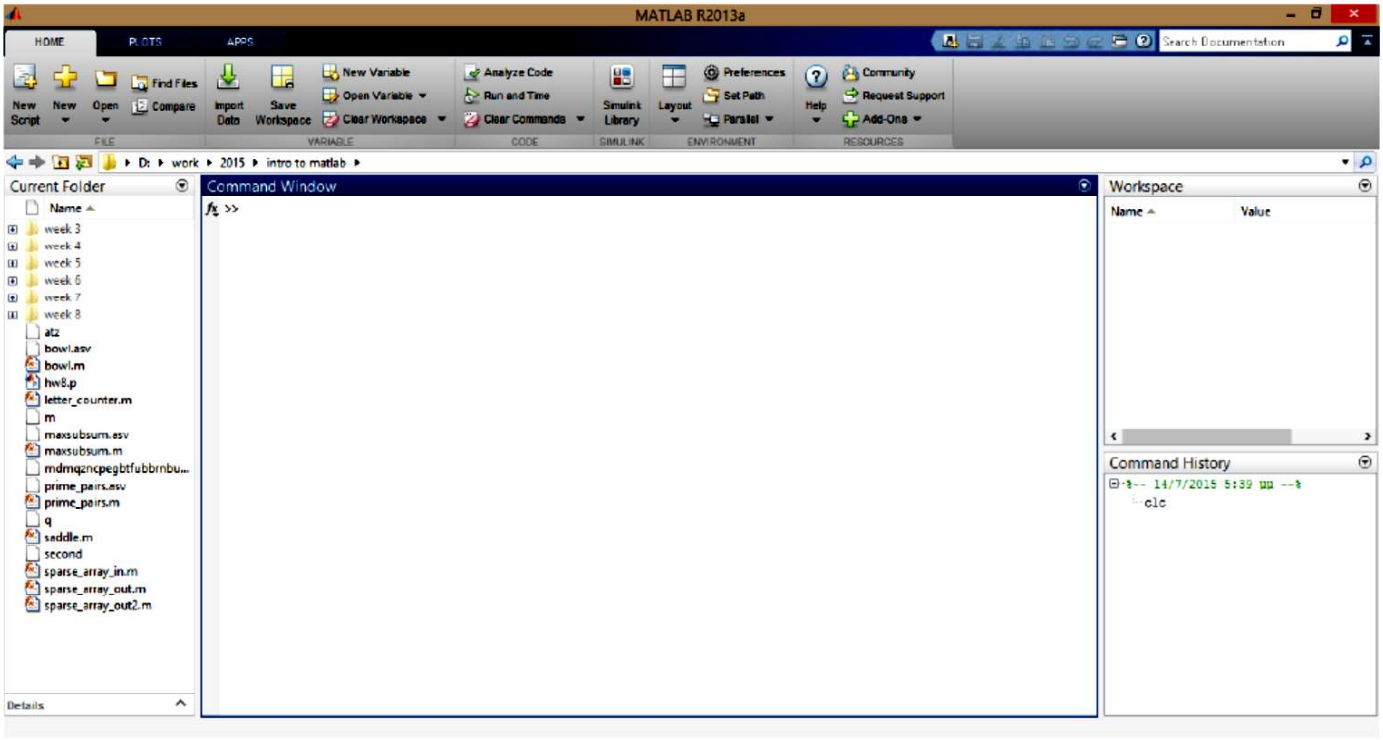
❖ Introduction

"MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation." - from Mathworks MATLAB is available on all three major operation systems.

SIMULINK is a powerful simulator for modeling and simulating dynamic systems.

One of the biggest advantages of using MATLAB is one does not need to declare ahead of time, the type of variable being used. The base version of MATLAB/SIMULINK is aptly supported by a variety of toolboxes.

The window is divided into three main parts.



- The Command Window is the main window where the commands are input.
- The Current Directory shows the directory from which MatLab runs the files and functions we have created.
- Command History shows the history of all the commands that we input into the Command Window.
- The main menu has three different tabs, HOME, PLOTS and APPS. From the Home tab the appearance and layout of MatLab can be changed.

- **Script File**

A script file in MATLAB is any set of MATLAB commands which are executed in that sequence. Script files have ".m" extension. It is recommended to use a meaningful name for the filename without including "space".
A script file can be run by simply typing the filename without its extension.
The scope of the variables used in script files is the general workspace. They can be accessed even after the execution.
A script file can contain functions which can be used within that script.
MATLAB executes the script file by interpreting every line, which makes its somewhat slow at times.
A script file can be put on path so that it can be called from any directory.

Note: Do not assign a script file name which is already a MATLAB in built function.

- **Matrix Algebra**

Create a row vector A = [1 3 6 8]
Create a column vector A = [1;3;6;8]
Transpose of matrix A = [1 3 6 8]' **Or use transpose(**A)
Create a matrix A = [1 3; 6 8]
Assign each element to matrix A(1, 1) = 1; A(2, 1) = 6; A(1, 2) = 3; A(2, 2) = 8;

- Both matrices and vectors are enclosed in square brackets
- Elements are accessed using parentheses

- **For creating uniformly spaced vector** B = (0 : 1 : 99)

B has 100 elements from 0 to 99 with spacing of 1.

Access first two and last two elements of B: B=([1 2 end-1 end])

Take every third element of B and store in C: C=B(1:3:end)

- **Special Matrices/Array**

Matlab allows to create a variety of special matrices that appear in several problems.

