System Dynamics Report:

Abstract:

This is the Final assignment for system Dynamics 2nd Electrical year, the goal of this assignment is computing and visualizing the output of the system for input signal (unit step, unit impulse) and draw the state space of the system by using numerical methods and computes the state space representation matrices (A, B, C, D) of the system.

introduction for the system simulation and model:

the program asks the user to input order of output (n) and order of input (m) and determine the input type unit step or unit impulse and enter values of a's and b's.

In the first gui the program asks the user to enter the data and when the user push the confirm button the second gui opens to plot the input then the third gui opens to plot the output and finally the fourth gui opens to plot the state

description for the numerical approximations:

-we are using Euler's method (first order Runge-Kutta) to solve the differential equation

say we the user enter equation like that:

$$\frac{d^3y(t)}{dt} + a2\frac{d^2y(t)}{dt} + a1\frac{dy(t)}{dt} + a0y(t) = u(t) + b1\frac{du(t)}{dt} \quad where\ u(t)\ is\ unit\ step$$

First, we will divide the equation by a(n) to make sure that, the coefficient of $\frac{d^ny(t)}{dt}$ will be zero
Second, we are dealing with the derivatives of the input, In our code we represent unit step as a vector [..0 0 0 0 1 1 1 1…] and unit impulse as a vector [..0 0 0 0 (1/h) 0 0 0…] where h is the step size, and based on the input type which user input we generate the input vector and get the derivatives of this vector so if m = 1 we get $\frac{du(t)}{dt}$ and if m=2 we get $\frac{du(t)}{dt}$ and $\frac{d^2u(t)}{dt}$, by multiply them by their coefficient and adding the result we get the input vector like in the equation above the input will be equal $u(t) + b1\frac{du(t)}{dt}$ So the equation will be:

$$\frac{d^3y(t)}{dt} + a2\frac{d^2y(t)}{dt} + a1\frac{dy(t)}{dt} + a0y(t) = \gamma(t) = input\ vector$$

We set the initial conditions for $\frac{d^2y(t)}{dt}, \frac{dy(t)}{dt}, y(t)$ to be equal to zero and we introduce three new variables x1, x2, x3:

$x1(t) = y(t)$, $x2(t) = \frac{dy(t)}{dt}$, $x3(t) = \frac{d^2y(t)}{dt}$

$x1'(t) = y'(t) = x2(t)$

$x2'(t) = y''(t) = x3(t)$

x3′(t)= y′′′(t) = γ(t)−a2y′′(t)−a2y′(t)−a0y(t) = γ(t)−a2*x3(t)−a1*x2(t)−a0*x1(t)

So we can use the state space matrices A and B so the equation will be

x′(t)=A*x(t)+B*γ(t)

$$\mathbf{q}'(t) = \begin{bmatrix} q_1'(t) \\ q_2'(t) \\ q_3'(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_2 & -a_1 & -a_0 \end{bmatrix} \begin{bmatrix} q_1(t) \\ q_2(t) \\ q_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \gamma(t)$$

$\mathbf{x}'(t_0) = \mathbf{A}\mathbf{x}(t_0) + \mathbf{B}\gamma(t_0)$ exact expression for derivative at t=t0

$\mathbf{k}_1 = \mathbf{A}\mathbf{x}^*(t_0) + \mathbf{B}\gamma(t_0)$ approximation for derivative

$\mathbf{x}(t_0 + h) = \mathbf{x}(t_0) + \mathbf{x}'(t_0)h + \mathbf{x}''(t_0)\frac{h^2}{2} + \cdots$ Taylor Series around t=t0

$\mathbf{x}(t_0 + h) \approx \mathbf{x}(t_0) + \mathbf{x}'(t_0)h$ Truncated Taylor Series

$\mathbf{x}^*(t_0 + \hbar) = \mathbf{x}^*(t_0') + \mathbf{k}_1 h$ Approximate Solution

By using for loop we can update the value of k1 and $\mathbf{x}^*$

-to get derivative of a vector we use differentiation formula where h= 0.001

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

-To get state space representation matrices we use Controllable Canonical Form (CCF) :

$$A = \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ -a_0 & -a_1 & \cdots & -a_{n-2} & -a_{n-1} \end{bmatrix} \quad \mathrm{B} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{c} = [(\mathrm{b}_0 - \mathrm{a}_0\mathrm{b}_n)(\mathrm{b}_1 - \mathrm{a}_1\mathrm{b}_n) \ldots \ldots \ldots. \ (\mathrm{b}_{n-1} - \mathrm{a}_{n-1}\mathrm{b}_n)] \quad \mathrm{D} = \mathrm{b}_n$$

Before we use this formula we need to be sure that a(n) will be equal to one.
This will be okay if (n=m) But if m is less than n we just put any missing b to be equal to zero.

simulation algorithms for simulating for every order:

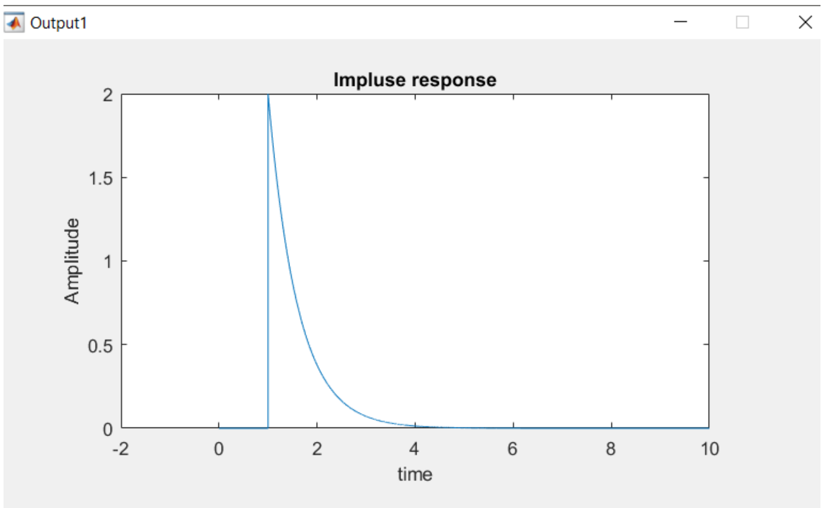
we use the algorithm above to solve the differential equation for any order n by using step size h=0.001 and time interval between 0-h to 10

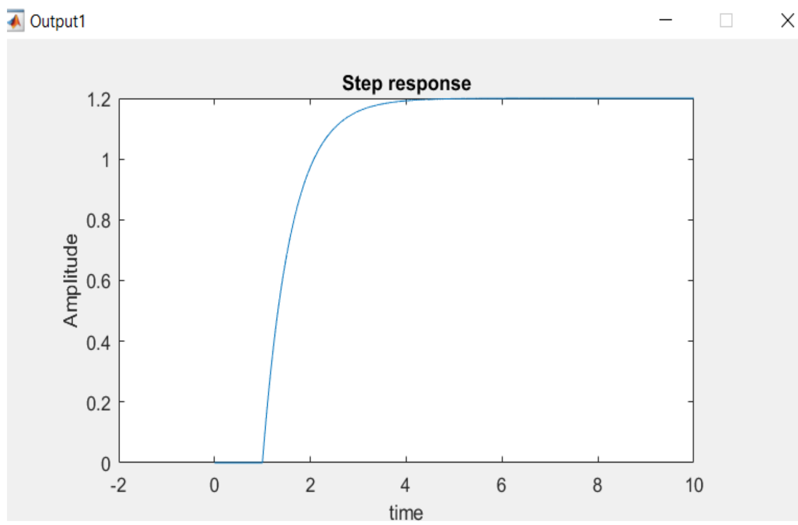
Experimental results section:

In these equations we refer to u(t) as the input (unit step or unit impulse):

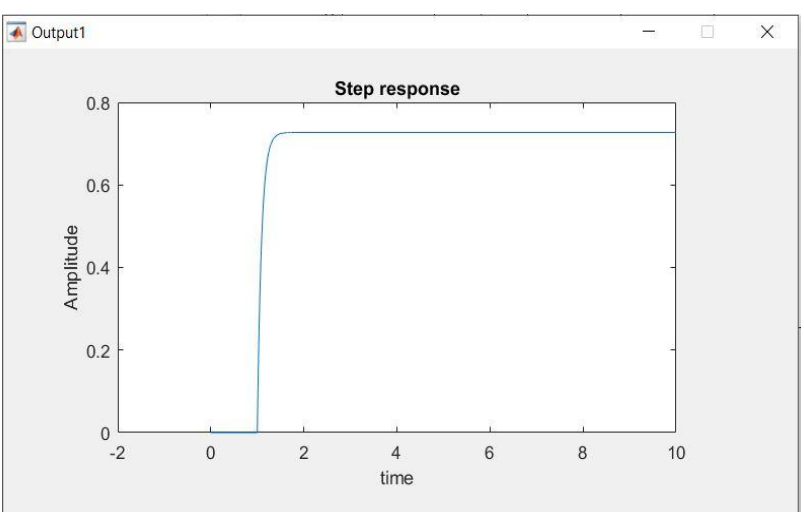
First Order:

