माधव प्रौद्योगिकी एवं विज्ञान संस्थान, ग्वालियर (म.प्र.), भारत
MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR (M.P.), INDIA
Deemed to be University
(Declared under Distinct Category by Ministry of Education, Government of India)
NAAC ACCREDITED WITH A++ GRADE

NEC - MATLAB SIMULINK(2000115)
Semester V

Assignments

Submitted to:
Dr. Deepak Batham
Assistant Professor
Department of Electronics Engineering

Submitted by:
Disha Pal
0901ET221021
Department of Electronics Engineering

ASSIGNMENT 1

Q.1 Give a brief introduction of MATLAB Simulink? Explain How Matlab Simulink is used in Engineering Applications?

Ans -

Introduction to MATLAB Simulink:

MATLAB Simulink is a graphical programming environment for modeling, simulating, and analyzing dynamic systems. It is part of the MATLAB software suite and provides a user-friendly interface where users can create block diagrams to represent systems visually. With an extensive library of pre-built blocks for mathematical operations, signal processing, control systems, and more, Simulink facilitates the design and simulation of complex systems without requiring extensive coding.

How MATLAB Simulink is Used in Engineering Applications:

1. Control Systems Design: Engineers use Simulink to design and simulate control systems, such as PID controllers, by creating block diagrams that represent the system dynamics and feedback loops. This helps in tuning parameters for optimal performance.

2. Signal Processing: Simulink is employed for designing and testing algorithms for processing signals, such as filtering and modulation. The visual nature of the tool allows for easy modifications and realtime analysis of the effects of changes.

3. Mechanical and Aerospace Engineering: Simulink is used to model the dynamics of mechanical systems, including robotics, automotive systems, and aerospace applications. Engineers can simulate how systems behave under various conditions and optimize designs accordingly.

4. Electrical Systems: In electrical engineering, Simulink aids in the simulation of electrical circuits, power systems, and electronics. It enables the modeling of system behaviors under different electrical loads and helps in the design of energy management systems.

5. System-Level Simulation: Simulink allows for the integration of different subsystems into a cohesive model. This is crucial in multidisciplinary

engineering projects, where multiple domains (like mechanical, electrical, and software) interact.

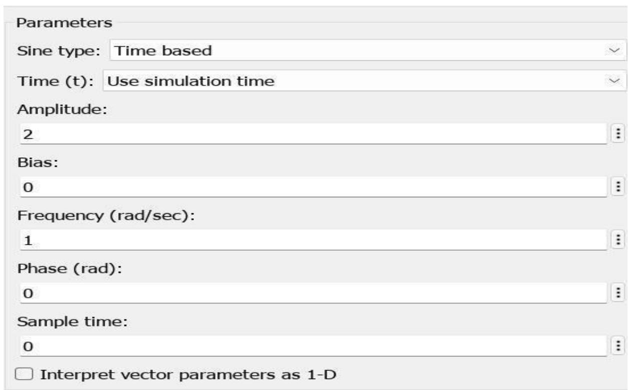
6. Rapid Prototyping: Engineers can use Simulink to quickly prototype and test algorithms. This helps in reducing the time and cost associated with hardware development by allowing for extensive simulation before physical implementation.

7. Model-Based Design: Simulink supports a model-based design approach, enabling engineers to design, simulate, and validate systems in a unified framework. This approach enhances collaboration across teams and ensures that designs meet requirements effectively.

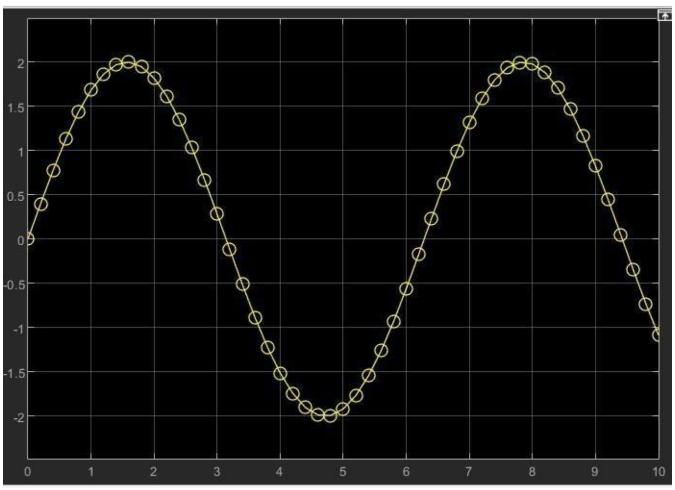
Q.2 Create Simulink Model to simulate and display all types of signals in oscilloscope using Simulink Sink and Source functional blocks:

a) Sine Wave -

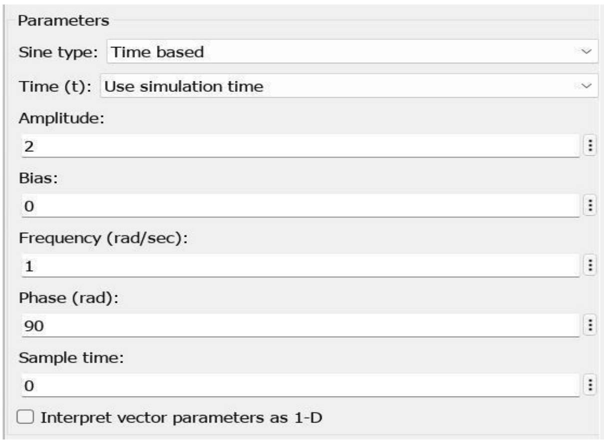
Model and Parameters:



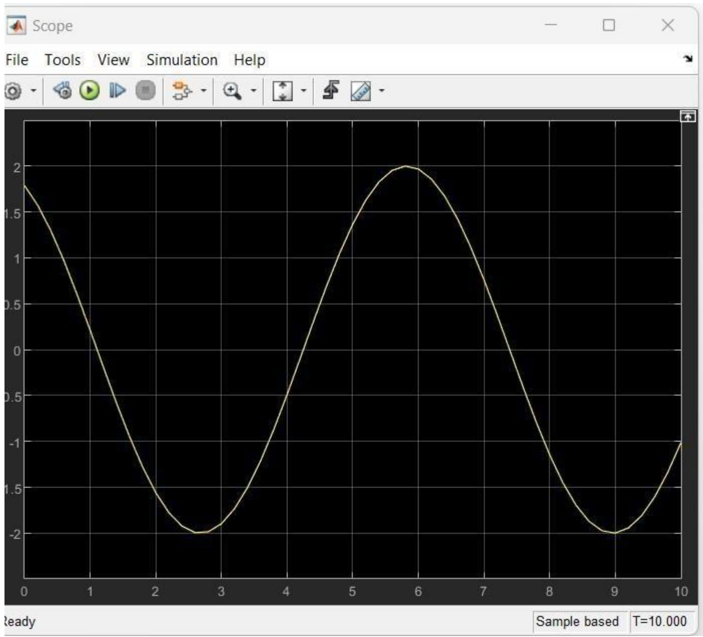
Output:



b) Cosine Wave -Model and Parameters:

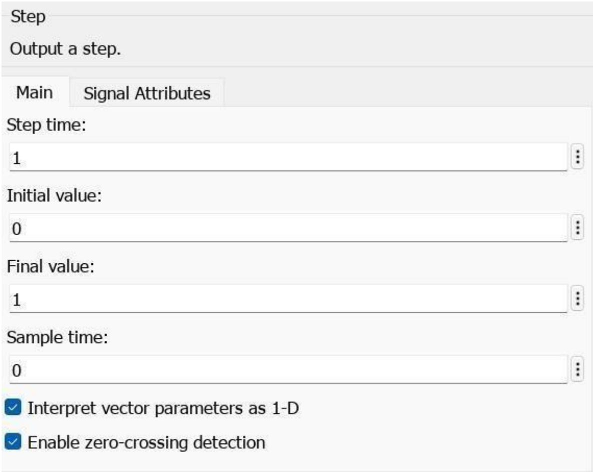


Output:

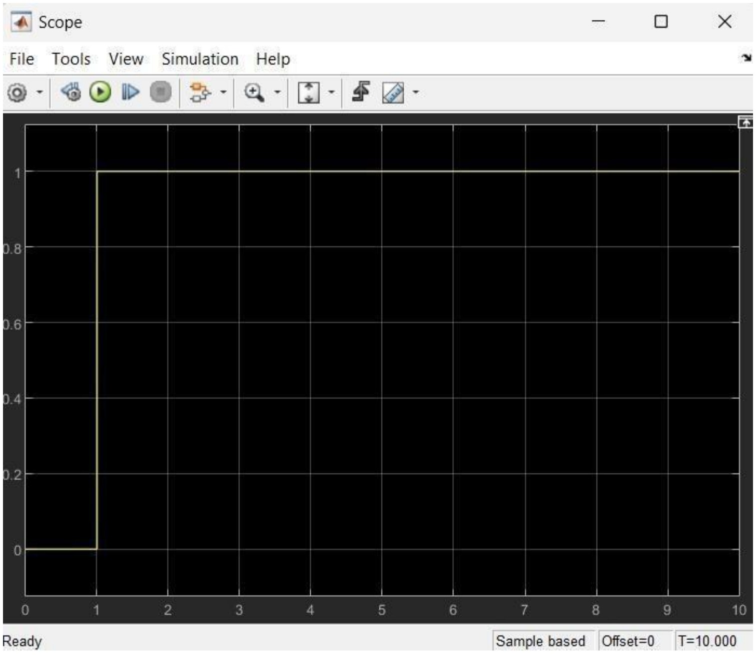


C) Step Function -Models

and Parameters:

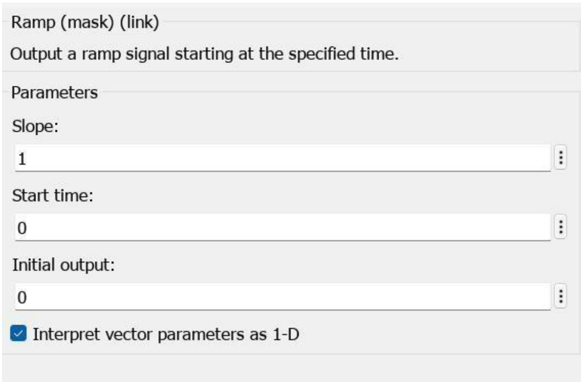


Output:

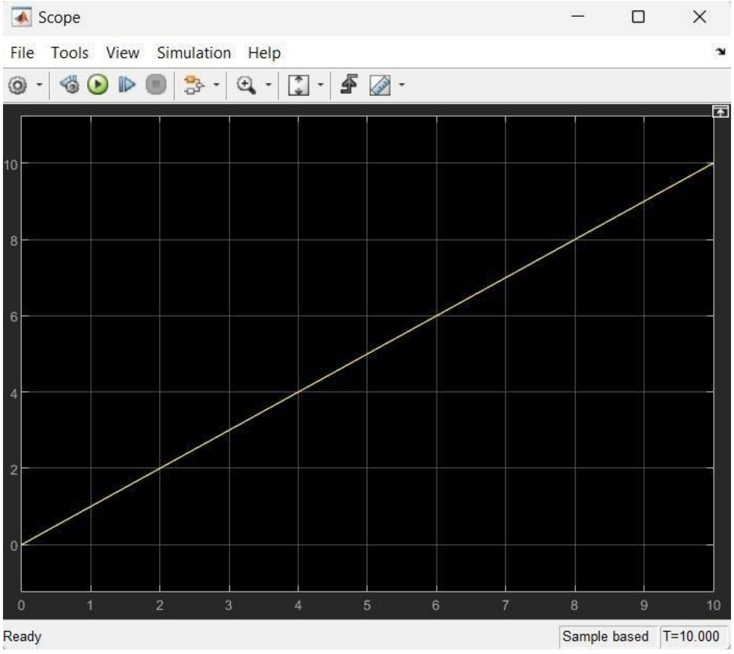


D) Ramp Function -Models

and Parameters:

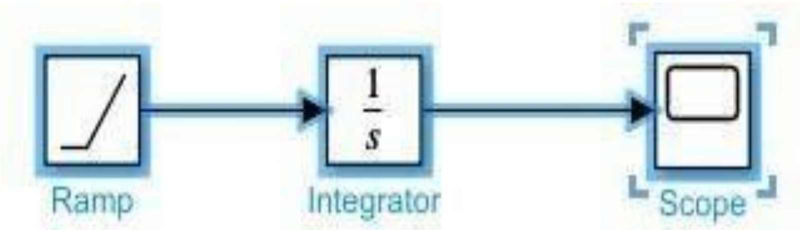


Output:

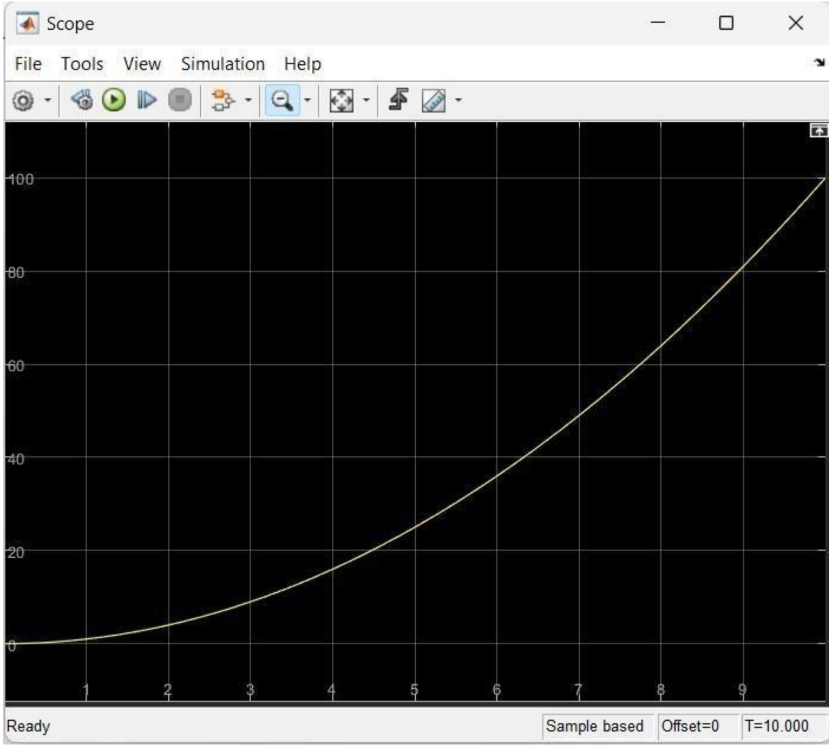


E) Parabolic Function -

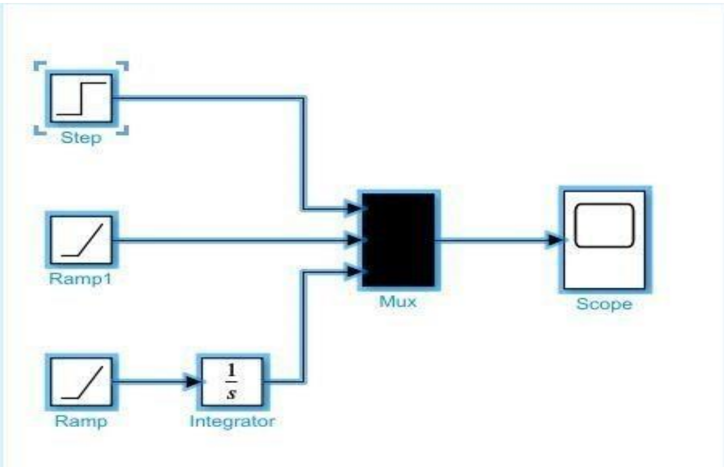
Model:



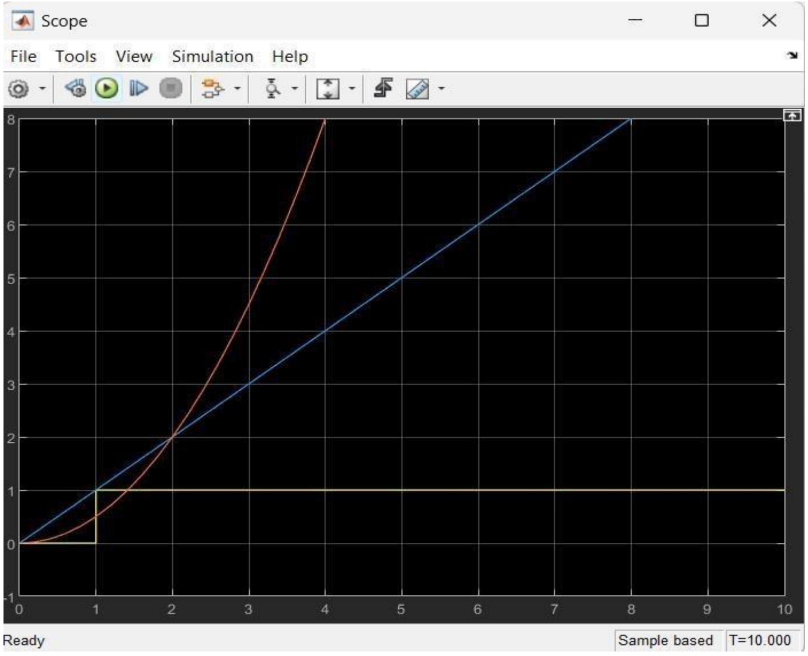
Output:



Q.3 Create a Simulink Model to Simulate and display at least three signals (step signal, ramp signal and parabolic signal) in one oscilloscope multiplexing signals. Ans Model:



Output:

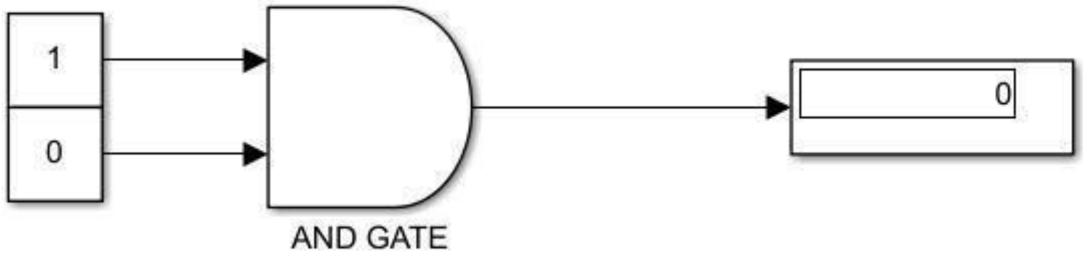


Scope
File Tools View Simulation Help
8
7
6
5
4
3
2
1
0
-1
0 1 2 3 4 5 6 7 8 9 10
Ready
Sample based
T=10.000

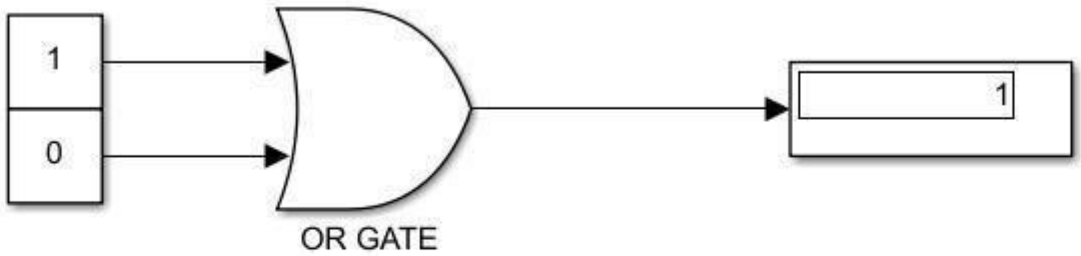
ASSIGNMENT 2

Q.1 Create a Simulink model to simulate and verify all logic gates.

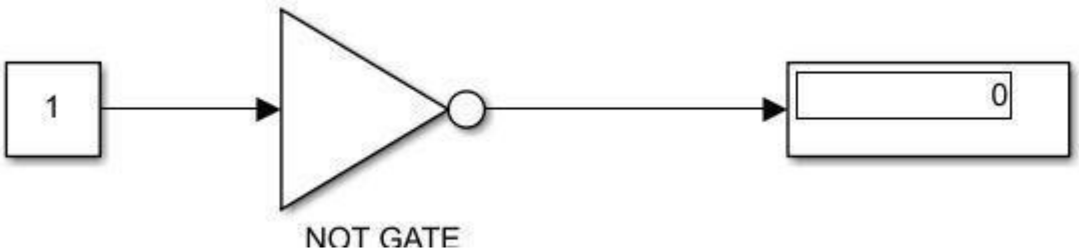
A. AND Gate



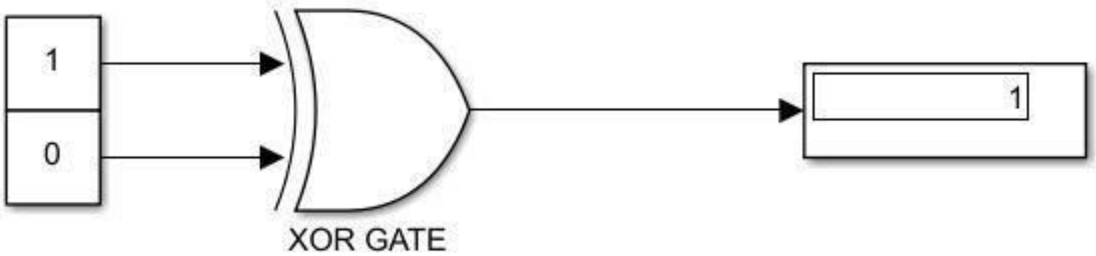
B. OR Gate



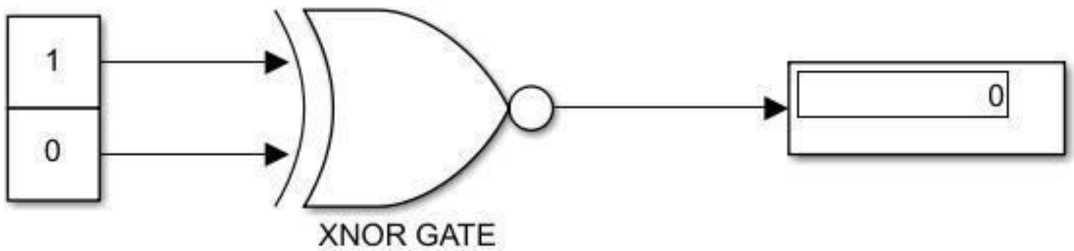
C. NOT Gate



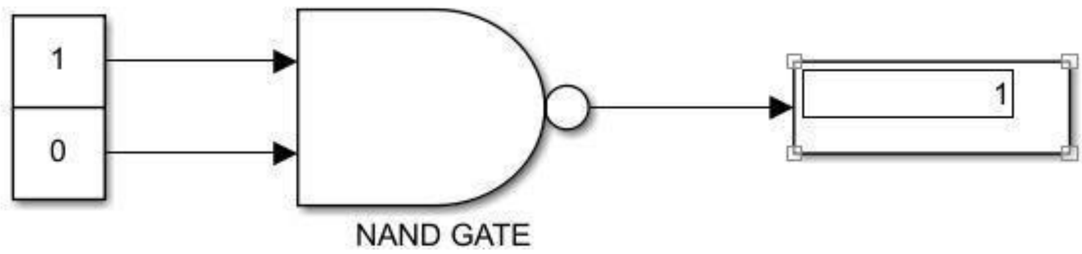
D. XOR Gate



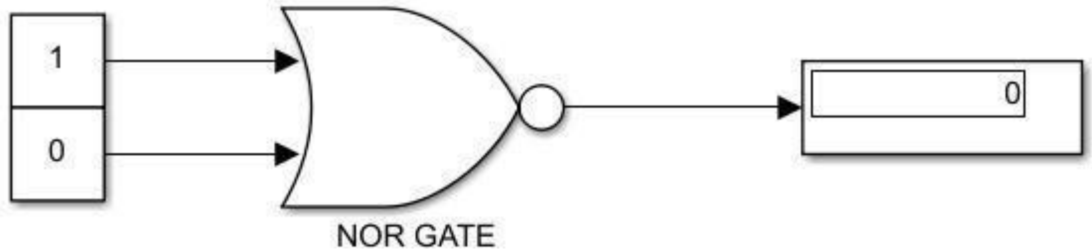
E. XNOR Gate



F. NAND Gate

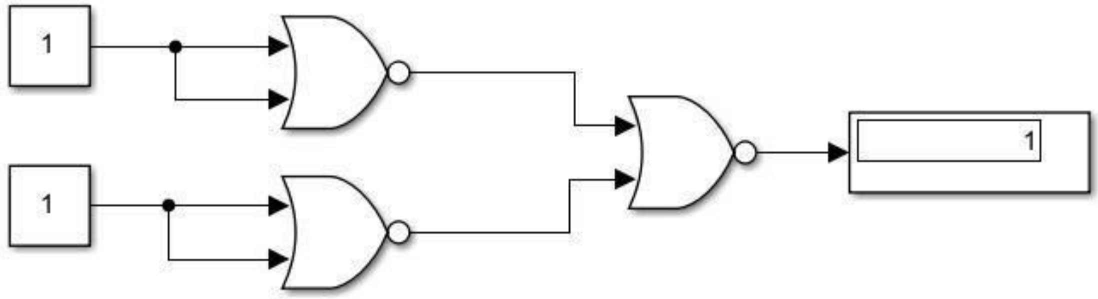


G. NOR Gate

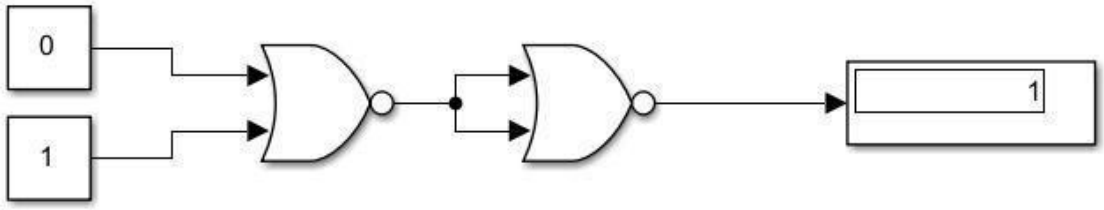


Q.2 Create a Simulink model and verify all logic gates.