- **Create an Identity matrix** Imat = eye(3; 3)
- **Create a matrix of ones and zeros** A = ones(3; 3) B = zeros(3; 3)
- **Hadamard Matrix** H = hadamard(2)
- **Magic Matrix** M = magic(3)
- **Toeplitz Matrix** T = toeplitz([34]; [32])
- **Simple checks on matrices** Check if the matrix is empty
  A=[ ]; isempty(A)
- **Check if two matrices are equal**
  A=exp([1 2 ; 3 4])
  B=expm([1 2; 3 4])
  isequal(A,B)
- **Check if a matrix contains real elements**
  A = [1 2; 3 4]
  B = ones(2; 2)

C = A + j * B
isreal(C)

- **Check if matrix elements are NaN (Not a Number)**
  A = [1 2; 3 inf]
  isnan(A)
- **Mathematical Operations on Matrices**
  Transpose of a matrix A = [1 8 3; 5 6 0] A'
- **Extract triangular part**
  tril(A)
- **Find indices of elements**
  find(A) find(A >= 4)
- **Matrix Multiplication**
  A = [1 8; -5 0; 9 2] B = [2 6; 8 3; -1 5]
- **Element wise multiplication:** A.*B
- **Product of A and B:** A*B
- **Maximum and minimum of A**
  max(A) min(abs(A))
- **Inverse of matrix A:** inv(A)
- **Determinant of matrix A:** det(A)
- **Matrix division** B
  B *inv(A)
- **Element wise division:** B./A
- **Eigen values of matrix A:** eig(A)
- **Rank of matrix A:** rank(A)

- **Characteristics equation**
  eqA = poly([1 2; 3 4])
  roots(eqA),
  Note: compare this with eigen values of A
- **LU factorization** [L,U] = lu([1 2; 3 4])
- **Orthogonalization** Q = orth([1 2; 3 4])
- **Special Numbers and variables**
  pi: The value of pi
  inf: Infinity (or a very very large number)
  eps: Floating point relative accuracy
  i or j: Imaginary number,
  NaN: Not-a-Number
  ans: The most recent answer
  end: The last element of a vector; OR to indicate end of a loop or a conditional statement
  all: Used with clear command to clear all variables

- **Elementary functions**
  Trigonometric: sin, sinh, cos, atan, sec
  Exponential: exp, log, log2, pow2, sqrt
  Complex: abs, imag, conj, unwrap, angle
  Rounding: fix, floor, ceil, mod, rem, sign
  Specialized: bessel, beta, erf, dot, gamma
  Number theoretic: factor, primes, factorial, gcd

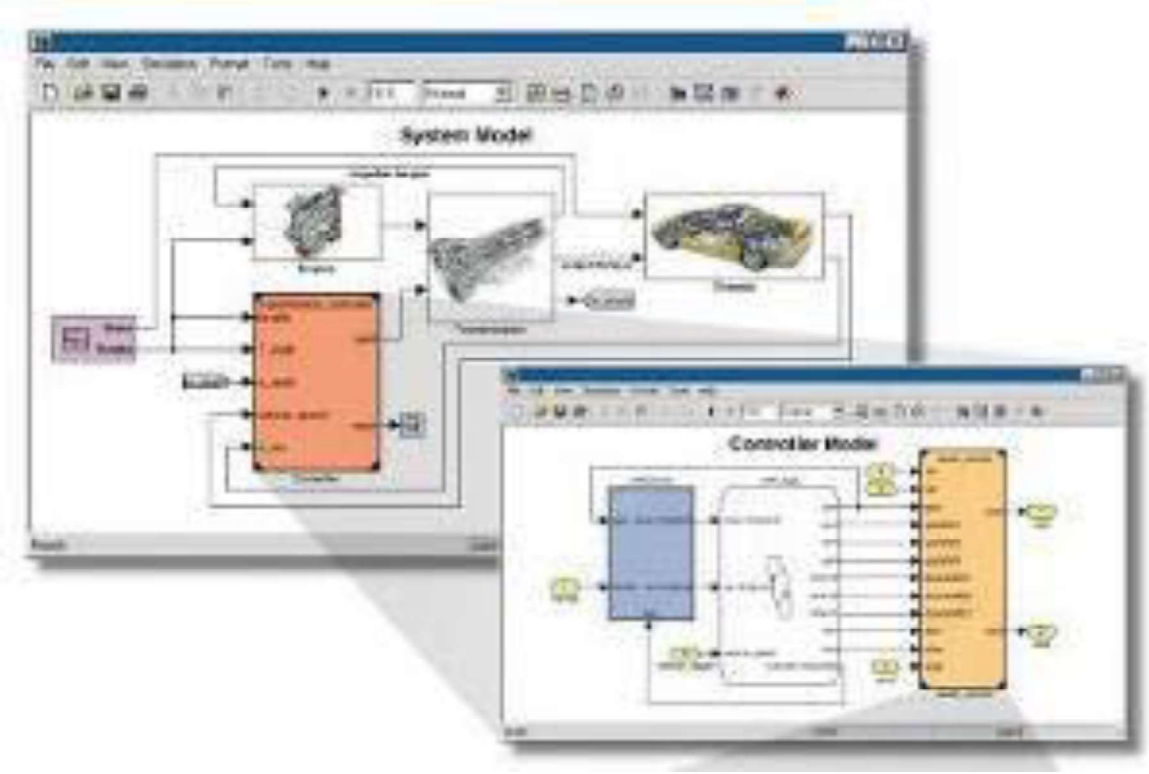
❖ **Create 100 samples of sine and cosine**

x = sin(2 *pi * 0:2 * (0 : 99)) ; y = cos(2* pi * 0:2 *(0 : 99))

Simulink

Simulink is a software package for modeling, simulating, and analyzing dynamical systems