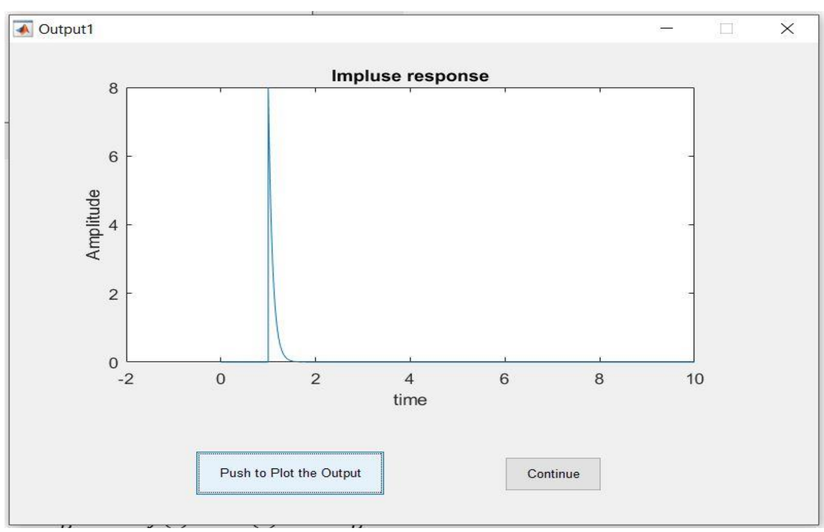
$$3\frac{dy(t)}{dt} + 5y(t) = 6u(t)$$





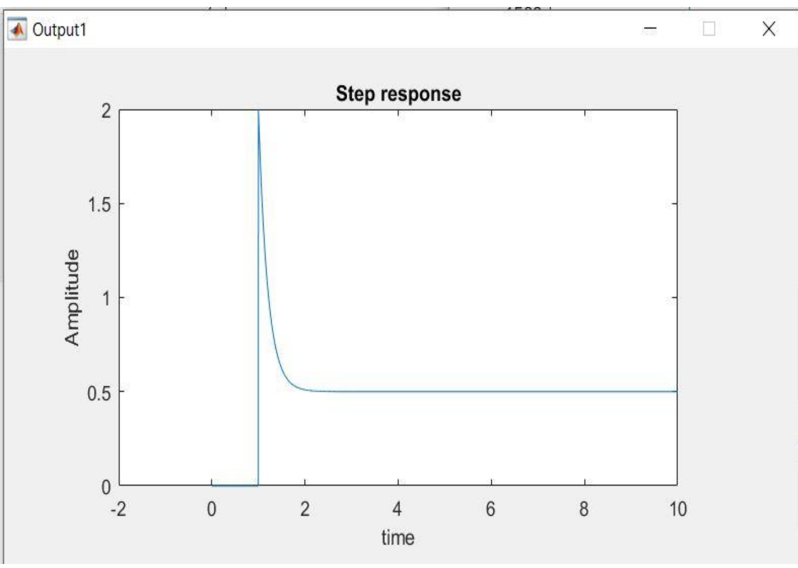
$$\frac{dy(t)}{dt} + 11y(t) = 8u(t)$$

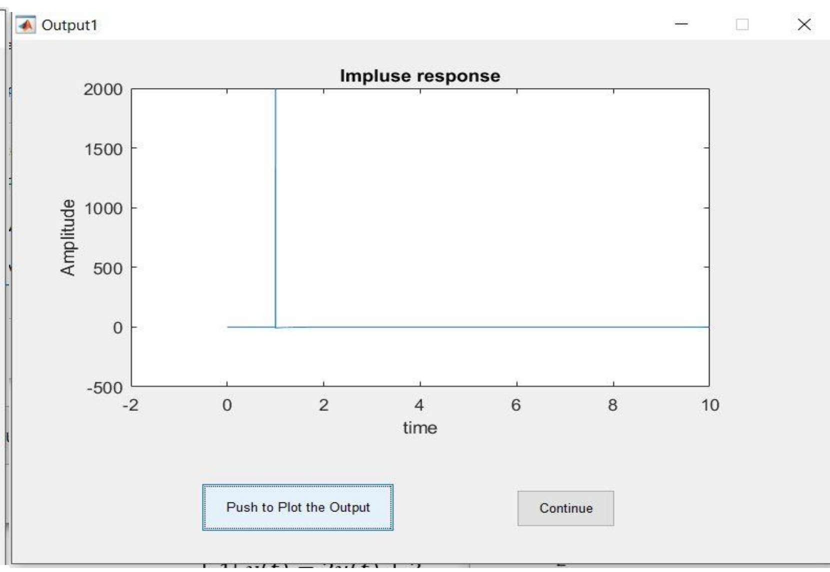




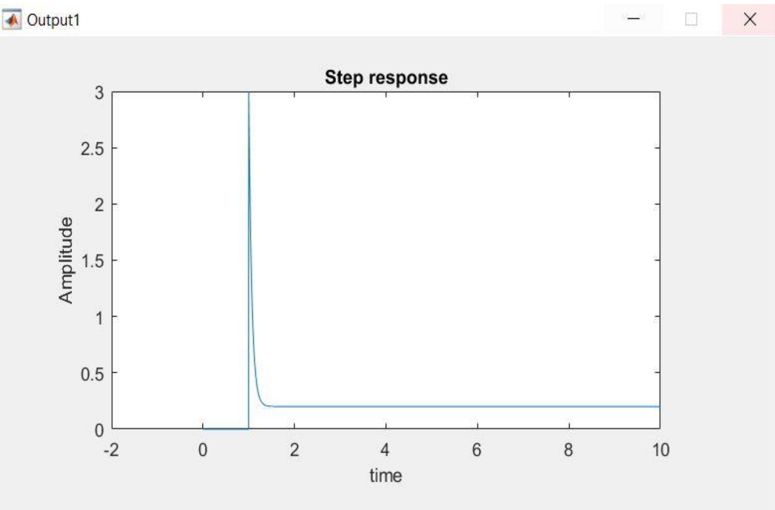
```
A =

   -11


B =

     1


C =

     8


D =

     0

fx >> 
```

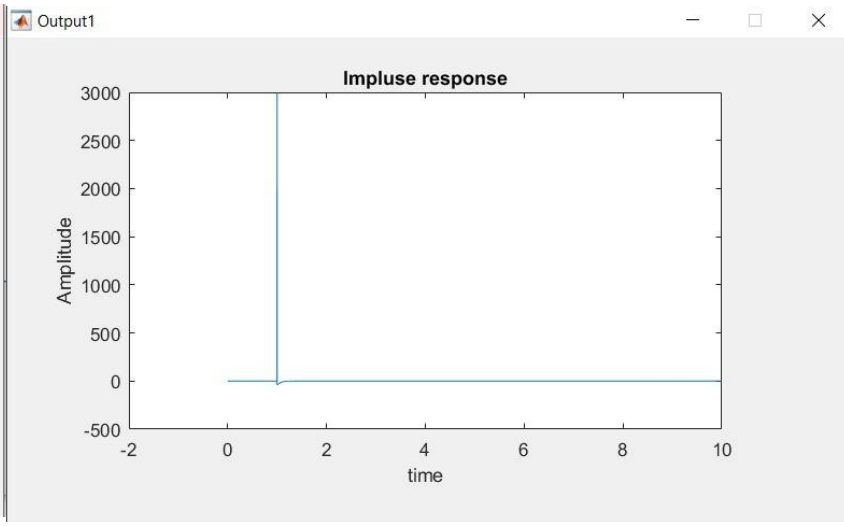
$$2\frac{dy(t)}{dt} + 10y(t) = 5u(t) + 4\frac{du(t)}{dt}$$





$$\frac{dy(t)}{dt} + 15y(t) = 3u(t) + 3\frac{du(t)}{dt}$$

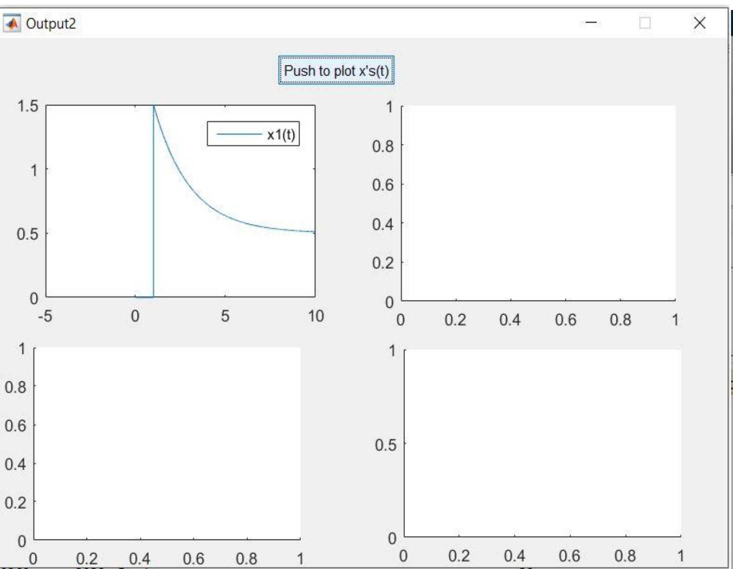


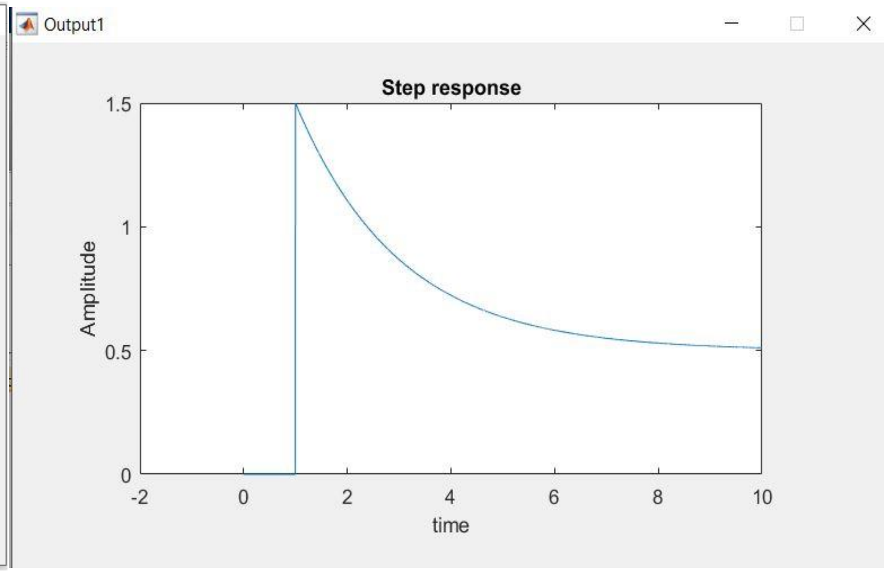


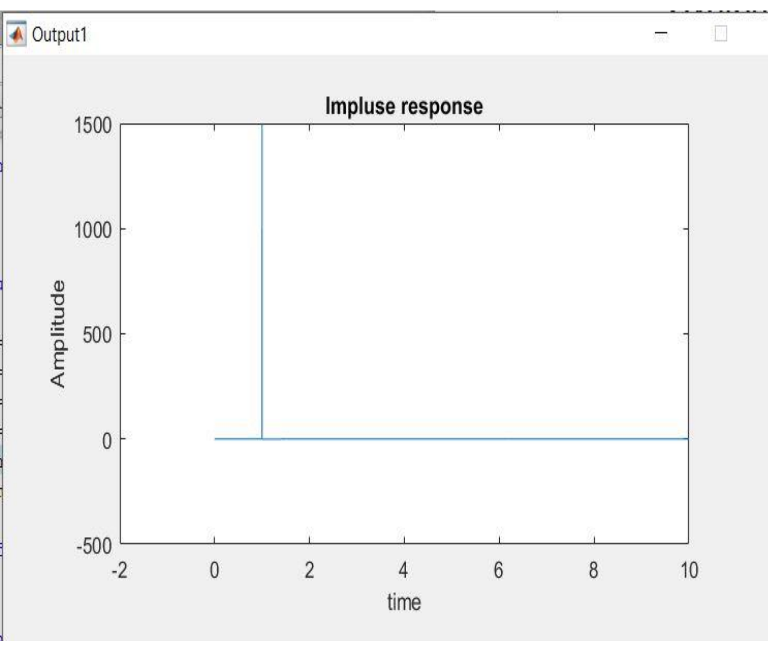
Command Window

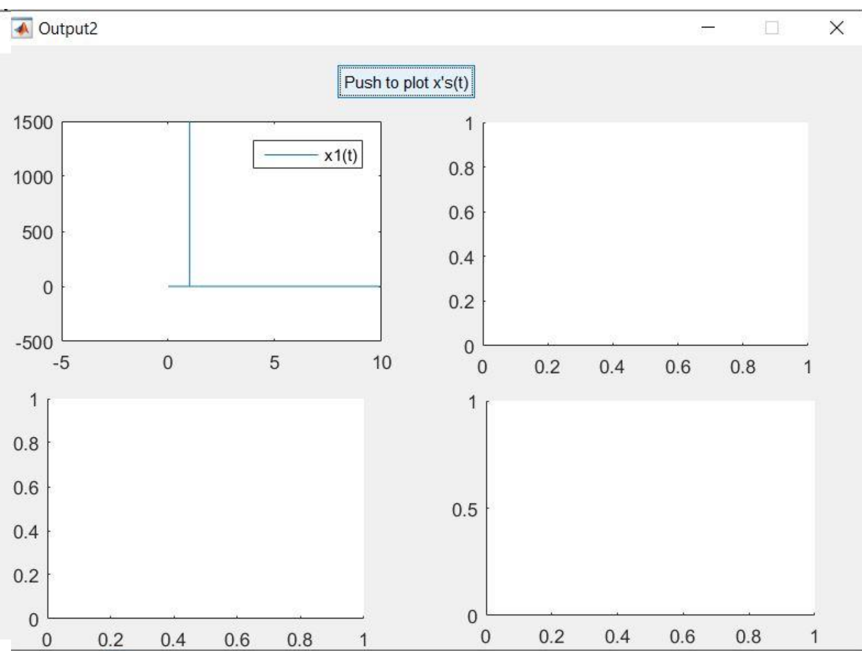
```
A =

   -15


B =

     1


C =

   -42


D =

     3
```