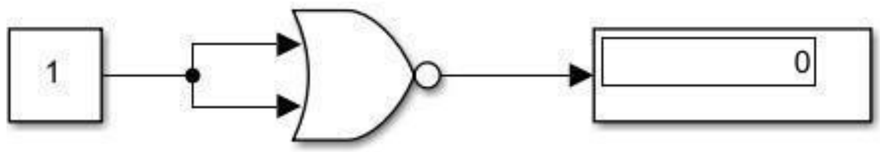
A) AND Gate Using NOR



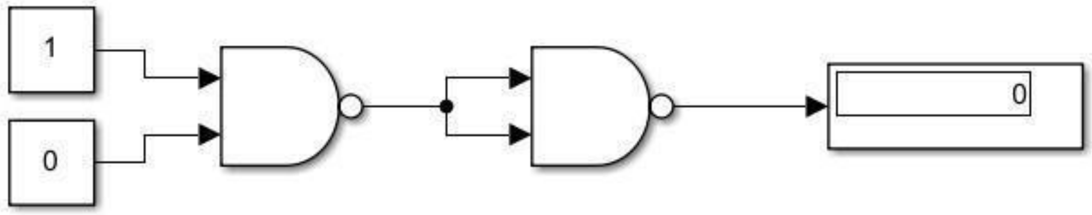
B) OR Gate Using NOR



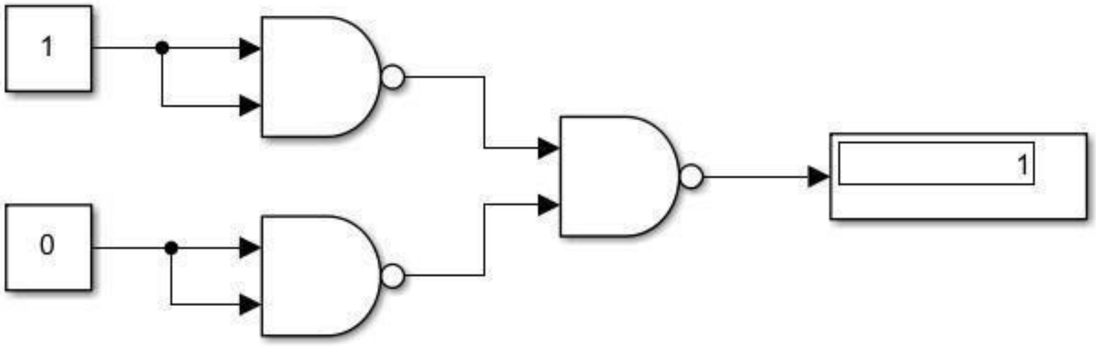
C) NOT Gate Using NOR



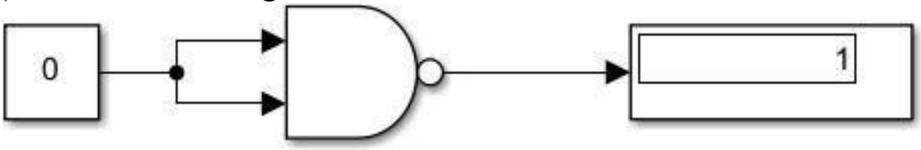
D) AND Gate Using NAND



E) OR Gate Using NAND

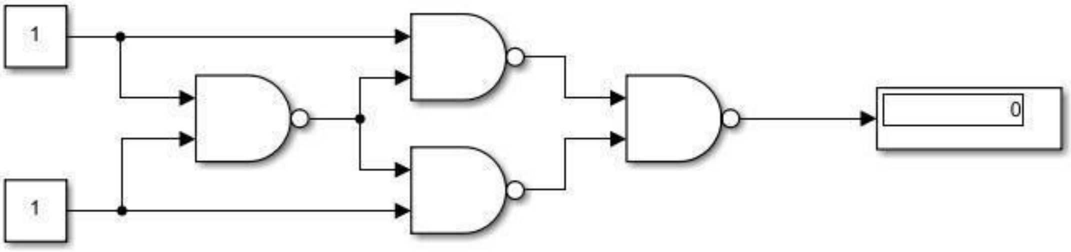


F) NOT Gate Using NAND

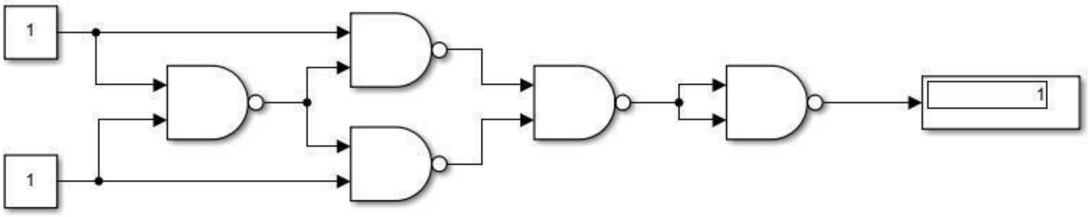


Q.3 Create a Simulink model to simulate and verify all logic gates.