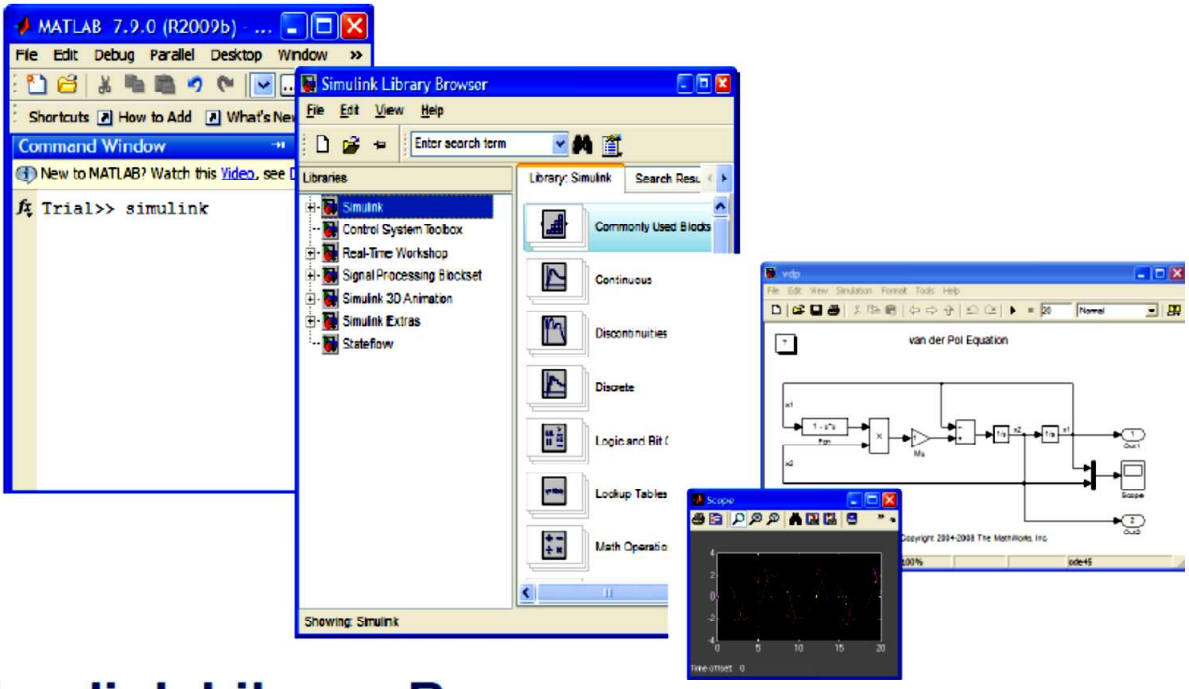
• Block diagram editing
• Nonlinear simulation
• Hybrid (continuous and discrete) models
• Asynchronous (non-uniform sampling) simulation
• Fully integrated with MATLAB, MATLAB toolboxes and blocksets.



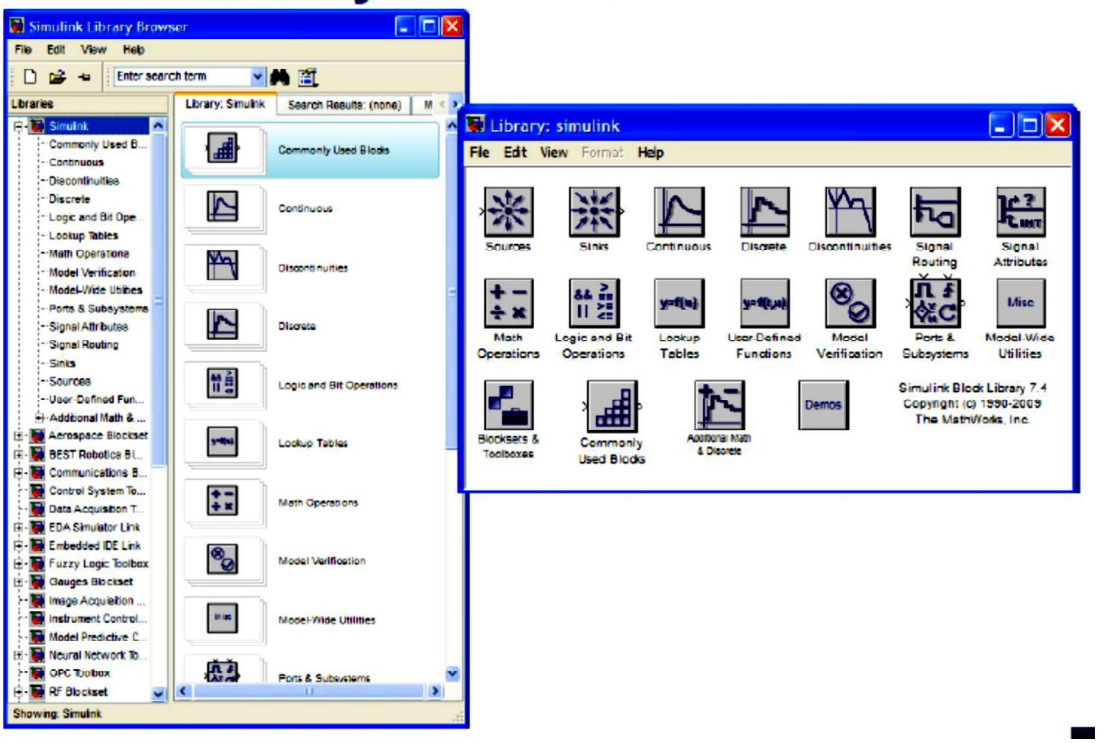
Simulink accurately designs, implements, and tests:

- Control systems
- Signal Processing systems
- Communications systems
- Embedded systems
- Physical systems
- other Dynamical systems