$$4\frac{dy(t)}{dt} + 2y(t) = u(t) + 6\frac{du(t)}{dt}$$



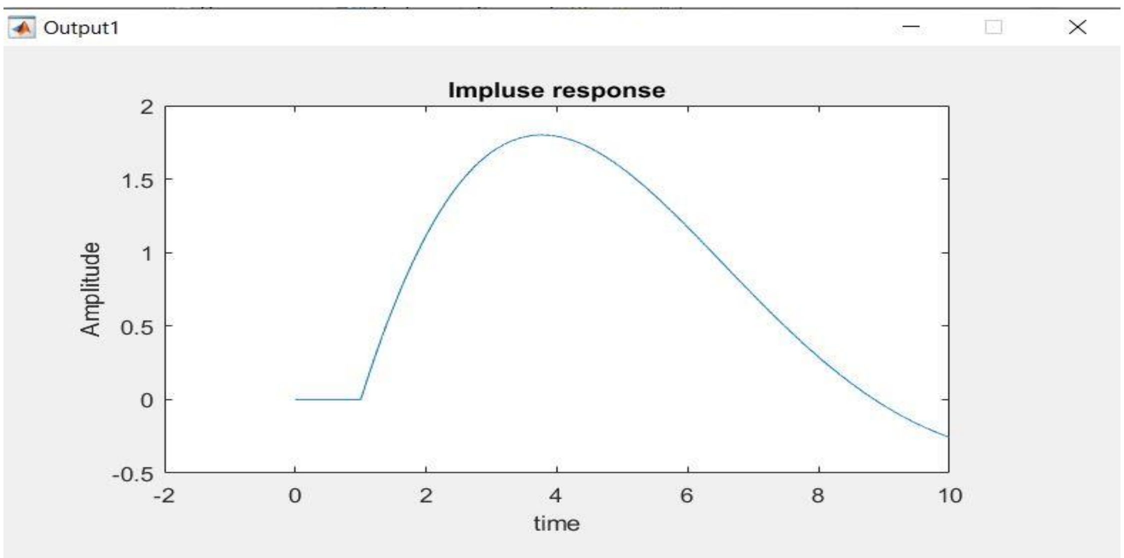


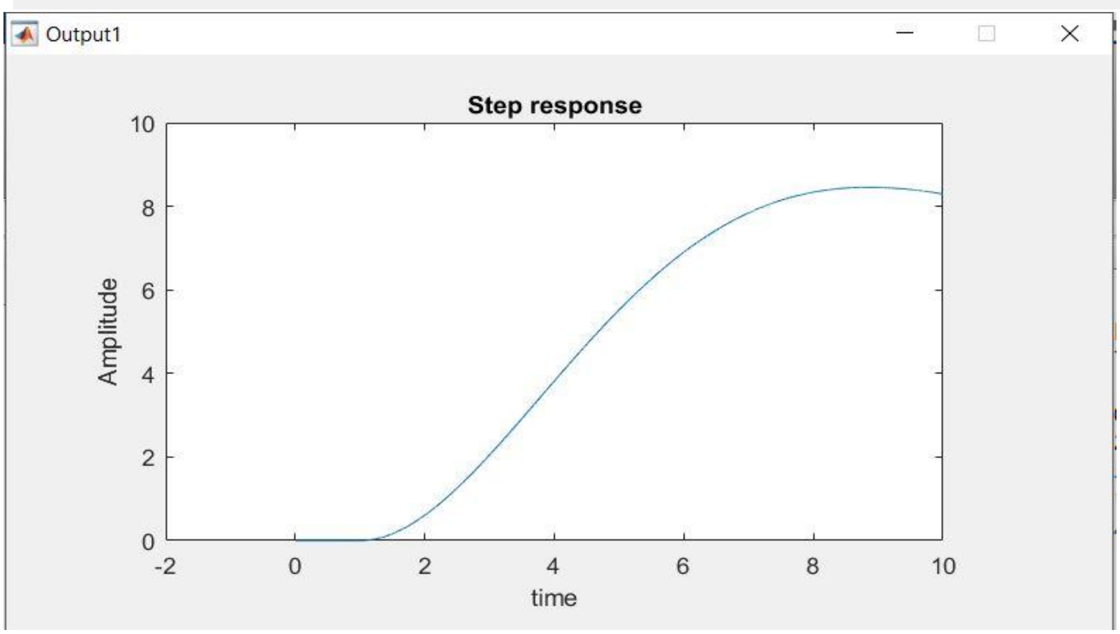




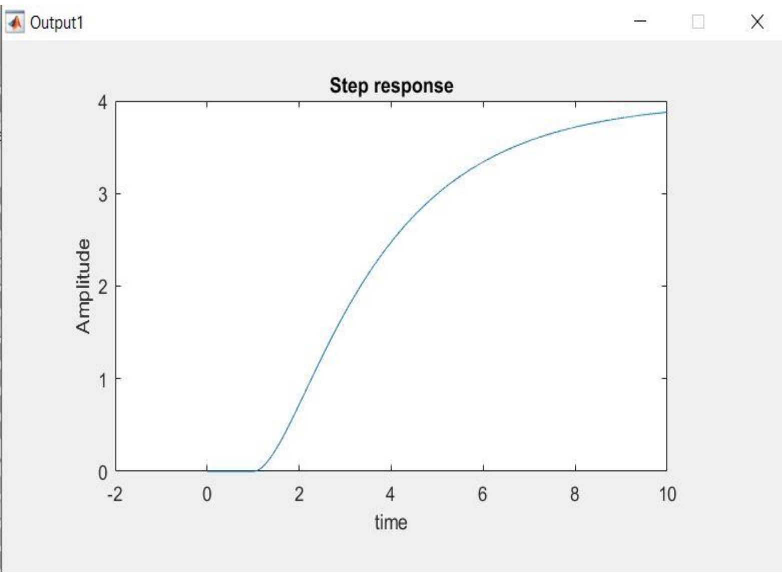
Second order:

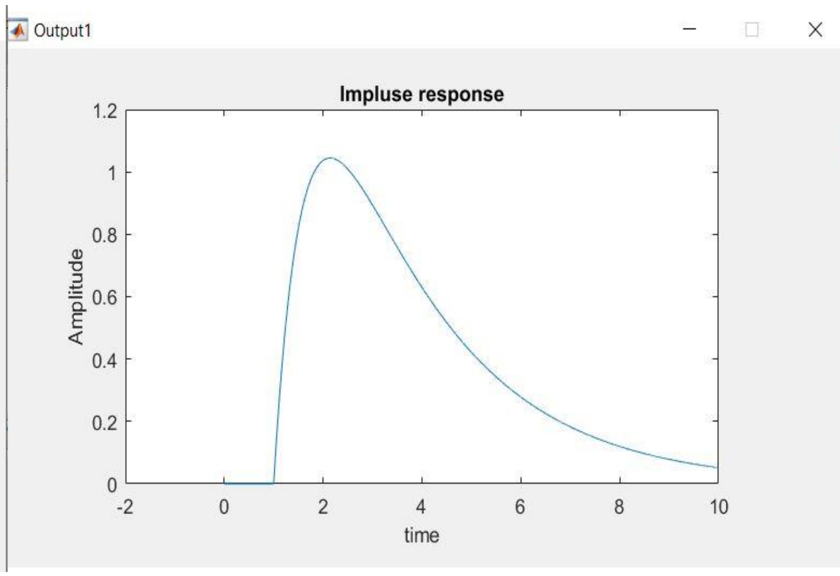
$$5\frac{d^2y(t)}{dt} + 2\frac{dy(t)}{dt} + y(t) = 7\,u(t)$$





$$3\frac{d^2y(t)}{dt}+6\frac{dy(t)}{dt}+2y(t)=8\ \mathrm{u(t)}$$

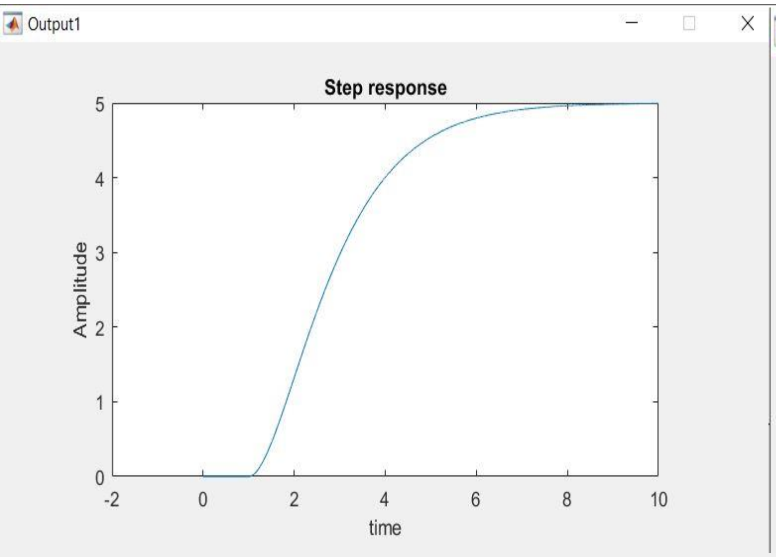


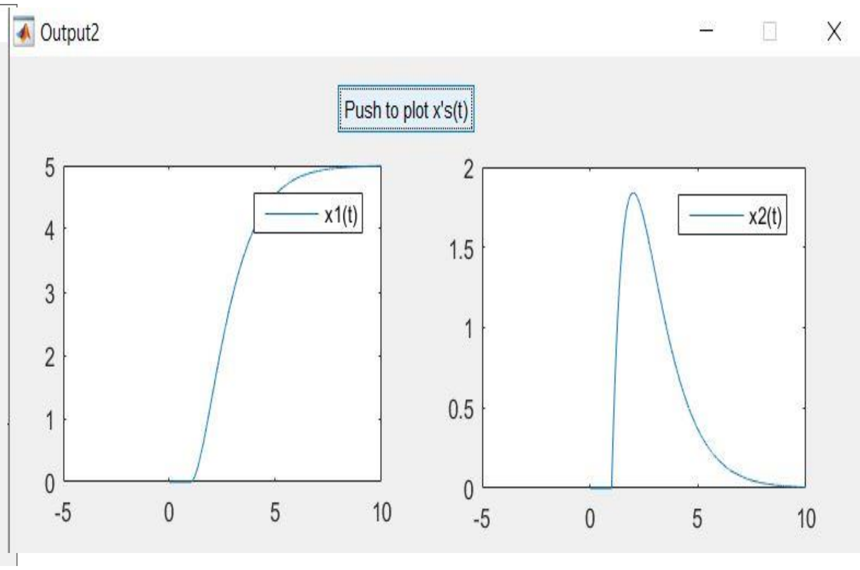


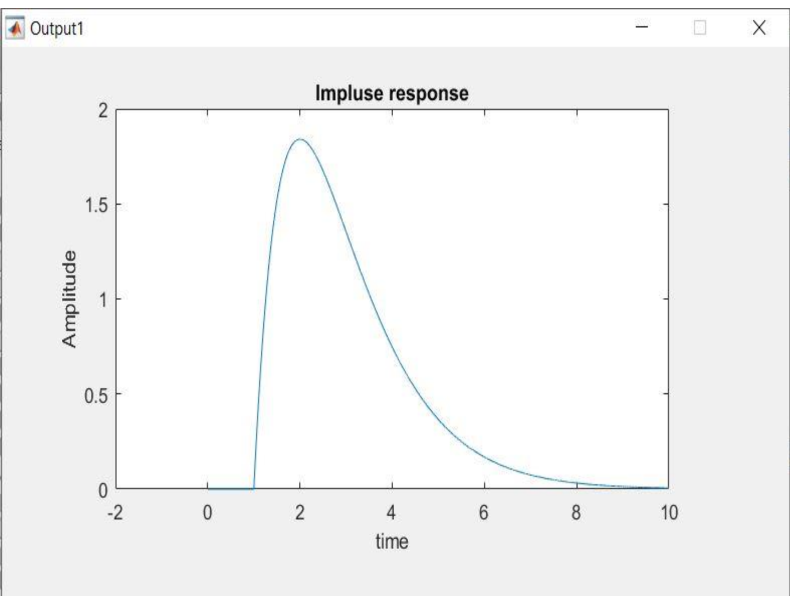
Command Window

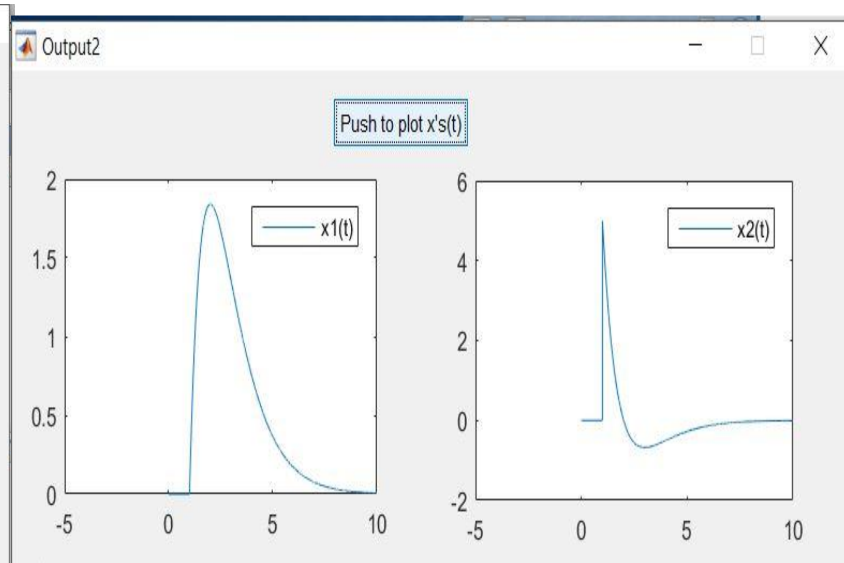
```
A =

         0    1.0000
   -0.6667   -2.0000


B =

     0
     1


C =

    2.6667         0


D =

     0
```