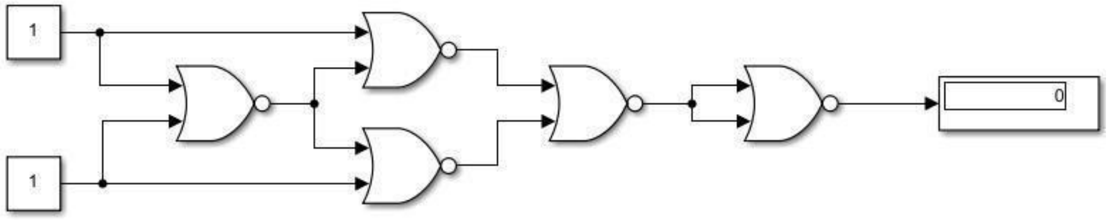
A) EXOR Using NAND



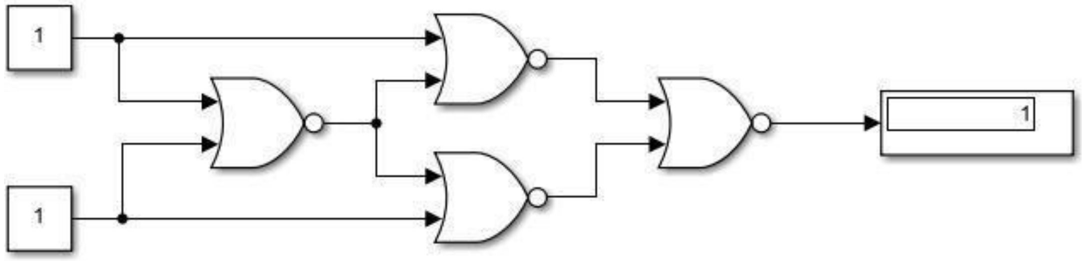
B) EXNOR Using NAND



C) EXOR Using NOR



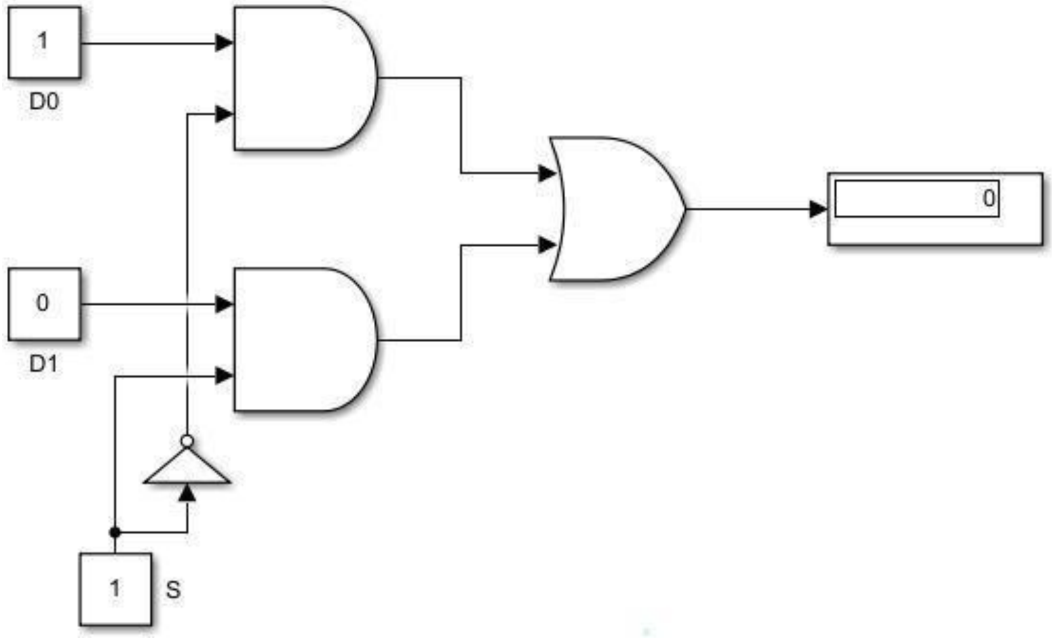
D) EXNOR Using NOR



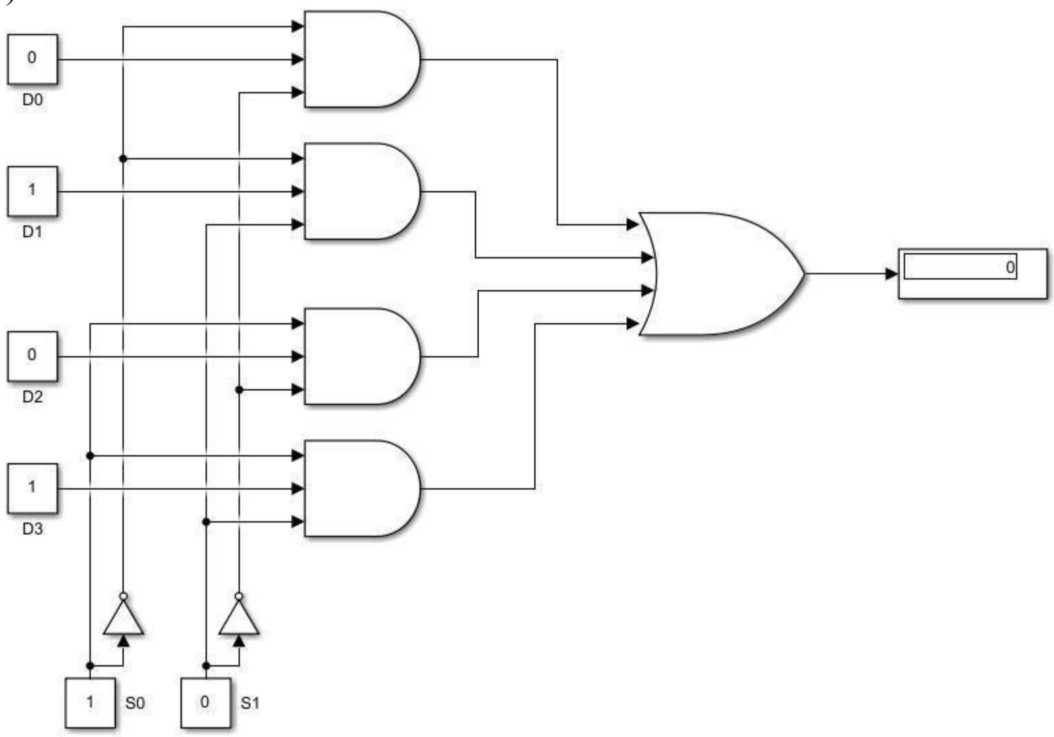
ASSIGNMENT 3

Q.1 Create a Simulink model to simulate and verify

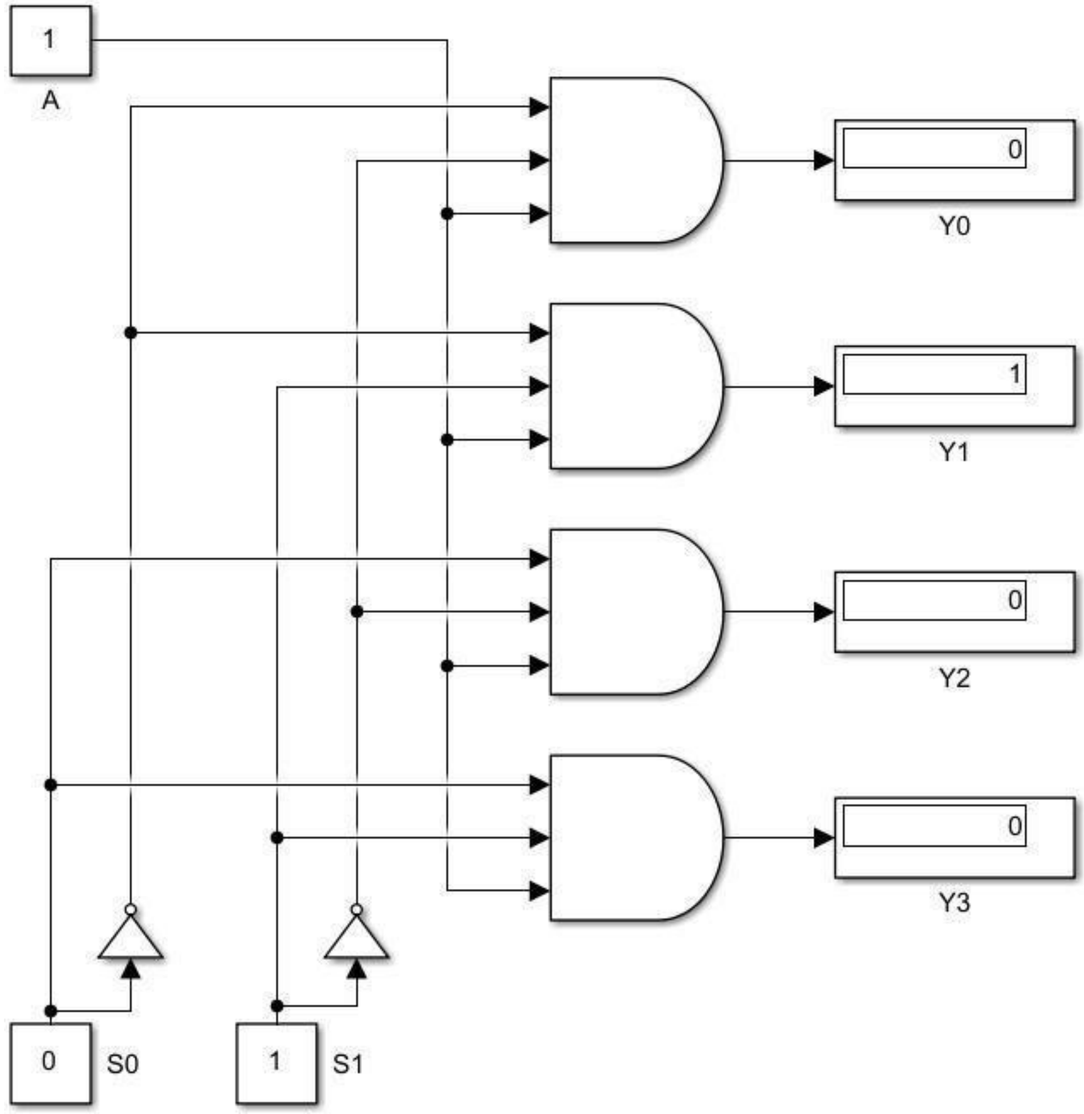
A) 2x1 MUX



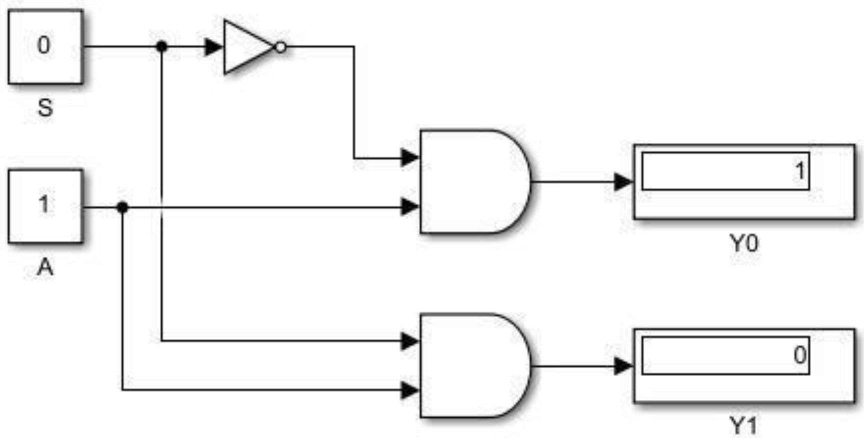
B) 4x1 MUX



C) 1x4 DEMUX



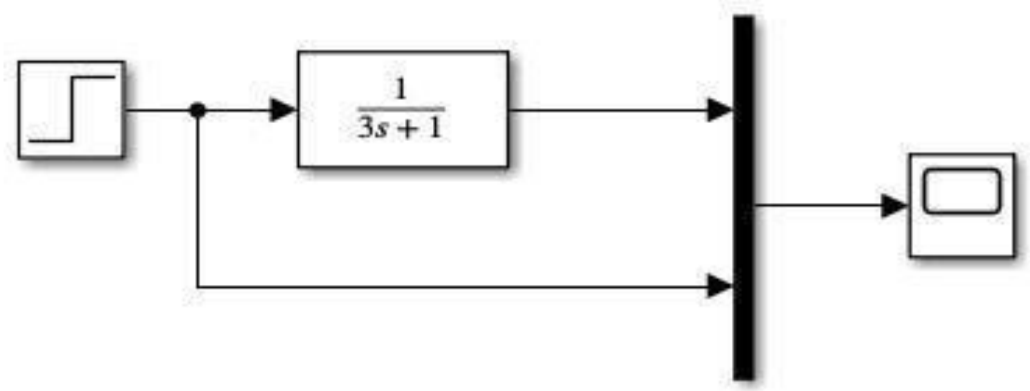
D) 1x2 DEMUX



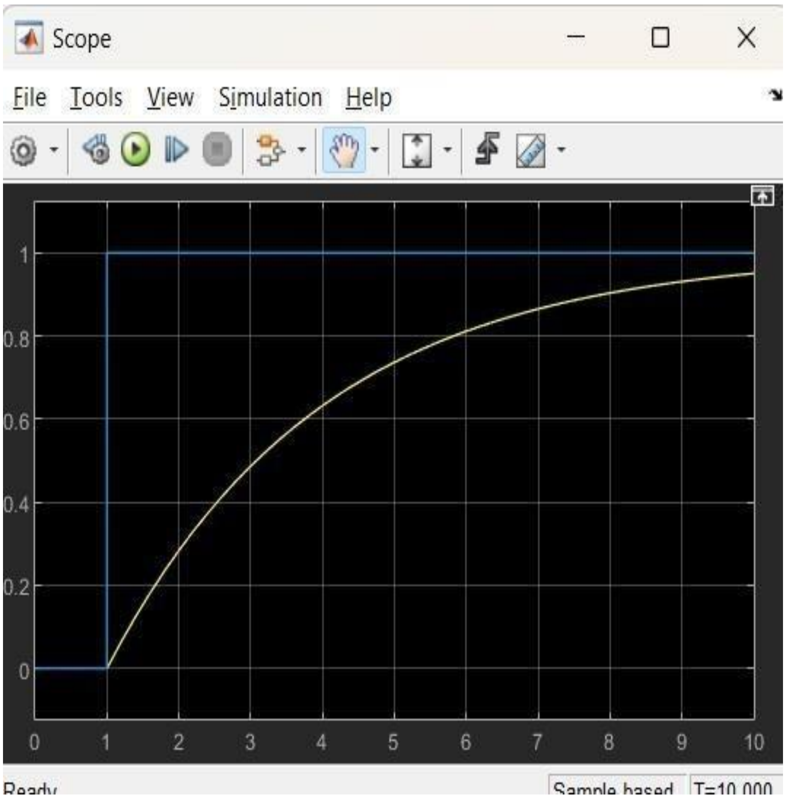
ASSIGNMENT 4