Launching Simulink



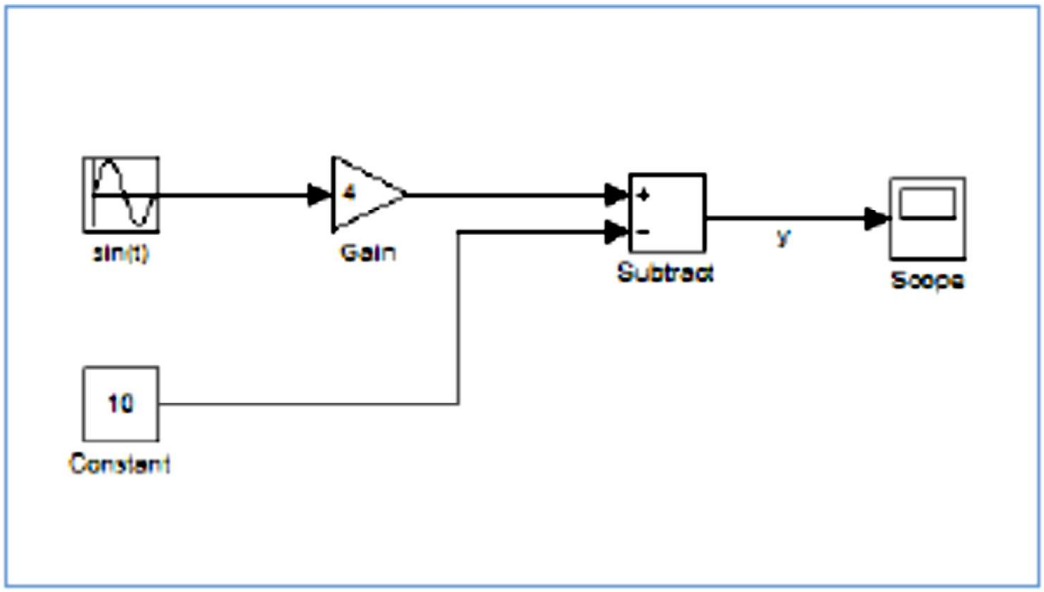
Simulink Library Browser



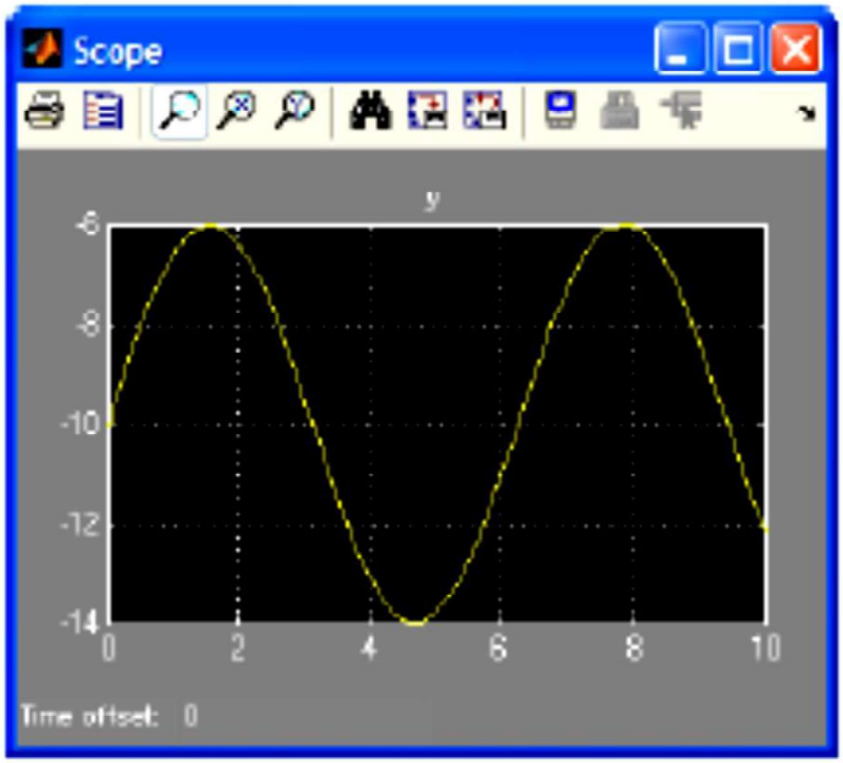
Demonstration: Working with a simple model

Create a Simulink model to generate a sinusoidal waveform given by: $y = 4 * \sin(t) - 10$

Simulink Block Diagram:



Output:



Conclusion:

We have learnt about the basics and different command used in MATLAB and SIMULINK.

Attempt the following and attach at the end of the experiment:

Q.1 Explain any 5 commands/syntax used in MATLAB (other than already mentioned in the report) and give a simple example how to use them. (Do not use basic syntax such as clc, clear all, etc). You may create a small code also to explain. (5 marks)

Solution:

1. **linsolve**: It is used to find solution to set of Linear Equations
   Example:

   A= [1,5;1,-4] B=[3,0]

   linsolve(A,B)

2. **ylabel:** It is used to name the y axis of the plot.
   Example:

   ylabel (Name,Value)
   t = ylabel( )

3. **Legend**: It creates descriptive label for plotted data.
   Example:

   legend ('cos(x)' ,' cos(2x)')

4. **Set**: to set the several properties in plot.
   Example:

   set (p,'Color','red')

5. **Polar**: We can plot the polar coordinates in Cartesian plane.
   Example:

   polar(theta, rho,'--r')

Q.2 Explain any three components/block from the Simulink library. You may also add the screenshot of the properties of the block. (5 marks)

Solution:

- Subsystem Reference: Reference to a reusable group of blocks with a dynamic interface, which can be visual or functional.
- Variant System: Multiple implementations of a component with only one active implementation. Variant systems allow you to address different sets of requirements within a single model.
- Inport and Outport: Use of port blocks is to move data (signals) and events (function calls) from outside a Subsystem block or referenced Model block to within the block, and vice versa.
- Bus Creator: Bus blocks combine signals into a virtual bus and manage the routing of signals around a complex block diagram.

Experiment No:-04
Design of smart meter for recording the electricity consumption

Aim: To create a simulink model of smart energy meter to understand it functioning.