$$\frac{d^2y(t)}{dt} + 2\frac{dy(t)}{dt} + y(t) = 5\ \mathrm{u(t)}$$

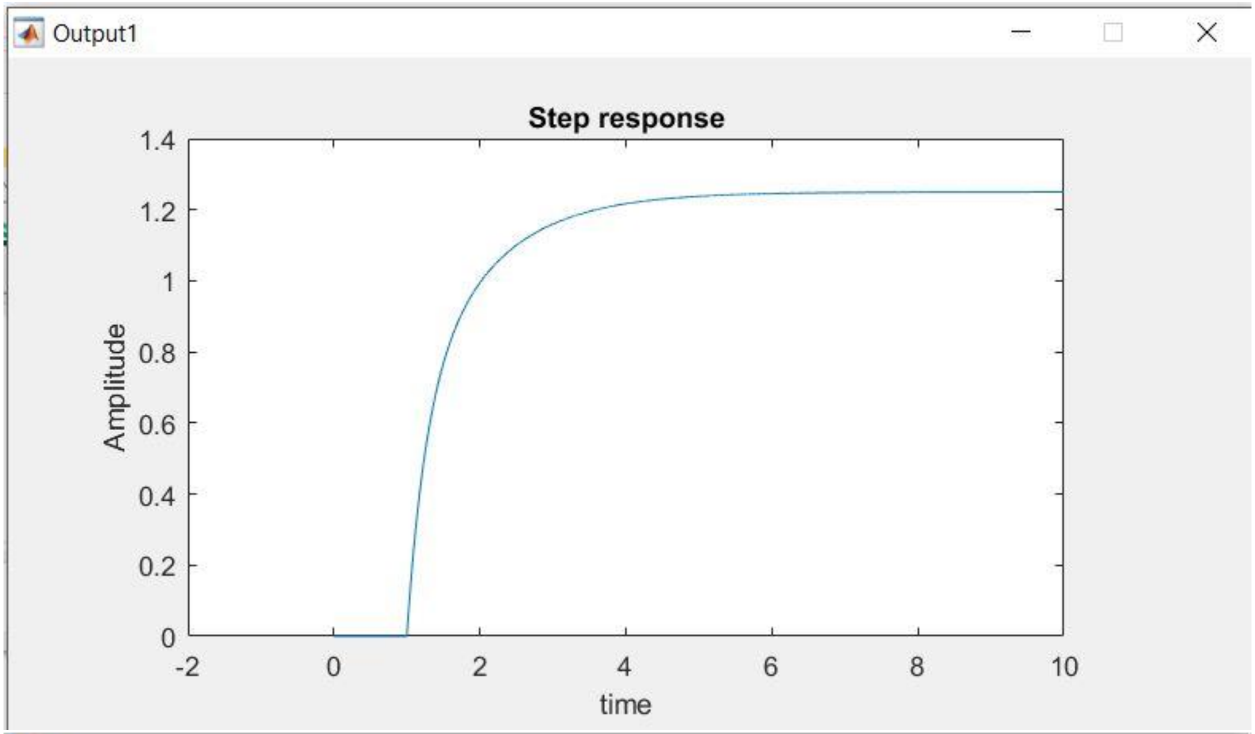


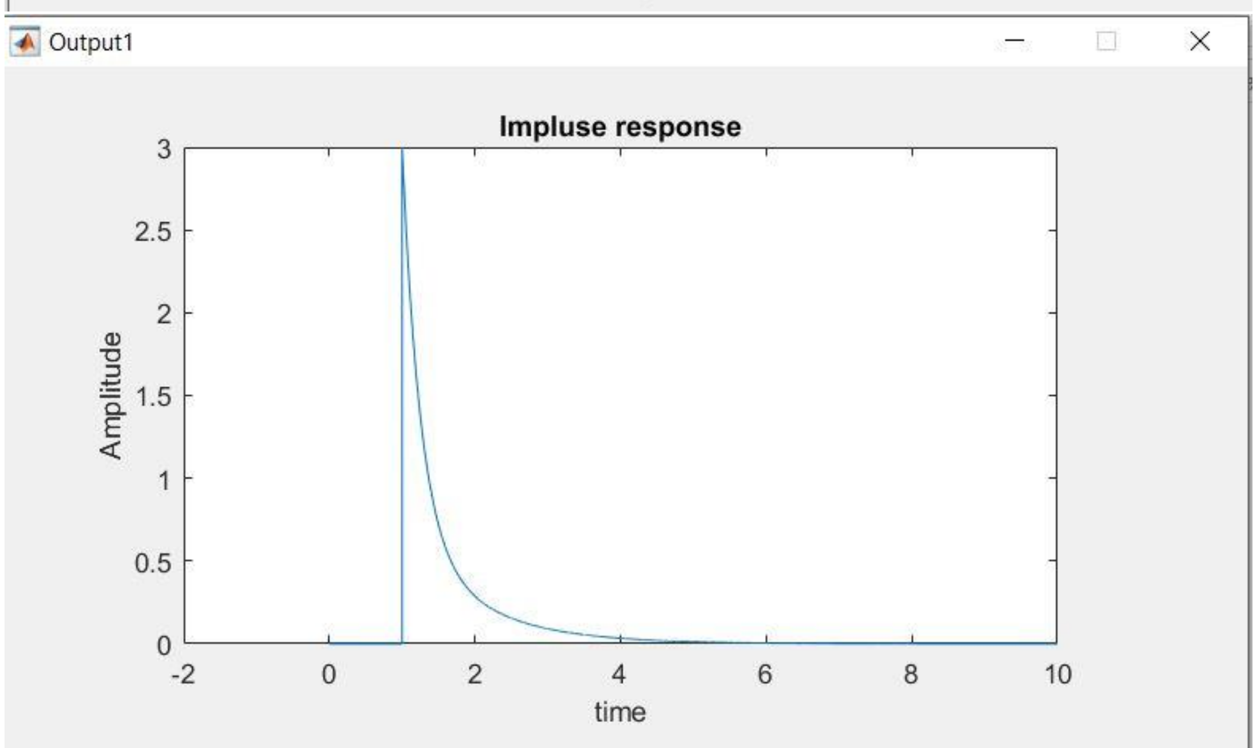




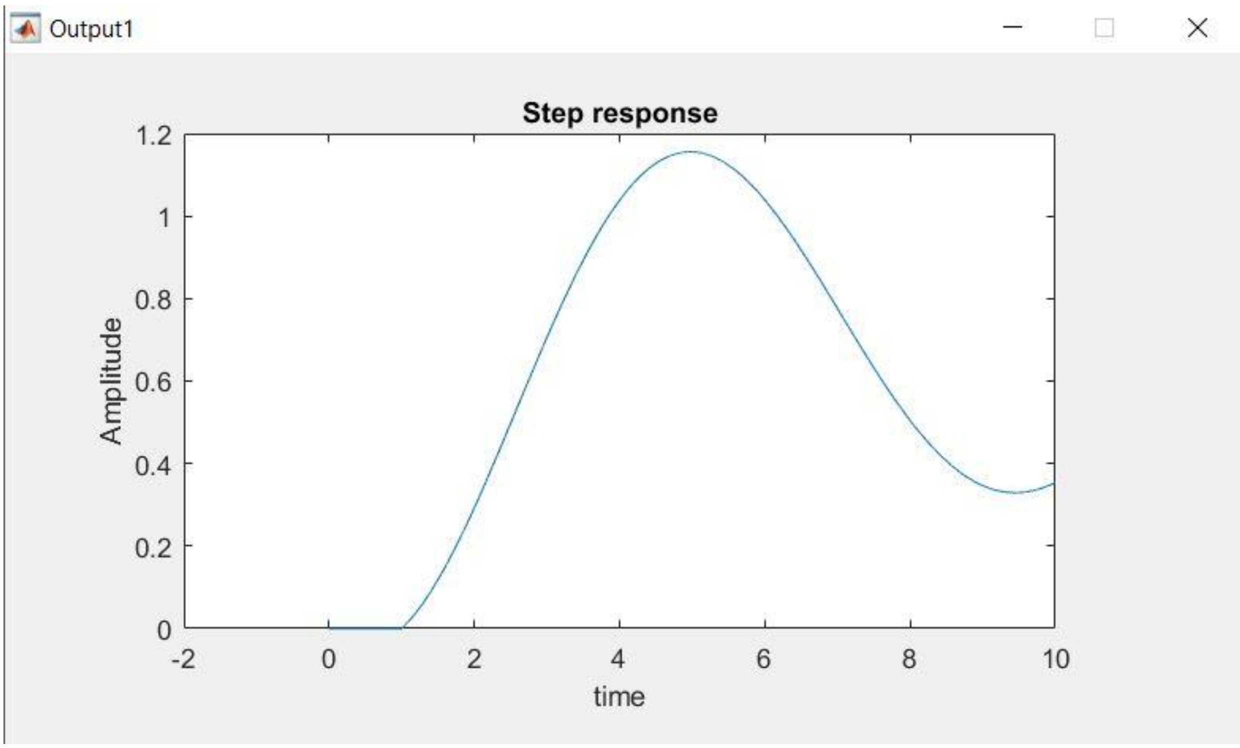


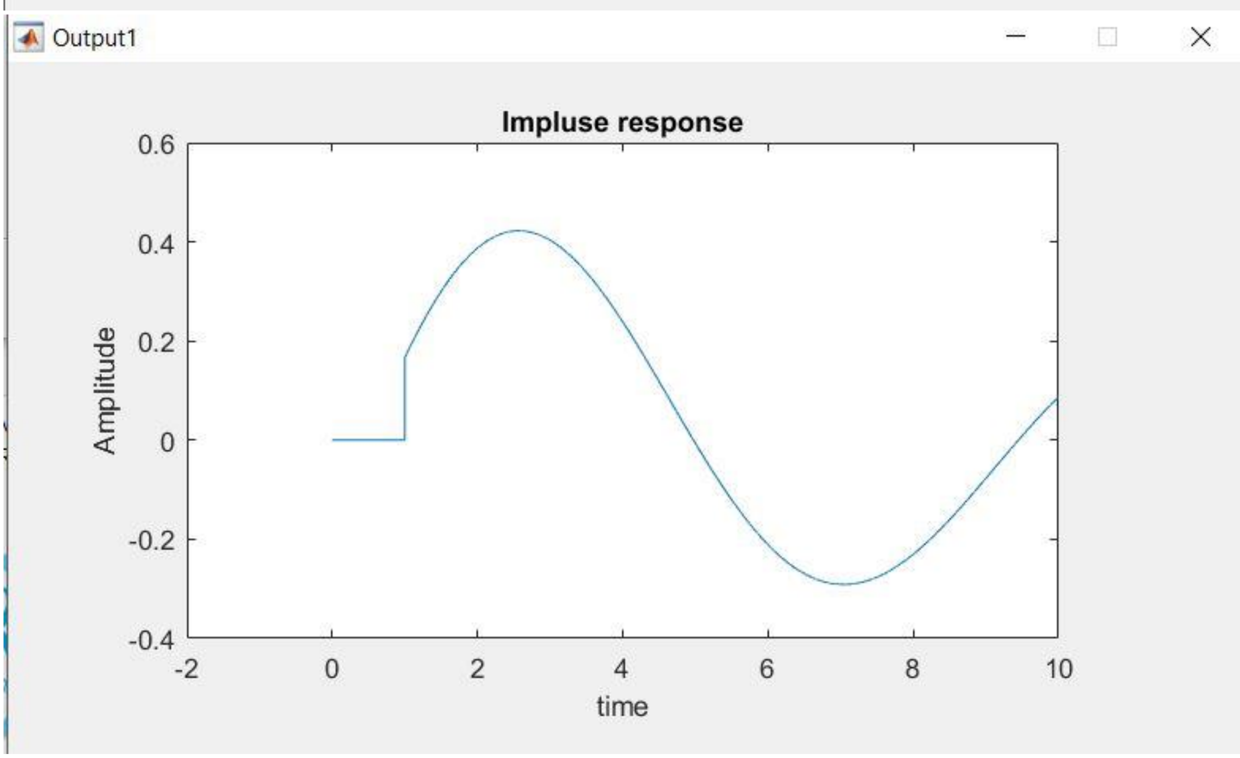
$$\frac{d^2y(t)}{dt} + 5\frac{dy(t)}{dt} + 4y(t) = 5\,\text{u(t)} + 3\frac{du(t)}{dt}$$