Q.1 Design a first order control system and analyze the system output for unit step input.

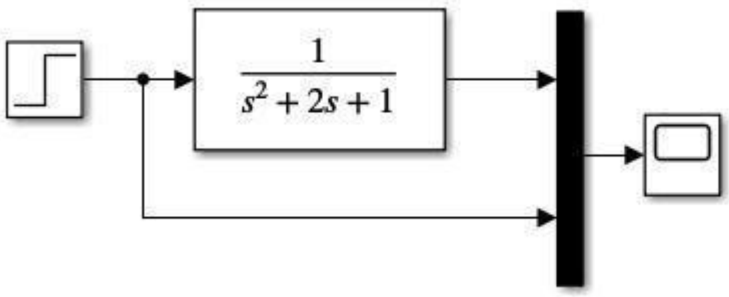
Simulink Model:



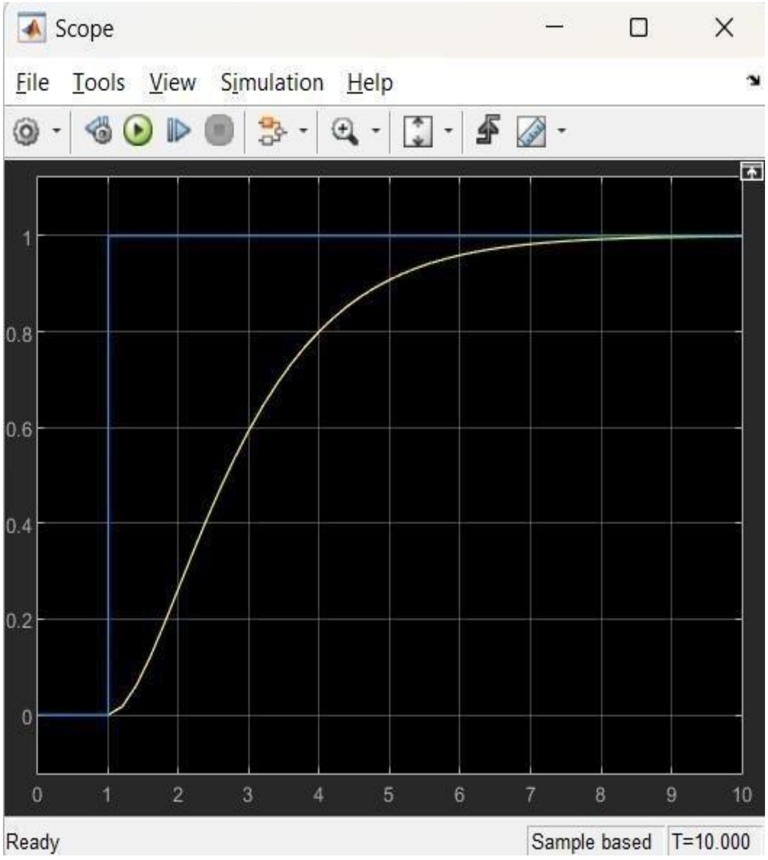
Output:



Q.2 Design a second order control system and analyze the system output for unit step input Simulink Model:

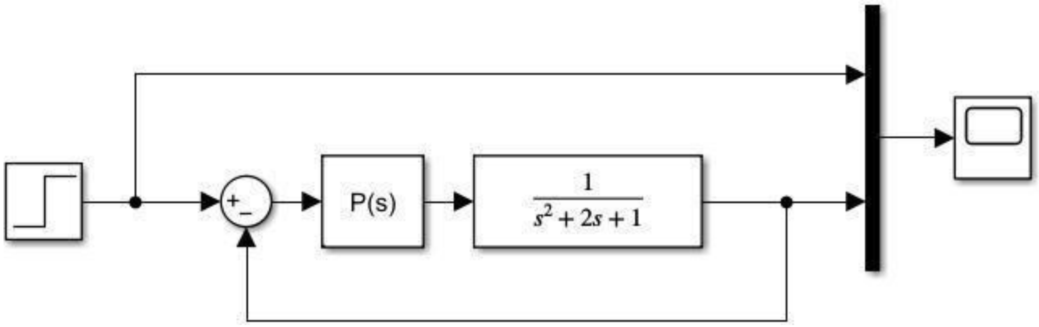


Output:

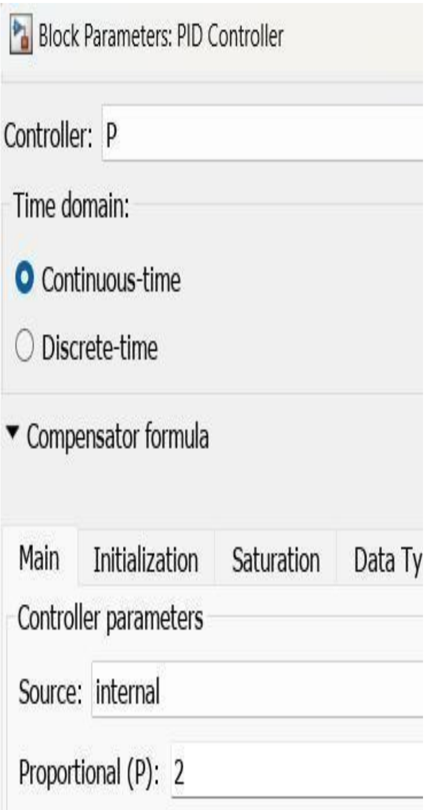


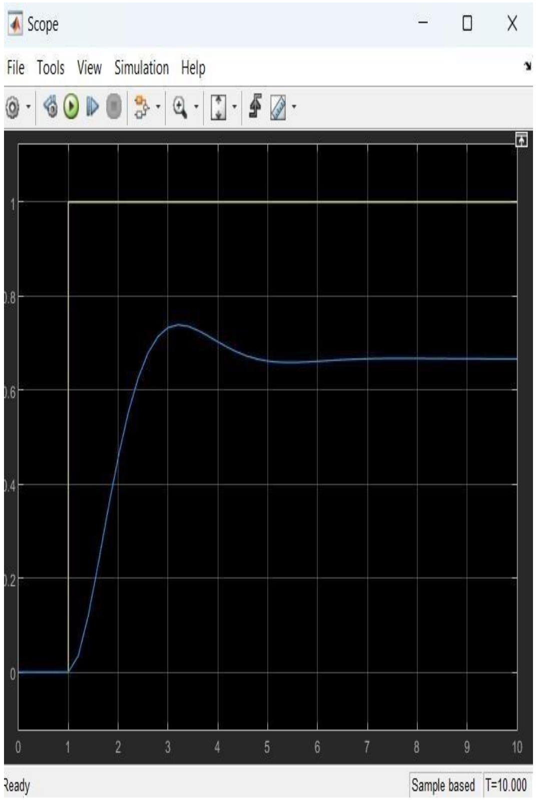
Q.3 Design a second order close loop control system with proportional (P) controller and analyze the system output for unit step input.