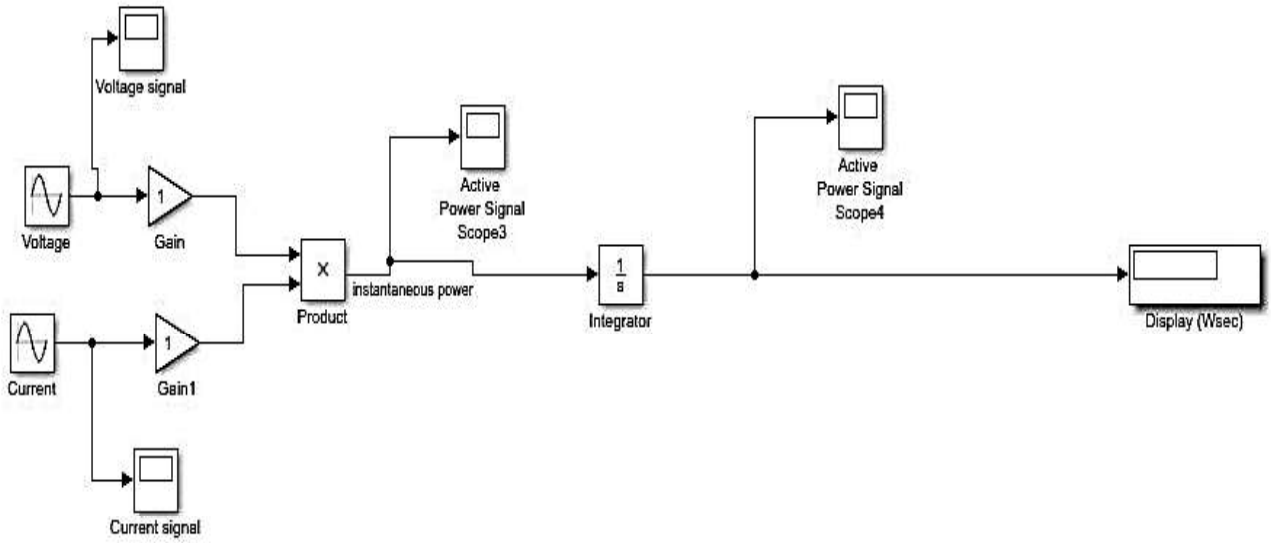
Theory:

- An electronic device that records consumption of electric energy in small intervals (an hour or less) and communicates that information daily back for monitoring and billing is called Smart energy meters.
- Smart meters provides two-way communication between the meter and the central system.
- Smart meters can also gather data for remote reporting.
- It can detect tampering of meter.
- Smart meter improves the power quality and is more efficient than conventional meters.
- Smart meter have automatic meter reading system through various networks.
- If at a time the bill comes more than a specific limit than, it will limit the use in the next month automatically.

Working of Smart energy meters:

Smart energy meters first measures or record the current flow and the voltage at regular intervals of time such as an hour or half and then it adds up all this and calculate the power flow for that specific interval of time. Then this data will be made available to the supplier or the company through various networks that helps in the communication system. At last the bills will be made available to the consumers.

Simulink diagram:

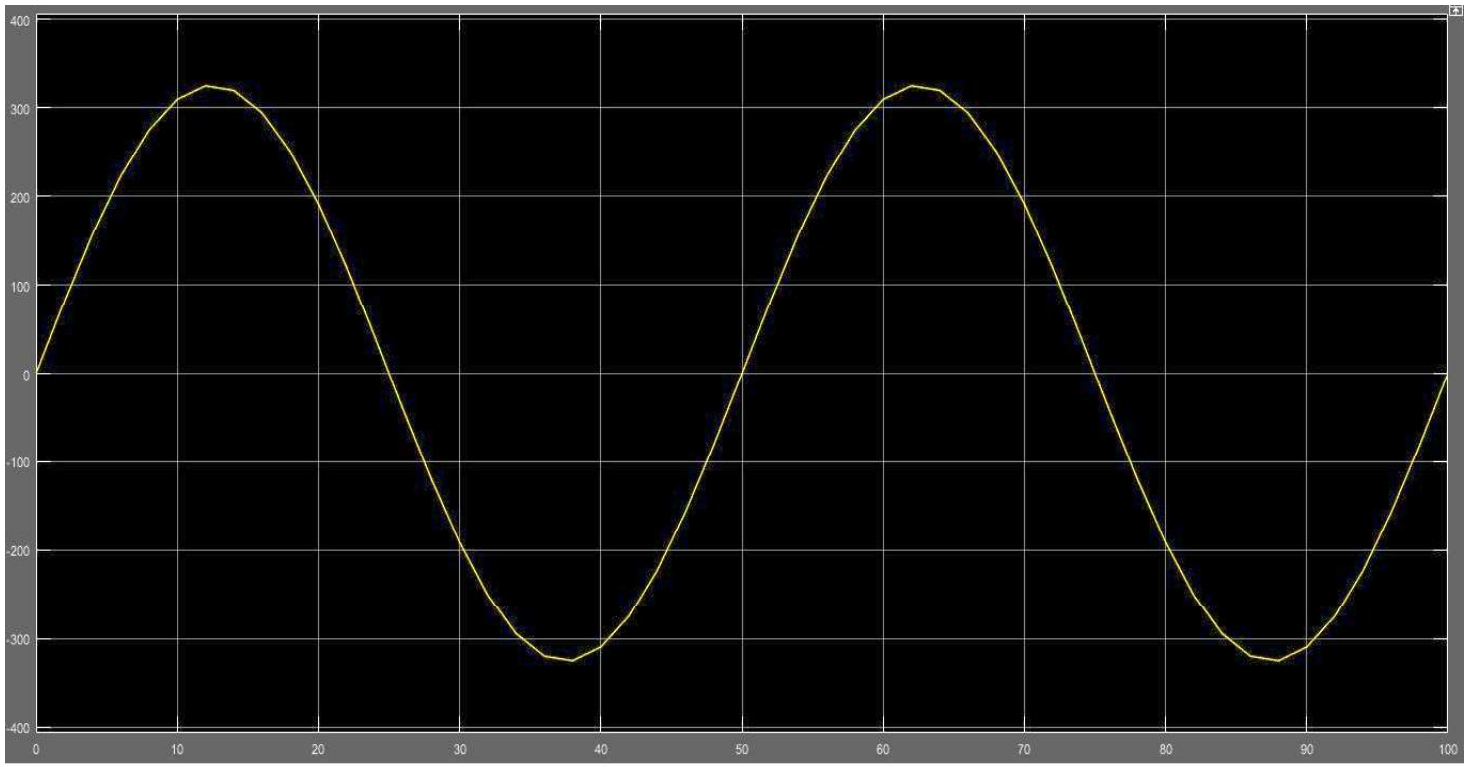


Input parameters:

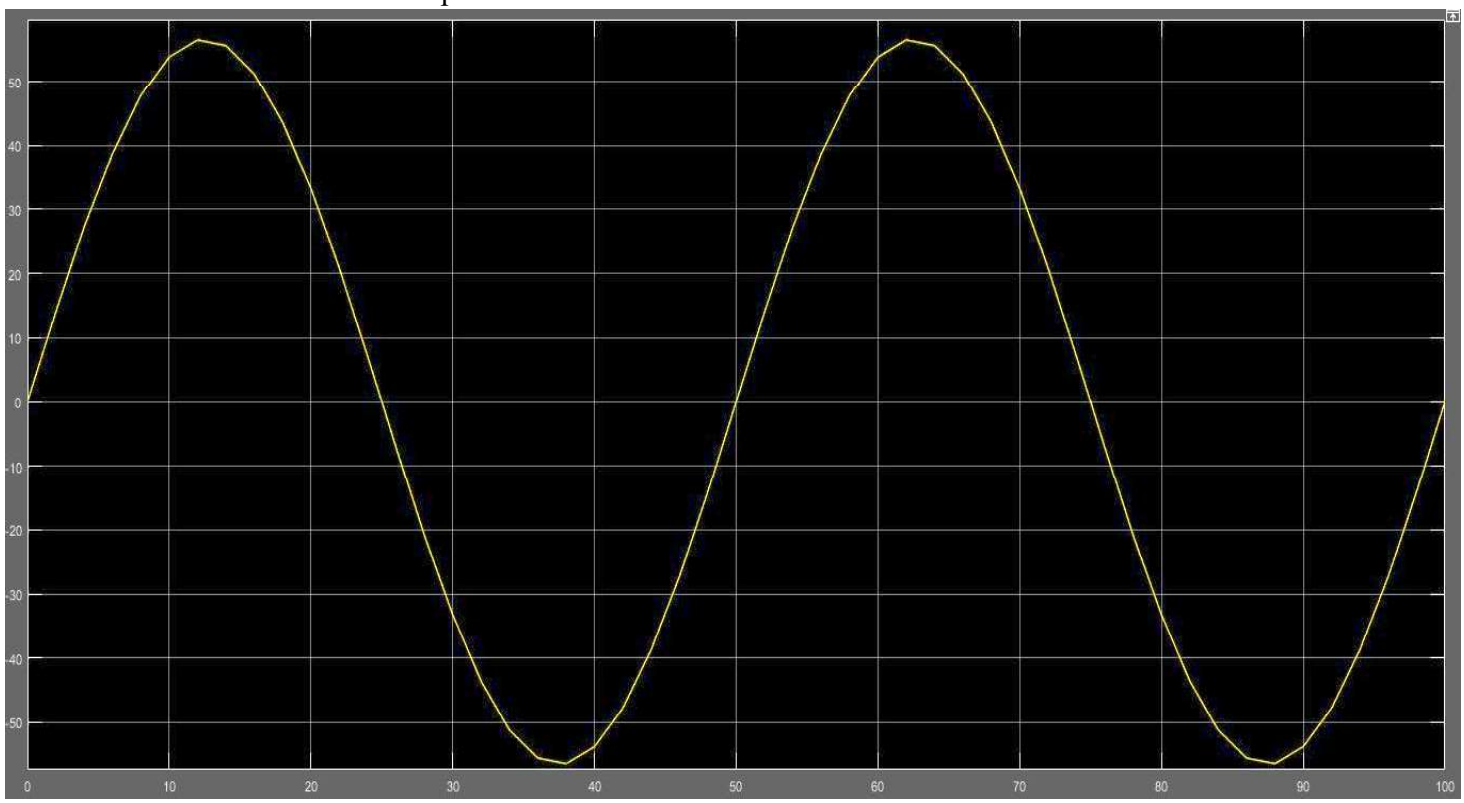
Voltage = 230 V single phase

Current = 56.56 A

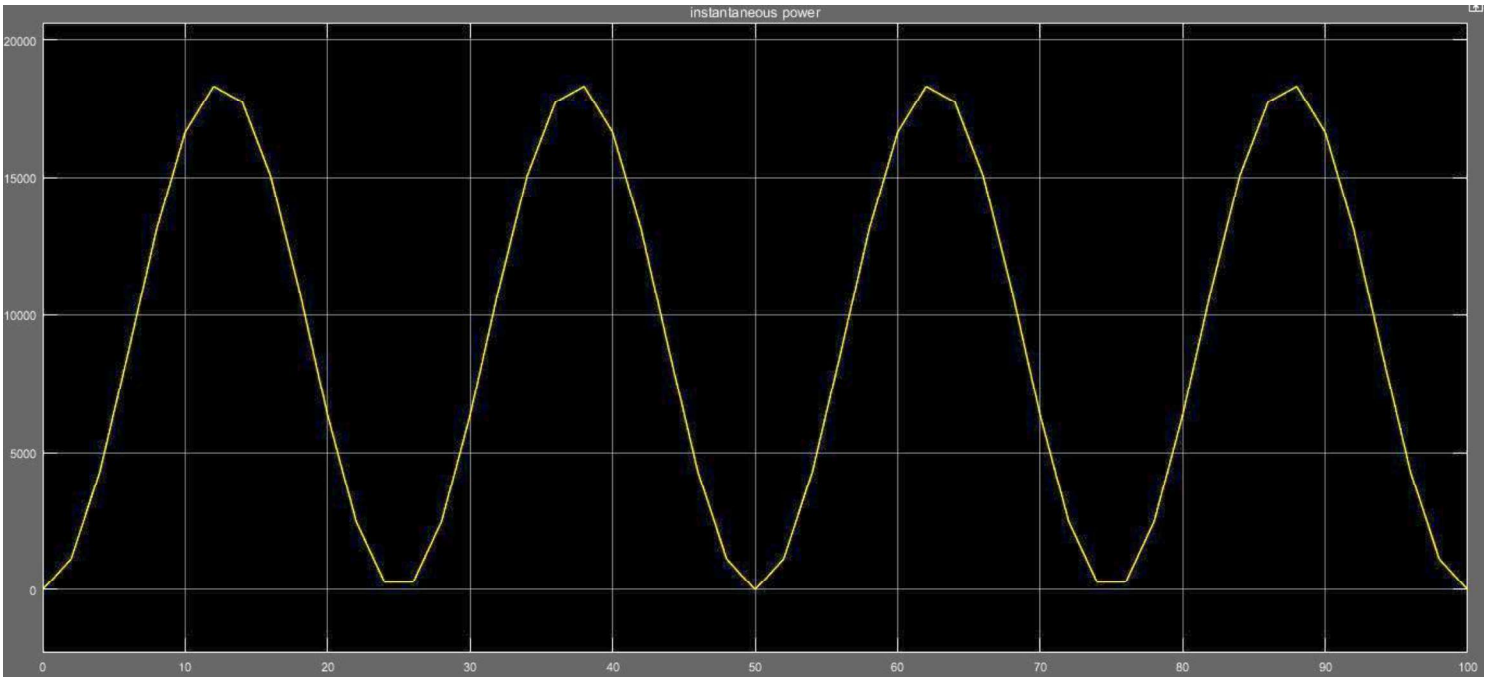
Voltage waveform from scope:



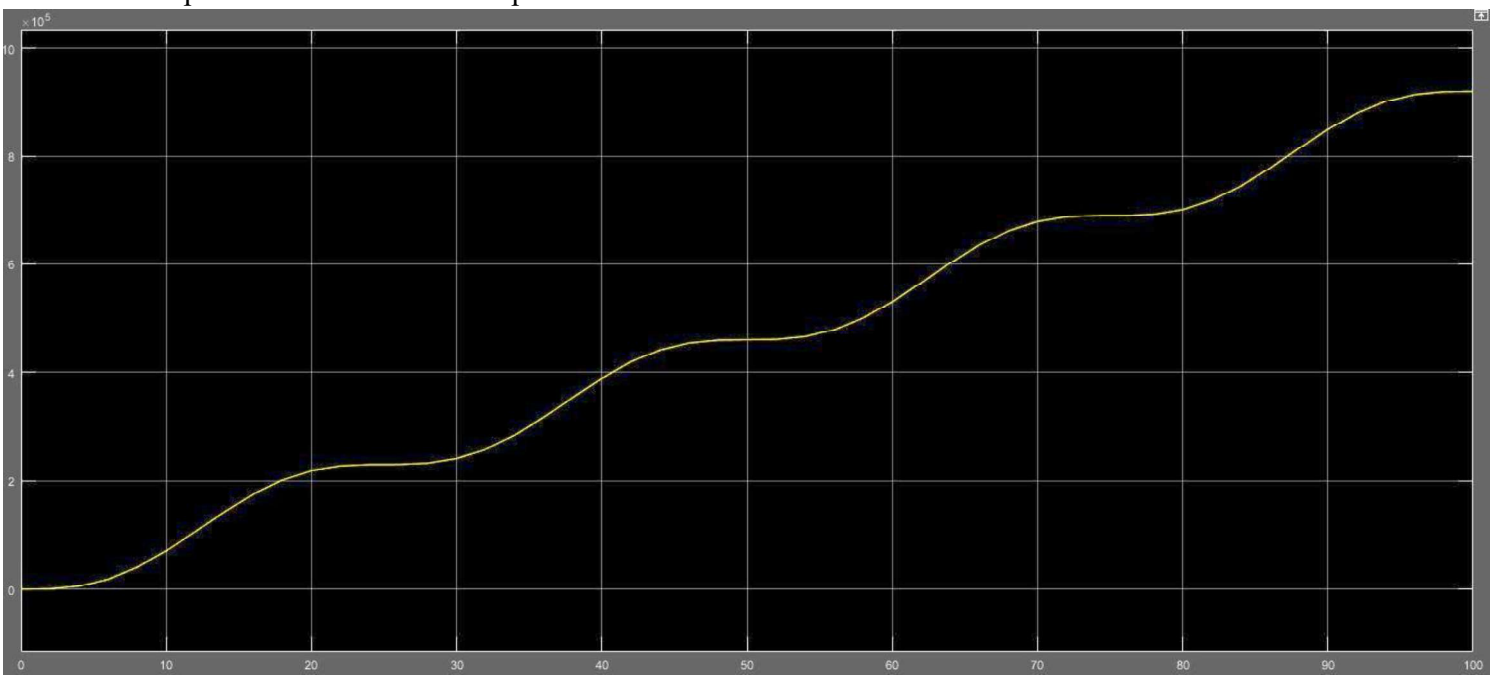
Current waveform from scope:



Active power waveform from scope3:



Active power waveform from scope4:



Working Principle:

- The AC current and voltage are calculated for a particular instance of time using Gain for voltage and Gain1 for current blocks.
- The X (Product block) is used to multiply the received instantaneous voltage and current to calculate instantaneous Power for the given inputs.
- It is then integrated by the Integrator block for a certain period of time and then final output is displayed at Display block in w/sec.
- The Signal Blocks give us waveforms for the corresponding Inputs i.e. voltage signal block for voltage Input and current signal Block for current Input.
- The Active Power Signal Scope3 Block is for Instantaneous Power and Active Power Signal Scope4 block is for Final Power generated as a function of time.