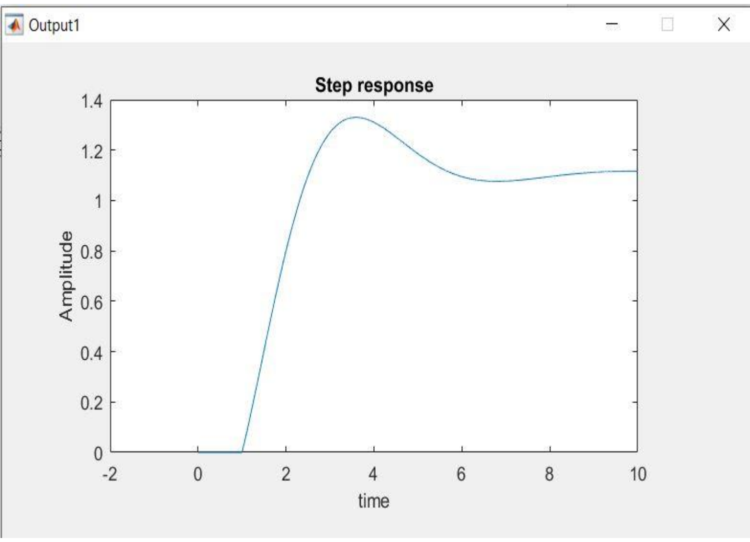


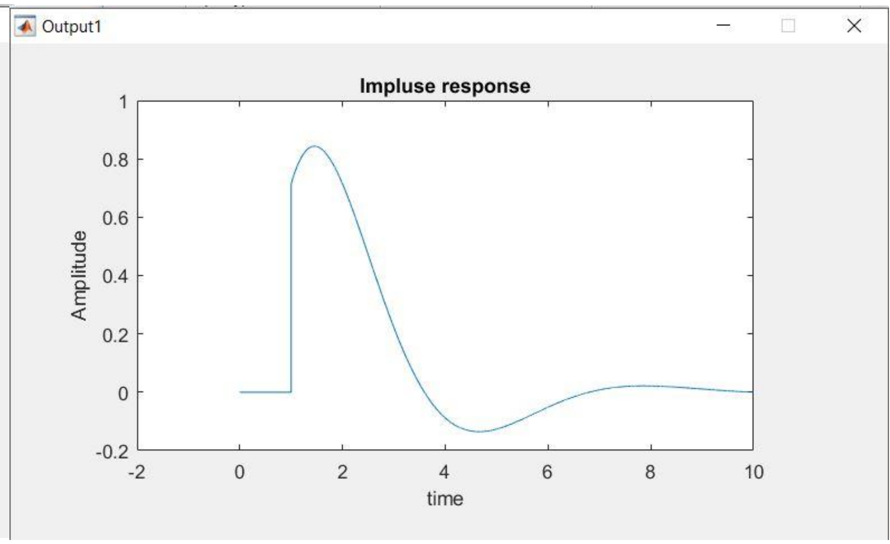
$$6\frac{d^2y(t)}{dt}+\frac{dy(t)}{dt}+3y(t)=2\ \mathrm{u(t)}+\frac{du(t)}{dt}$$





$$7\frac{d^2y(t)}{dt} + 8\frac{dy(t)}{dt} + 9y(t) = 10\ \mathrm{u(t)} + 5\frac{du(t)}{dt}$$

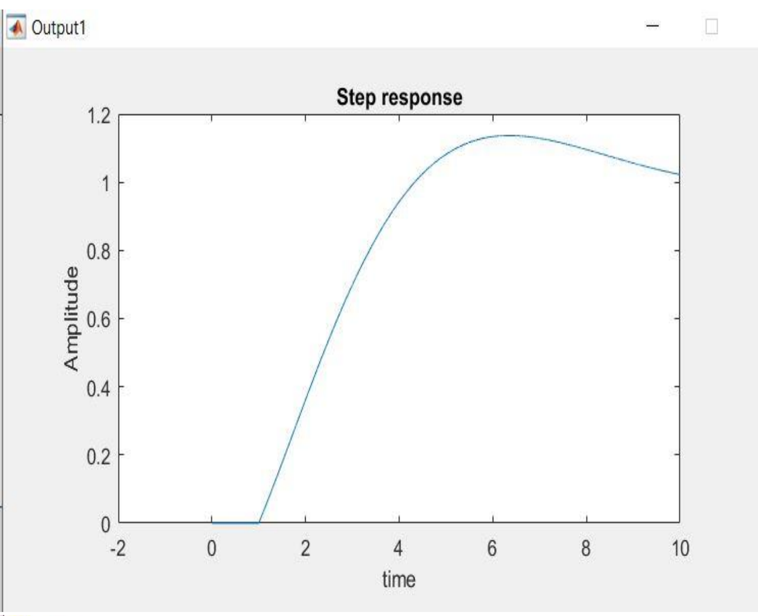


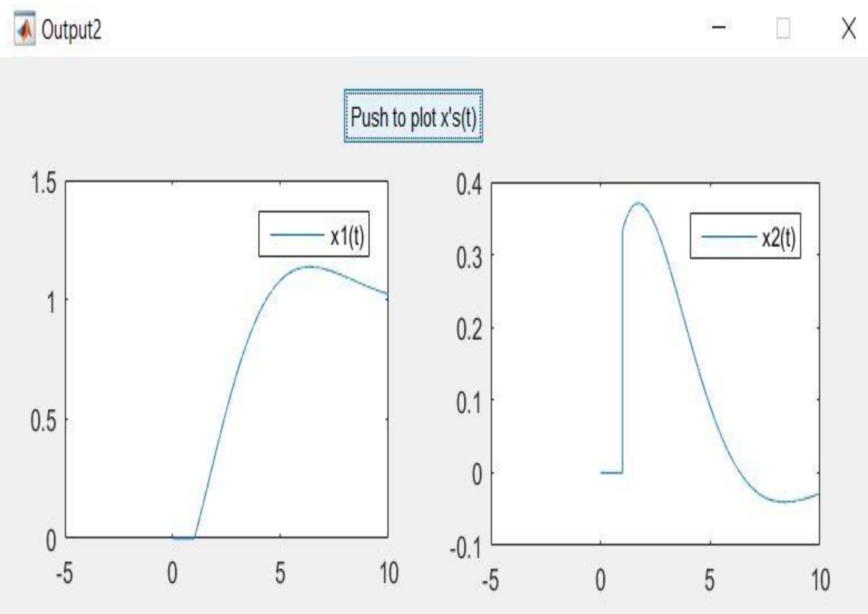


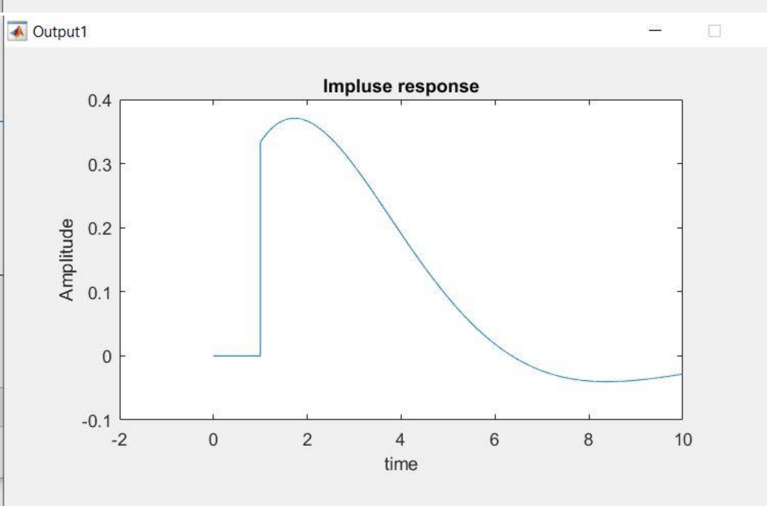
Command Window

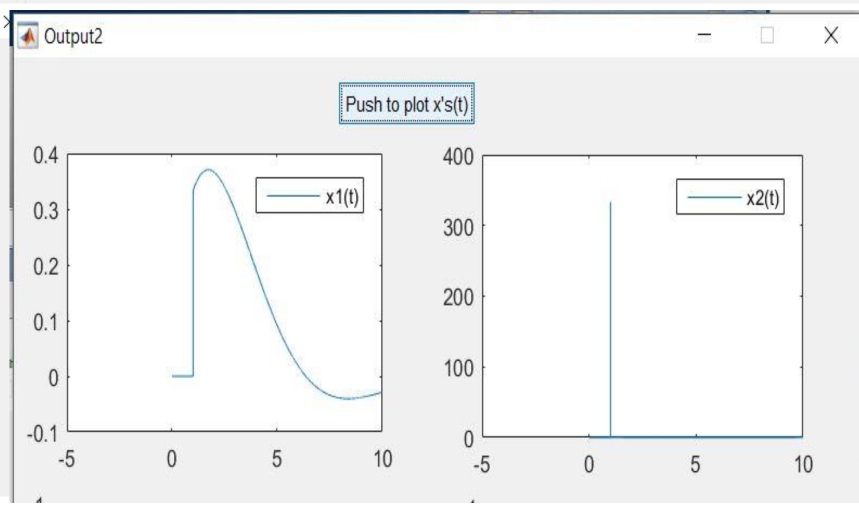
```
A =

         0    1.0000
   -1.2857   -1.1429


B =

     0
     1


C =

    1.4286    0.7143


D =

     0
```

$$3\frac{d^2y(t)}{dt} + 2\frac{dy(t)}{dt} + y(t) = \mathrm{u(t)} + \frac{du(t)}{dt}$$