Simulink Model:



Parameters and Output:

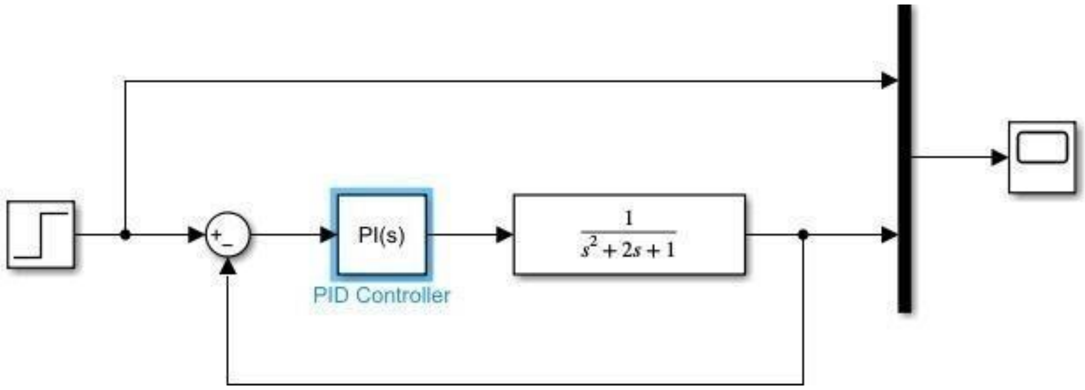




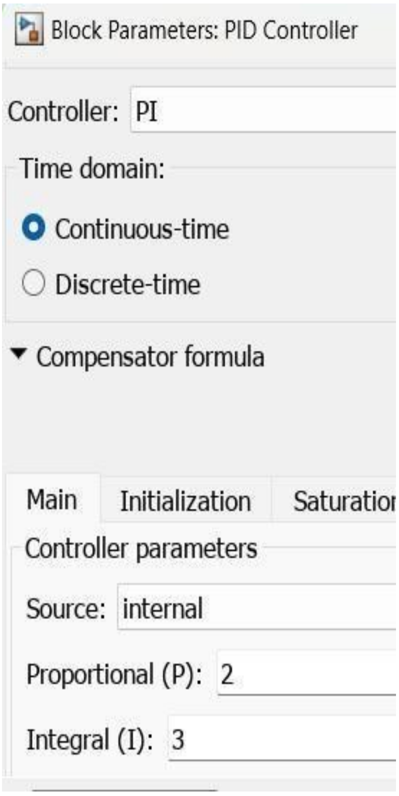
Proportional Integral (PI) Controller and analyze the system output for unit step input.

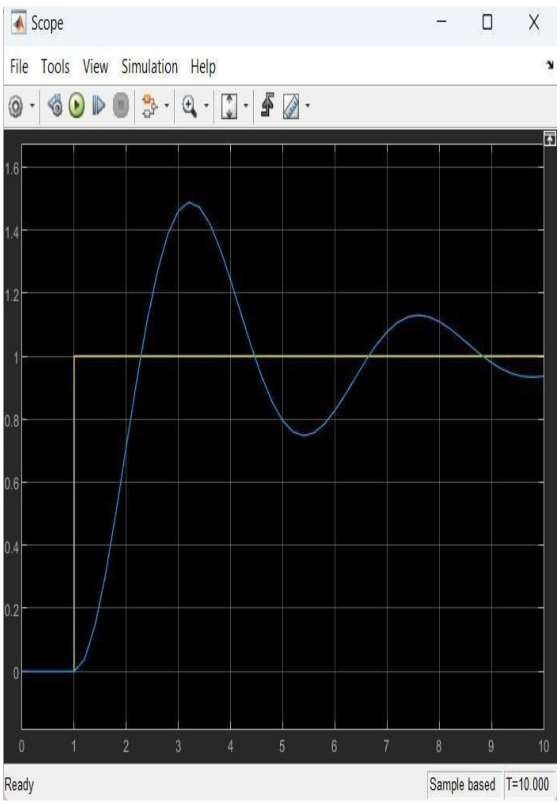
Q.4 Design a second order close loop control system with

Simulink Model:



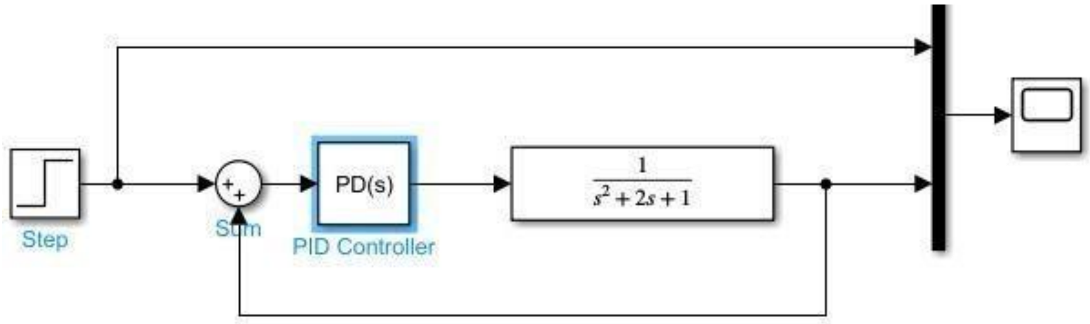
Parameters and Output:



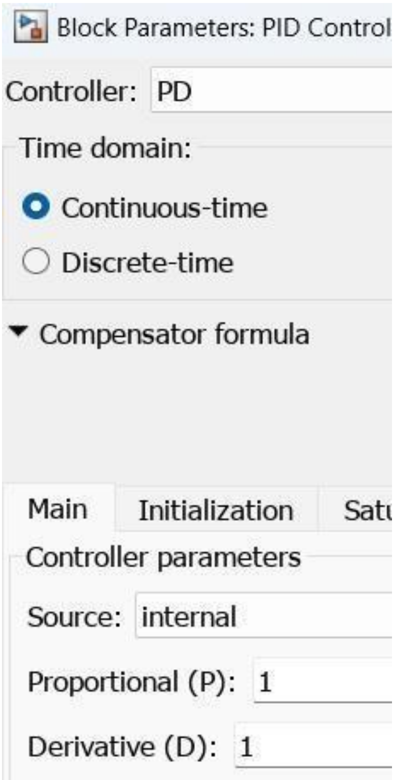


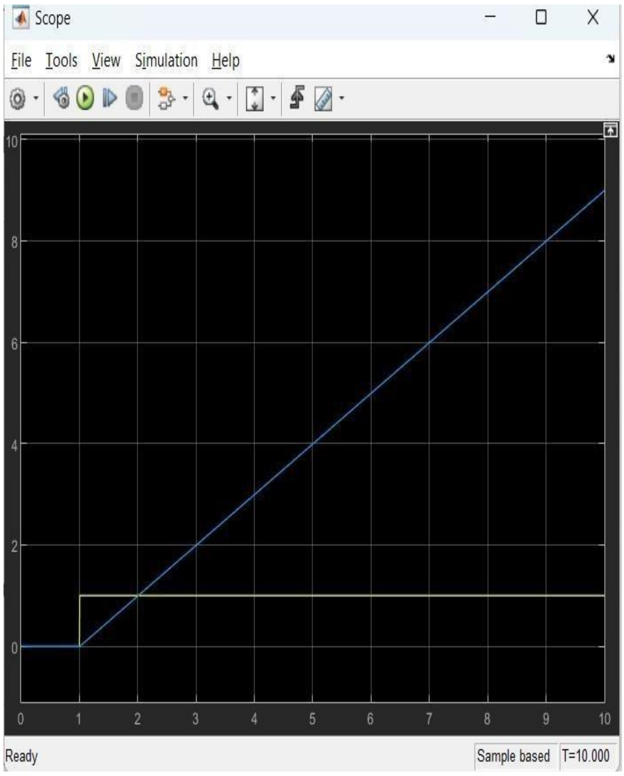
Q.5 Design a second order close loop control system with Proportional Derivative (PD) Controller and analyze the system output for unit step input.

Simulink Model:



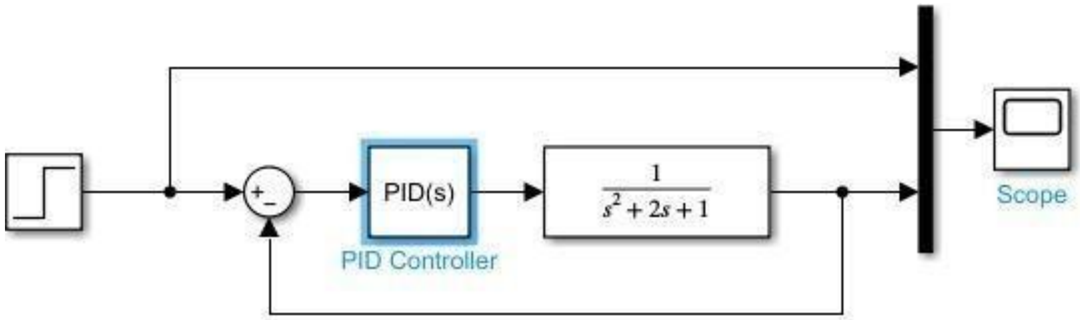
Parameters and Output:



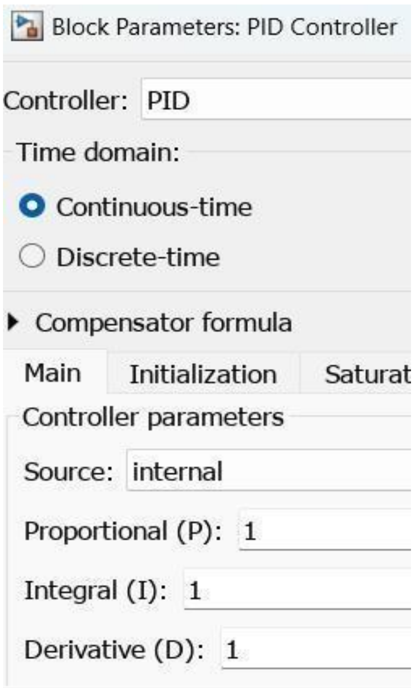


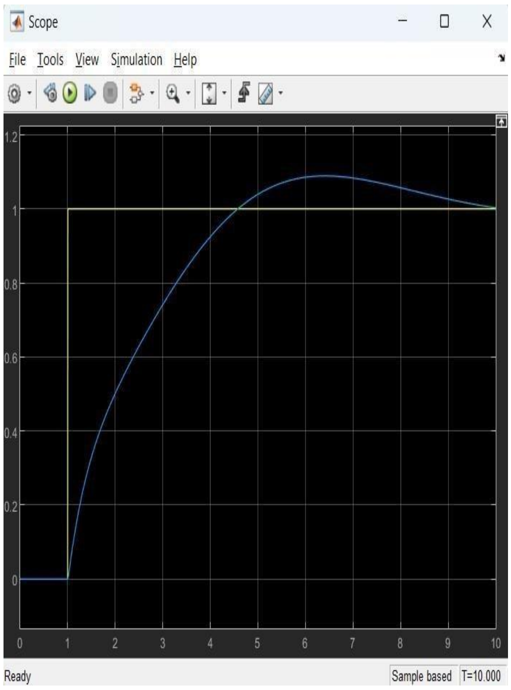
Q.6 Design a second order close loop control system with Proportional Integral Derivative (PID) Controller and analyze the system output for unit step input.

Simulink Model:



Parameters and Output:





ASSIGNMENT 5

Q.1 Design a GUI (Graphical User Interface) model for a Calculator.

Code for the GUI Model of Calculator:

```
function varargout = Calculator_1(varargin)
% CALCULATOR_1 MATLAB code for Calculator_1.fig
% CALCULATOR_1, by itself, creates a new CALCULATOR_1 or raises the existing %
singleton*.
%
% H = CALCULATOR_1 returns the handle to a new CALCULATOR_1 or the handle to
% the existing singleton*.
%
% CALCULATOR_1('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in CALCULATOR_1.M with the given input arguments.
%
% CALCULATOR_1('Property','Value',...) creates a new CALCULATOR_1 or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before Calculator_1_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property application %
stop. All inputs are passed to Calculator_1_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one %
instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Calculator_1

% Last Modified by GUIDE v2.5 10-Oct-2024 16:58:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1; gui_State =
struct('gui_Name', mfilename, ...
'gui_Singleton', gui_Singleton, ...
'gui_OpeningFcn', @Calculator_1_OpeningFcn, ...
'gui_OutputFcn', @Calculator_1_OutputFcn, ...
'gui_LayoutFcn', [] , ... 'gui_Callback', []); if
nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Calculator_1 is made visible.
function Calculator_1_OpeningFcn(hObject, eventdata, handles, varargin) %
This function has no output args, see OutputFcn.
```

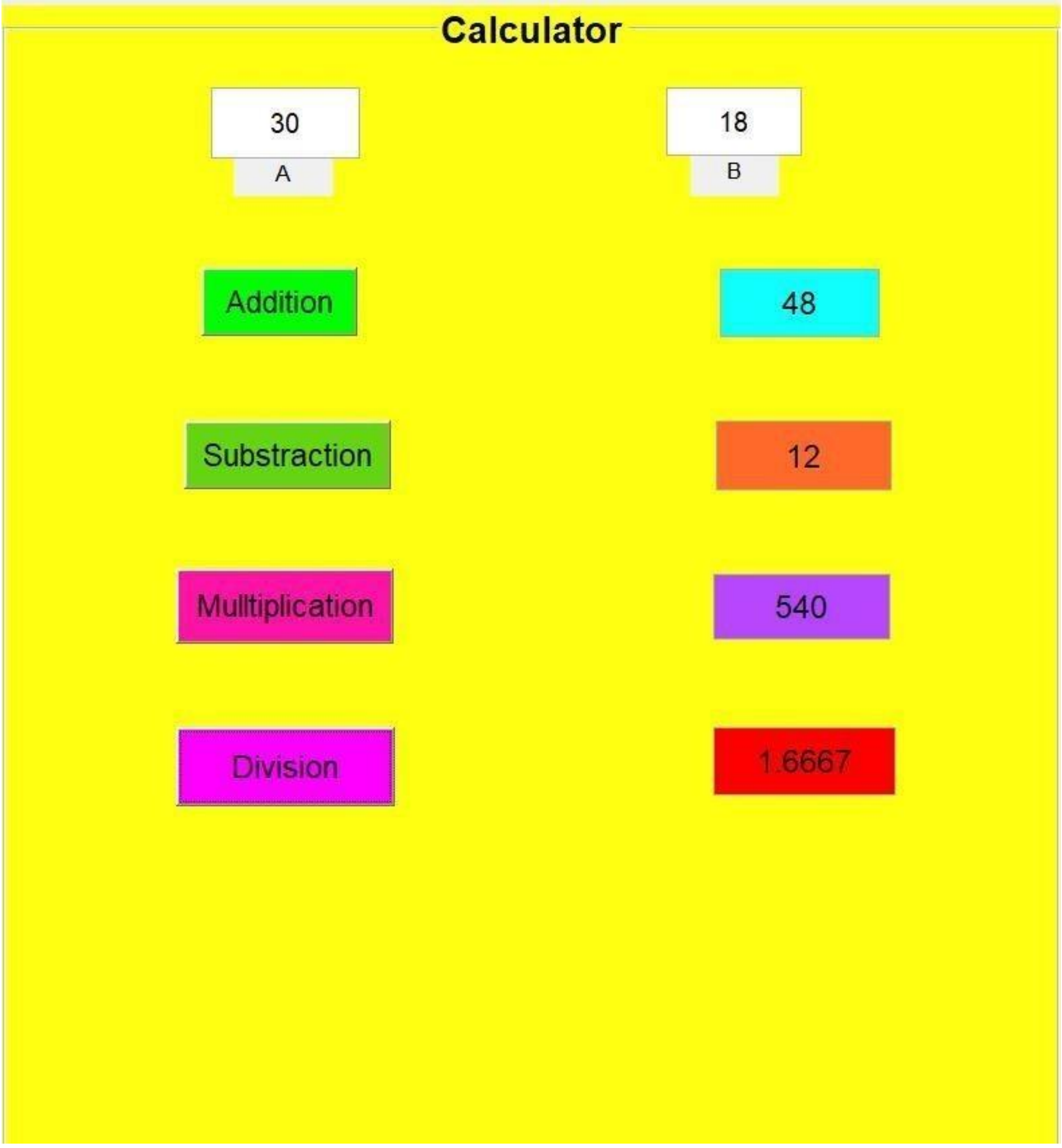
```
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to Calculator_1 (see VARARGIN)

% Choose default command line output for Calculator_1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Calculator_1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line. function
varargout = Calculator_1_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function    edit2_Callback(hObject, eventdata, handles)
% hObject handle to edit2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB %
handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
% str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white'); end



function    edit3_Callback(hObject, eventdata, handles)
% hObject handle to edit3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
% str2double(get(hObject,'String')) returns contents of edit3 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white'); end


% --- Executes on button press in pushbutton1. function
pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
A=str2double(get(handles.edit2,'string'));
B=str2double(get(handles.edit3,'string'));
C=A+B
set(handles.edit4,'string',num2str(C))


function   edit4_Callback(hObject, eventdata, handles)
% hObject handle to edit4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
% str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white'); end


% --- Executes on button press in pushbutton2. function
pushbutton2_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
A=str2double(get(handles.edit2,'string'));
B=str2double(get(handles.edit3,'string'));
D=A-B
set(handles.edit5,'string',num2str(D))

function   edit5_Callback(hObject, eventdata, handles)
```

```
% hObject handle to edit5 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
% str2double(get(hObject,'String')) returns contents of edit5 as a double
% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit5 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white'); end


% --- Executes on button press in pushbutton3. function
pushbutton3_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
A=str2double(get(handles.edit2,'string'));
B=str2double(get(handles.edit3,'string'));
E=A*B
set(handles.edit6,'string',num2str(E))


function   edit6_Callback(hObject, eventdata, handles)
% hObject handle to edit6 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB %
handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
% str2double(get(hObject,'String')) returns contents of edit6 as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit6 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white'); end


% --- Executes on button press in pushbutton4. function
pushbutton4_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
A=str2double(get(handles.edit2,'string'));
B=str2double(get(handles.edit3,'string'));
F=A/B
set(handles.edit7,'string',num2str(F))


function    edit7_Callback(hObject, eventdata, handles)
% hObject handle to edit7 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB %
handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
% str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit7 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white'); end
```

Output GUI of the Calculator:


Calculator
30
A
18
B
Addition
48
Substraction
12
Mulltiplication
540
Division
1.6667