Conclusion:

The conclusion is that smart energy meter are better than conventional meters as they are more efficient and helps to save the money and energy.

Attempt the following and attach at the end of the experiment:

Q.1 Explain the advantages of smart energy meters over conventional meters. (3 marks)

Solution: The advantages of smart energy meters over conventional meters are:

- Smart energy meters provide more accurate bills than conventional meters and also prevents some other person coming in our home.
- Smart energy meters takes reading in small intervals that is hourly while conventional meters gives reading at the end of two to three months.
- It helps to save the money as we are able to know the usage and so that we can control it by changing our habits.
- It provides two way communication while there is not communication network in conventional meters.

Q.2 What are the challenges faced in implementing/installing smart energy meters on a wide scale (2 marks)

Solution: The challenges faced in implementing/installing smart energy meters on a wide scale:

- Cost of installing a smart energy meter is very high.
- Difficulty in testing of the components.
- It is difficult to change the existing conventional meters to the smart meters.
- Communication network is not strong in the rural areas and therefore it is difficult to install.

Experiment No:-05
Design of Ultrasonic proximity sensors with Arduino

Aim: To design an ultrasonic proximity sensor using Arduino.

Components: Arduino Uno, HC-SR04 proximity sensor, breadboard, jumper wires, small object for demonstration (pen, ball or disk)

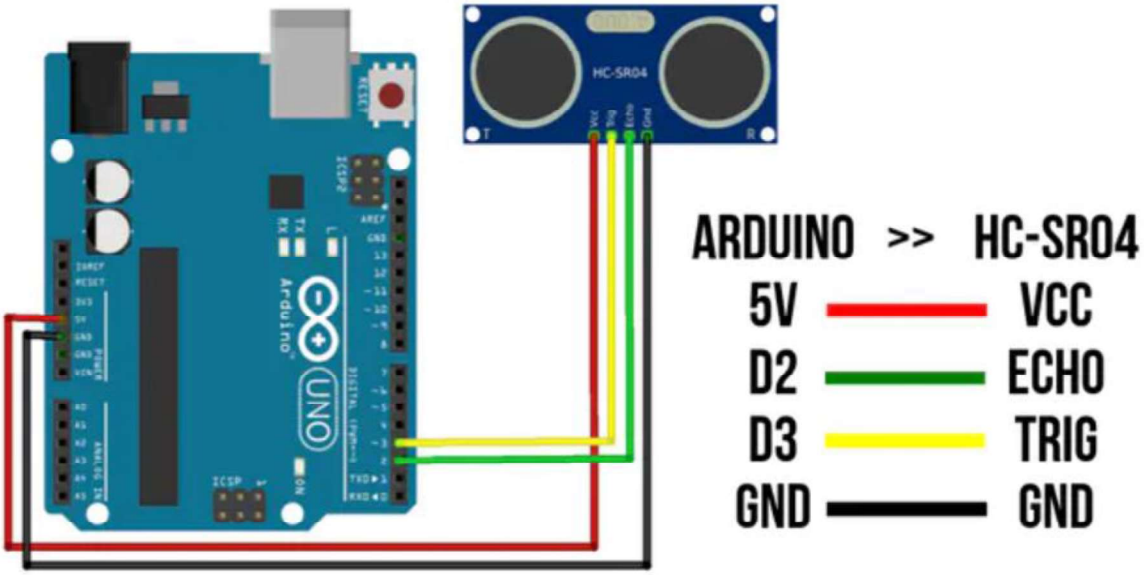
Theory:

- The sensors that can detect the presence of objects nearby without physical contact is called proximity sensors.
- With the help of electromagnetic radiation it finds the object by looking the changes in the field that has returned to it.
- The object is called the target and according to it the different proximity sensors are used to detect it.
- It can be used for a long interval of time.
- Different types of proximity sensors are: Capacitive, Optical, Ultrasonic etc.

Working of proximity sensor:

Proximity sensor produces electromagnetic field with the help of power supply, when the field is interrupted by the object or target then a current is produced and it moves in the object and due to this field changes and the proximity sensor is able to detect it.

Connection diagram:



Connection of Arduino UNO and HC-SR04

Code implementation for sensor application:

For calculating the distance of the target or object from the motion sensor the loop function is used. It send the Ultrasonic wave with the help of transmitter and then receives the wave after getting interrupted by the object with the help of receiver.

The formula is:

S= (t*0.034)/2

where t is the time taken by the Ultrasonic Sound wave to go from transmitter and return to receiver.

Code:

```
#define echoPin 2 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 3 //attach pin D3 Arduino to pin Trig of HC-SR04
// defines variables
long duration; // variable for the duration of sound wave travel
int distance; // variable for the distance measurement
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
  Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate speed
  Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor
  Serial.println("with Arduino UNO R3");
}

void loop() {
  // Clears the trigPin condition
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
```

Conclusion:

Proximity sensors are used to detect the object without having the physical contact and it has number of applications such as in self driving cars and smartphones etc.

Attempt the following and attach at the end of the experiment:

Q.1 Explain practical application of proximity sensor; one specific for your domain engineering (Electrical, CSE, Chemical appropriately) and one for day to day common usage. (6 marks) (Maximum one page limit)

Solution: A practical application of proximity sensor is line follower robot:

- The line follower robot is the self-operated robot. It follows the line drawn on the surface.
- The line is indicated by the white line on a surface or black line on a white surface. The system must sense the line so that it can follow or detect it.
- This application of sensors is very important in this as two sensors that is proximity sensor and IR sensor are used for the path detection.
- The proximity sensor is used for path detection and IR sensor is used for obstacle or object detection. These sensors are fixed at front end of the robot.
- The microcontroller is used to control the whole device.

Day to day application of proximity sensor:

- The proximity sensor are used in the Roller Coaster in the amusement parks and other places.
- Rides such as roller coasters, where the sensor is mounted on the track and they can detect the metal by help of inductive proximity sensors.
- Inductive proximity sensors are the good replacements for limit switches.