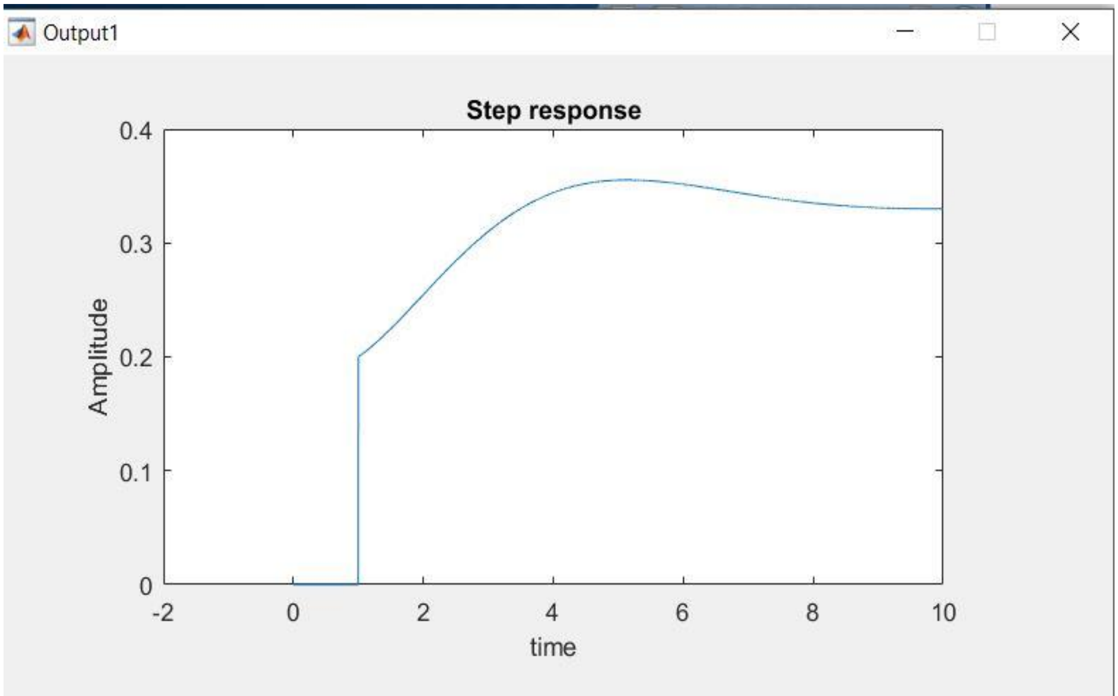


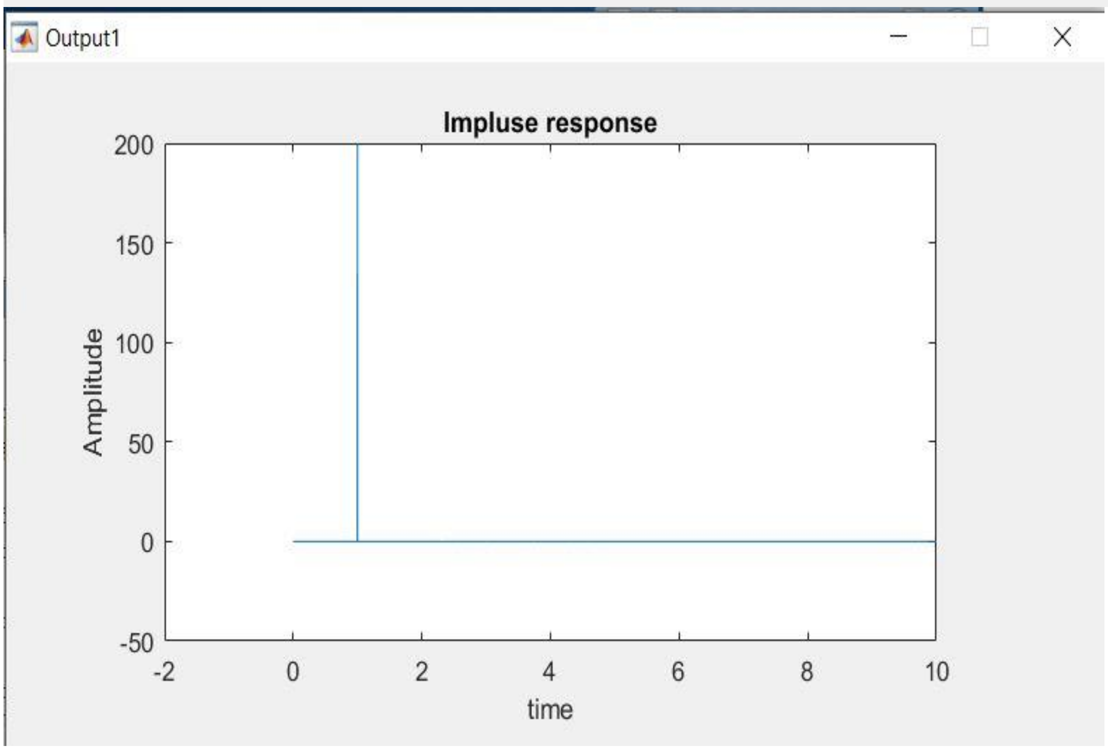




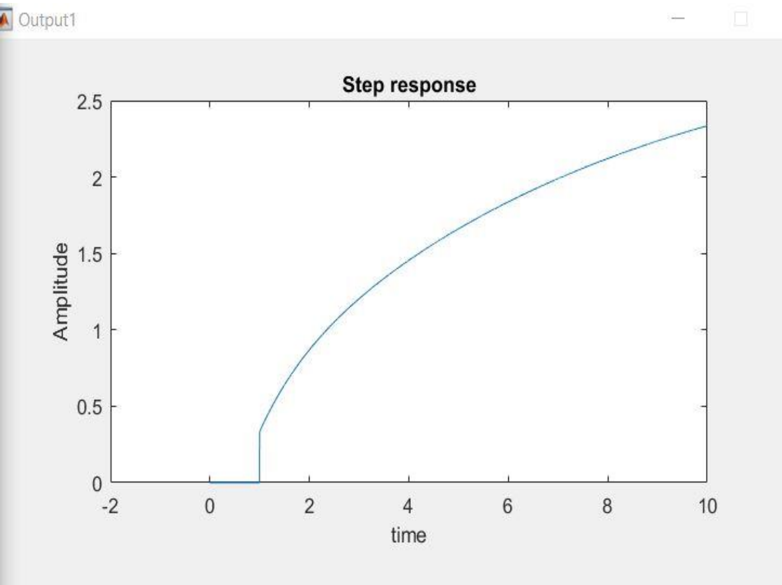


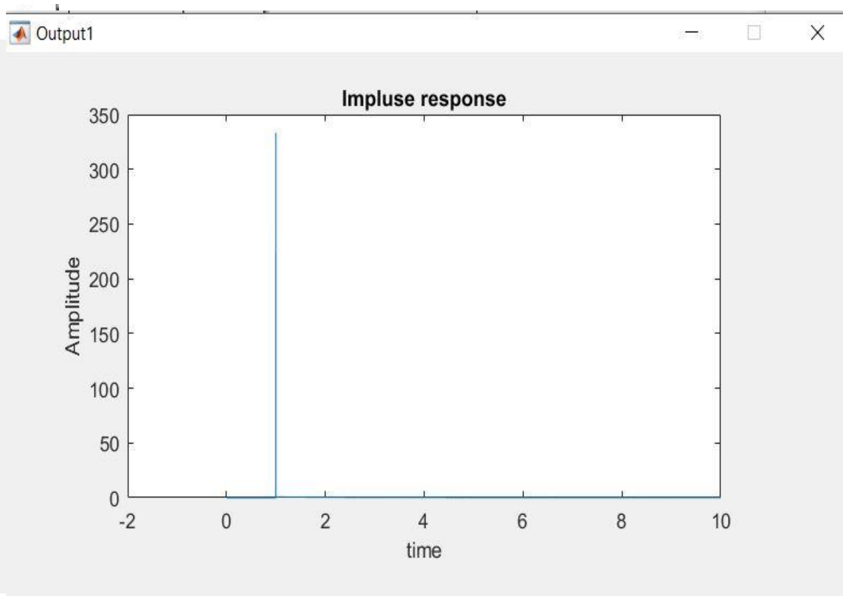
$$5\frac{d^2y(t)}{dt} + 4\frac{dy(t)}{dt} + 3y(t) = \mathrm{u(t)} + \frac{du(t)}{dt} + \frac{d^2u(t)}{dt}$$





$$6\frac{d^2y(t)}{dt} + 8\frac{dy(t)}{dt} + y(t) = 3\mathrm{u(t)} + 7\frac{du(t)}{dt} + 2\ \frac{d^2u(t)}{dt}$$

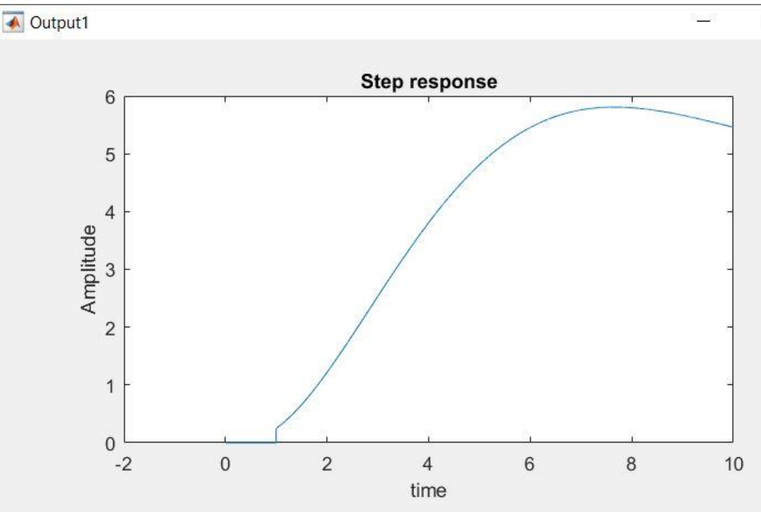


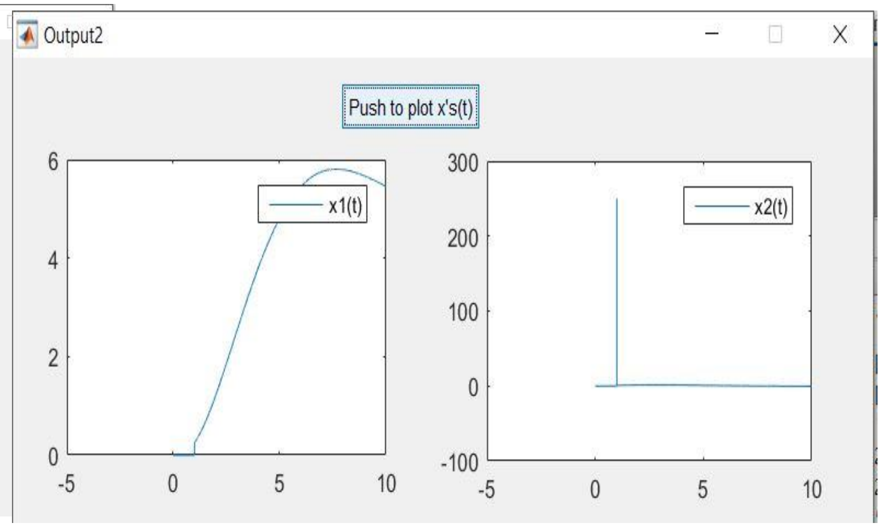


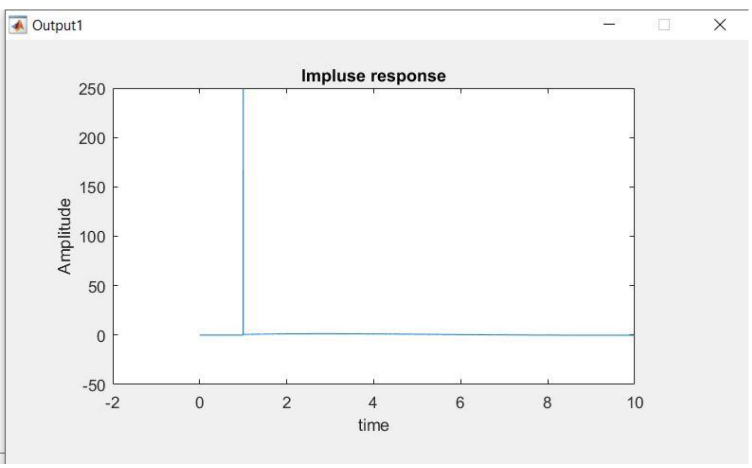
Command Window

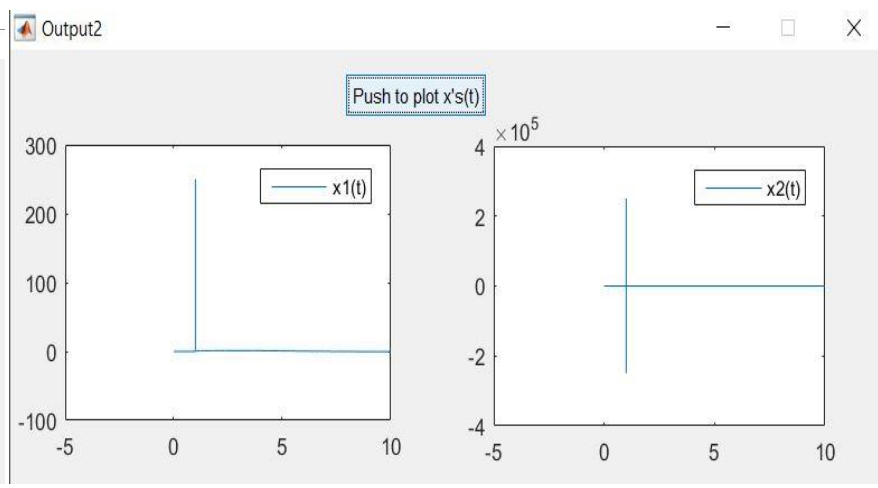
```
A =

        0    1.0000
  -0.1667   -1.3333


B =

     0
     1


C =

    0.4444    0.7222


D =

    0.3333
```

$$4\frac{d^2y(t)}{dt} + 2\frac{dy(t)}{dt} + y(t) = 5\mathrm{u(t)} + 3\frac{du(t)}{dt} + \frac{d^2u(t)}{dt}$$



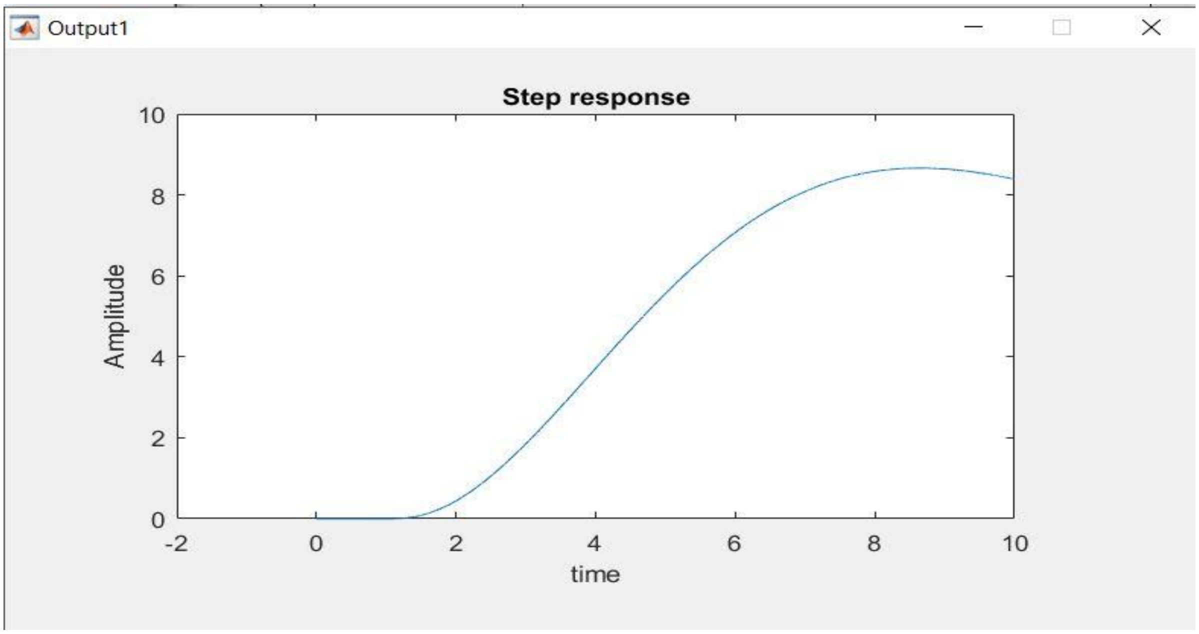


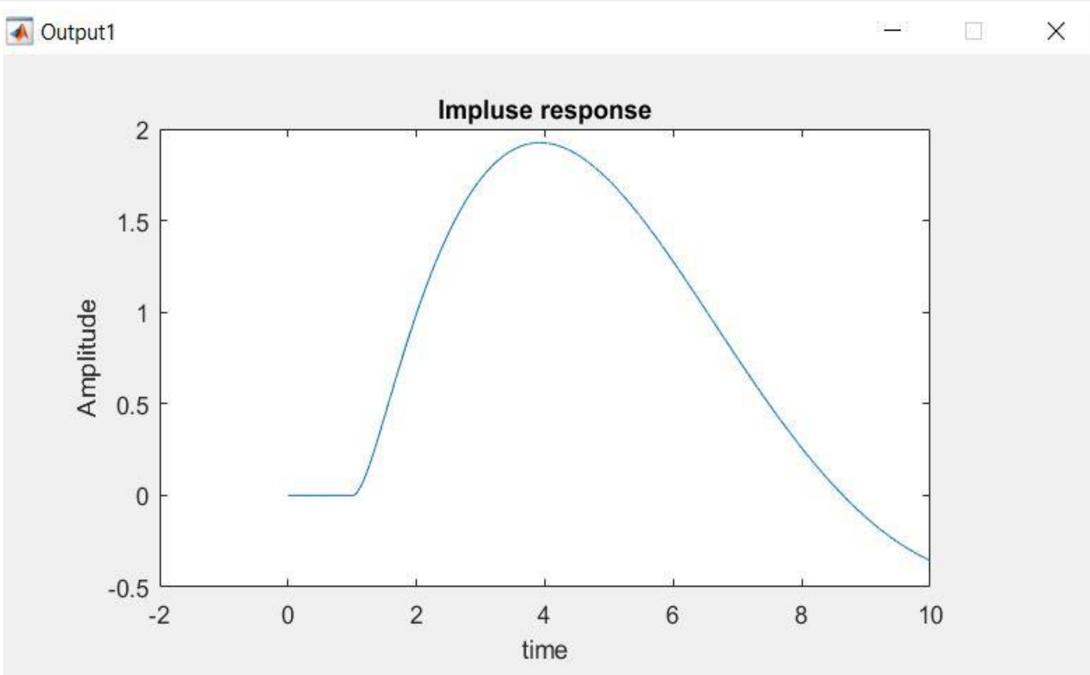




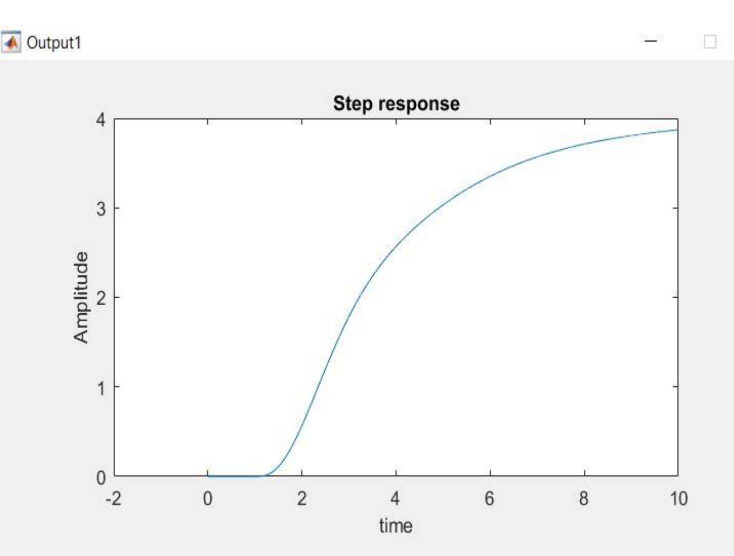
Third Order:

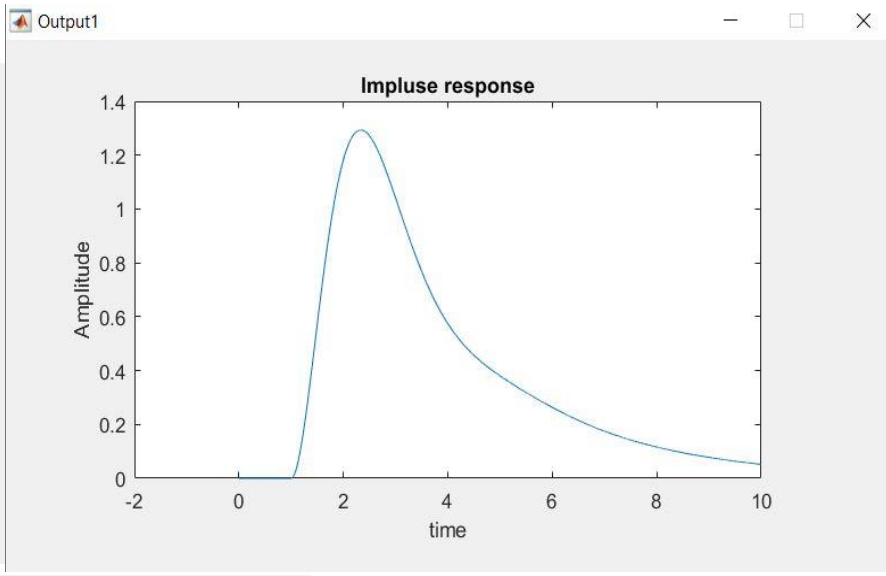
$$\frac{d^3y(t)}{dt}+5\,\frac{d^2y(t)}{dt}+2\,\frac{dy(t)}{dt}+y(t)=7\text{u(t)}$$





$$\frac{d^3y(t)}{dt} + 3\,\frac{d^2y(t)}{dt} + 6\,\frac{dy(t)}{dt} + 2y(t) = 8\mathrm{u(t)}$$

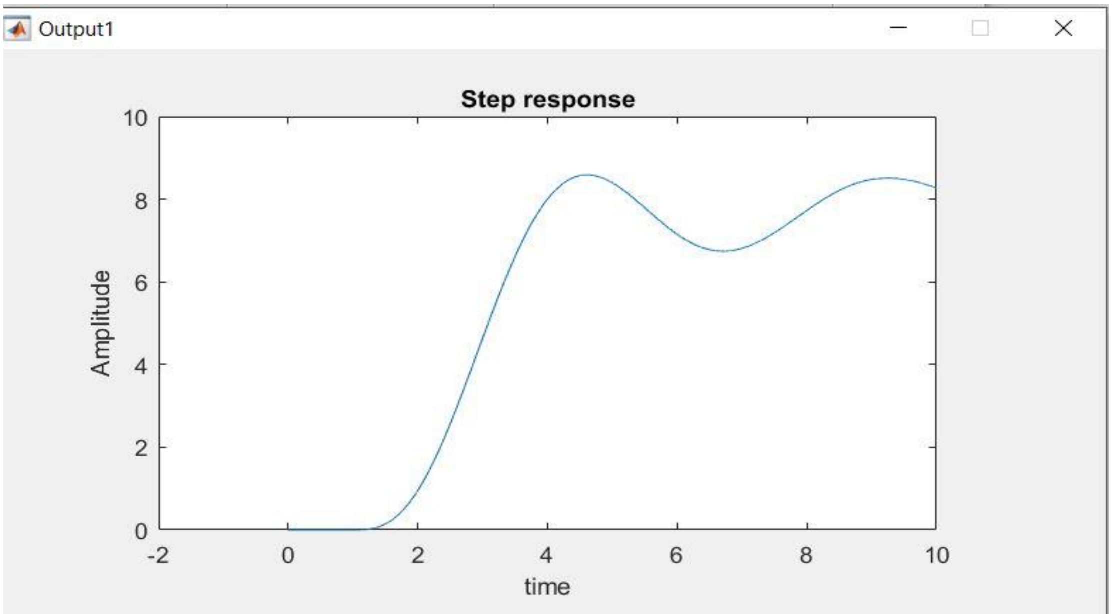


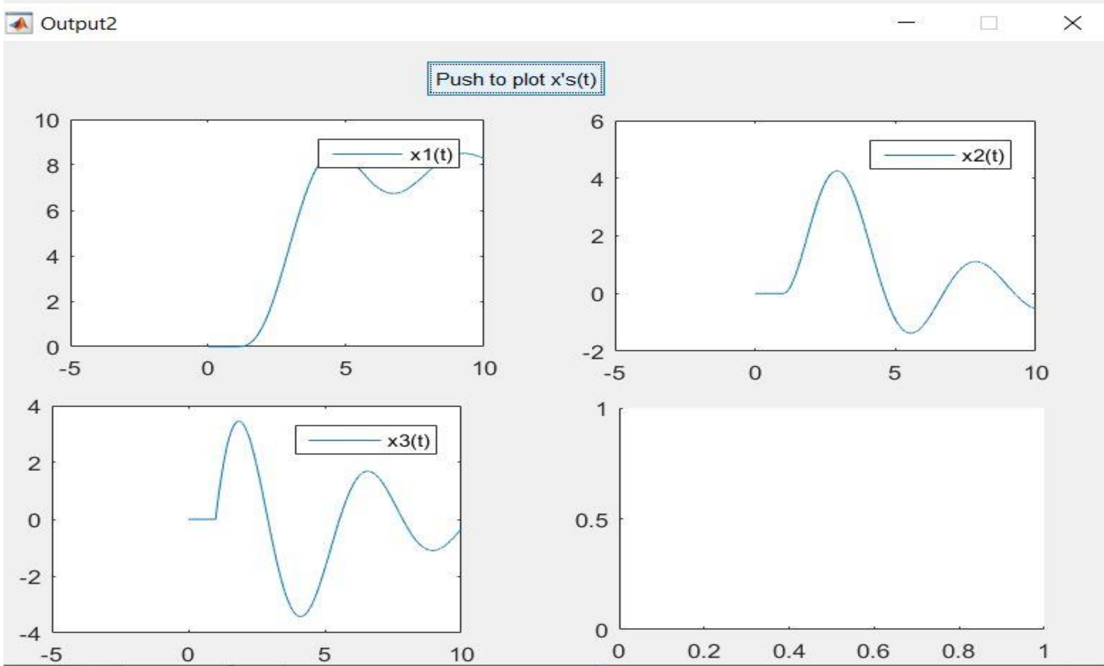


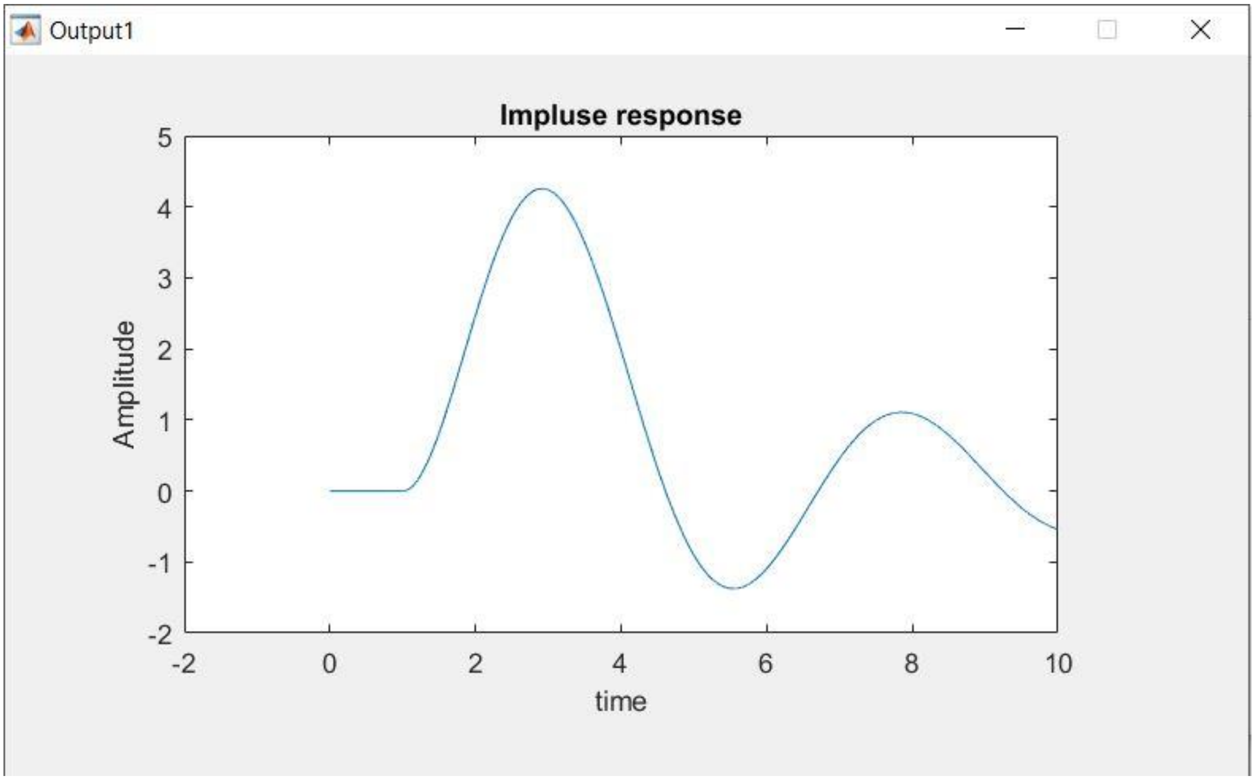
Command Window

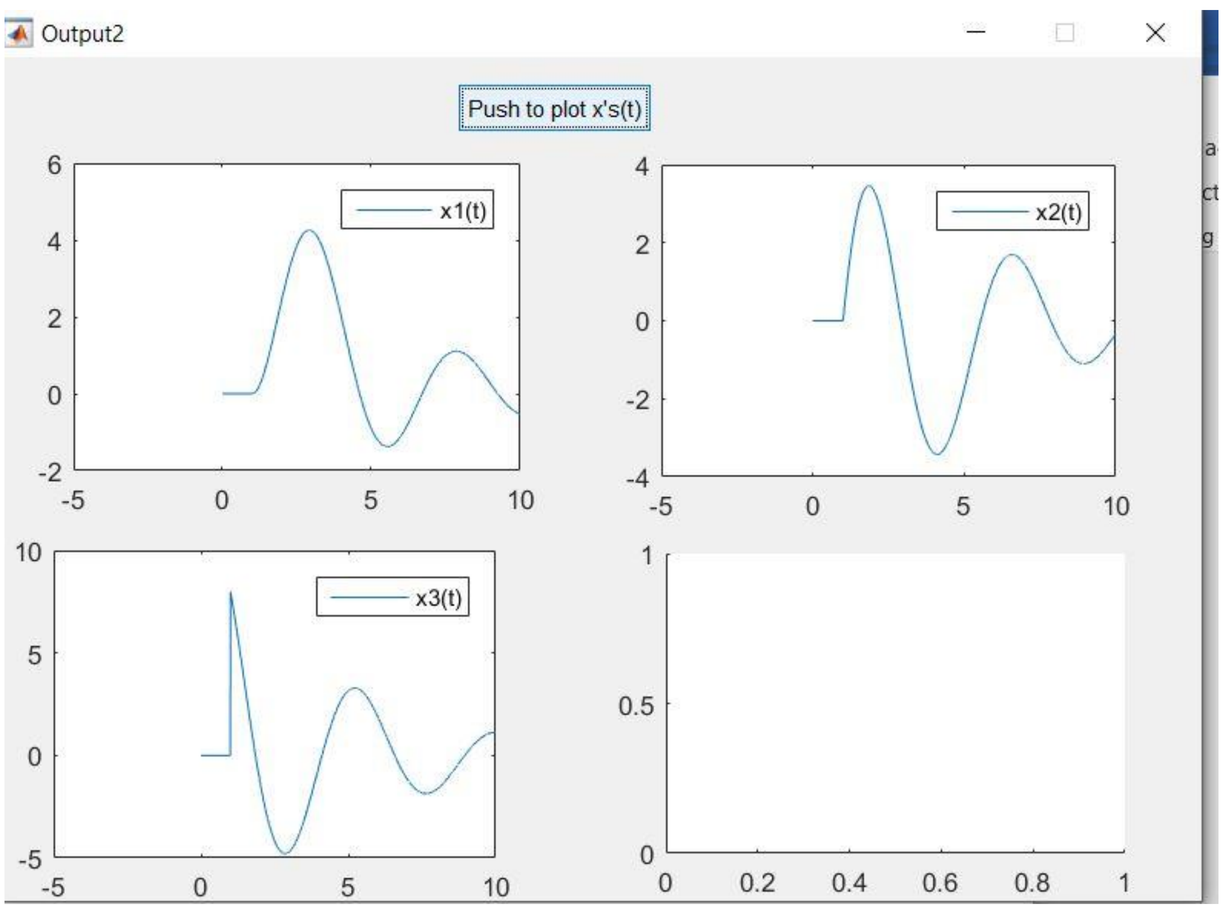
```
A =

     0     1     0
     0     0     1
    -2    -6    -3


B =

     0
     0
     1


C =

     8     0     0


D =

     0
```

$$4\frac{d^3y(t)}{dt}+\frac{d^2y(t)}{dt}+2\frac{dy(t)}{dt}+y(t)=8\text{u(t)}$$

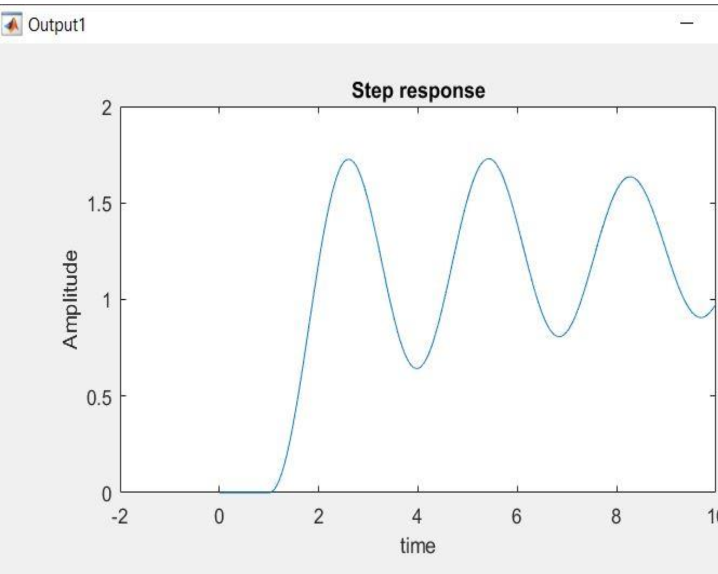


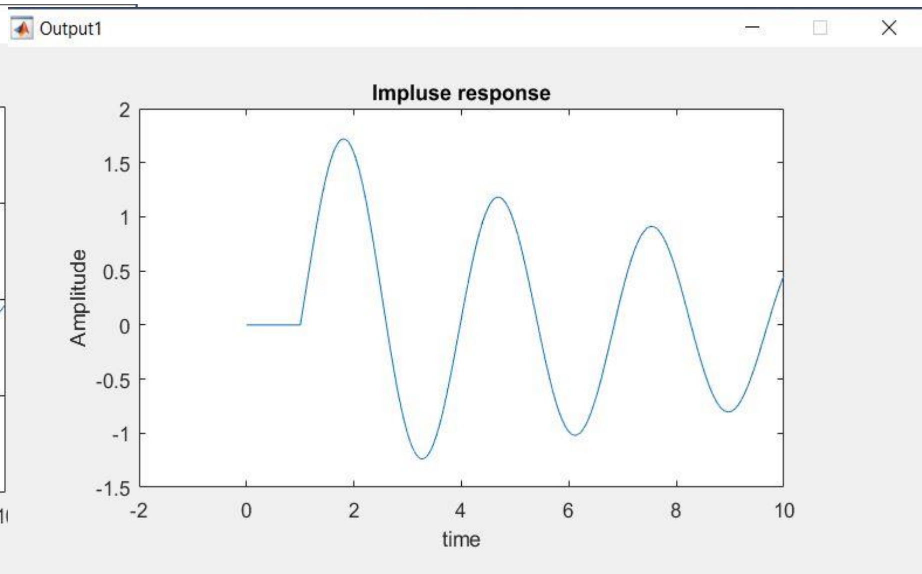



Output1
Impluse response
Amplitude
time
5
4
3
2
1
0
-1
-2
-2
0
2
4
6
8
10


Output2
Push to plot x's(t)
x1(t)
x2(t)
x3(t)

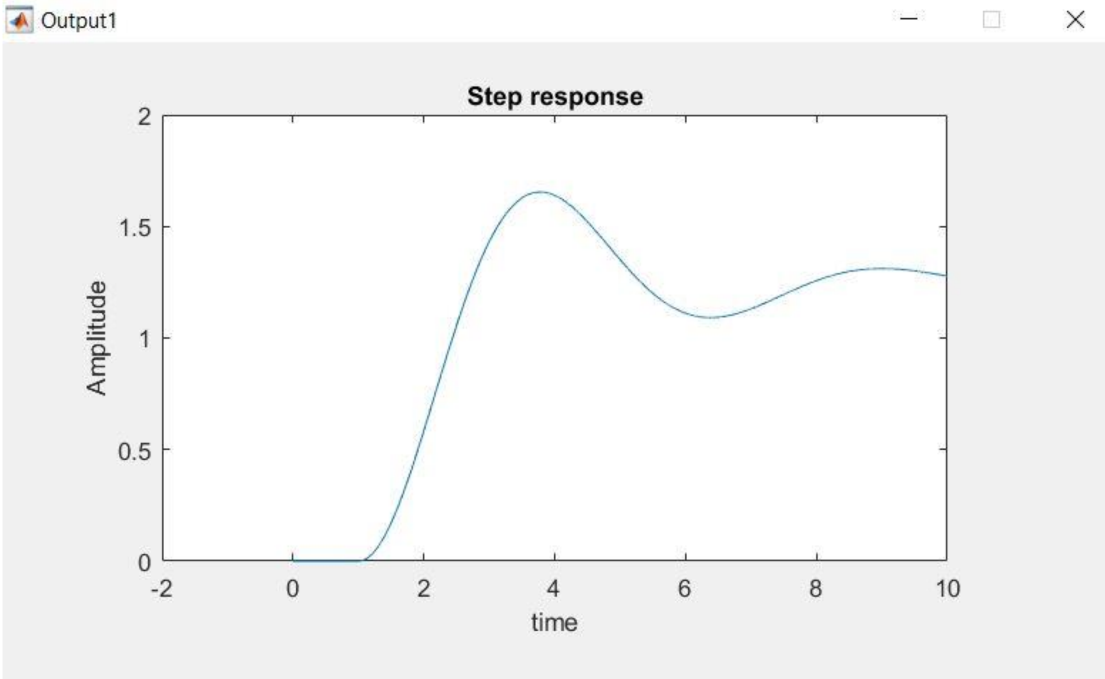
$$\frac{d^3y(t)}{dt}+\frac{d^2y(t)}{dt}+5\frac{dy(t)}{dt}+4y(t)=5\text{u(t)}+3\frac{du(t)}{dt}$$

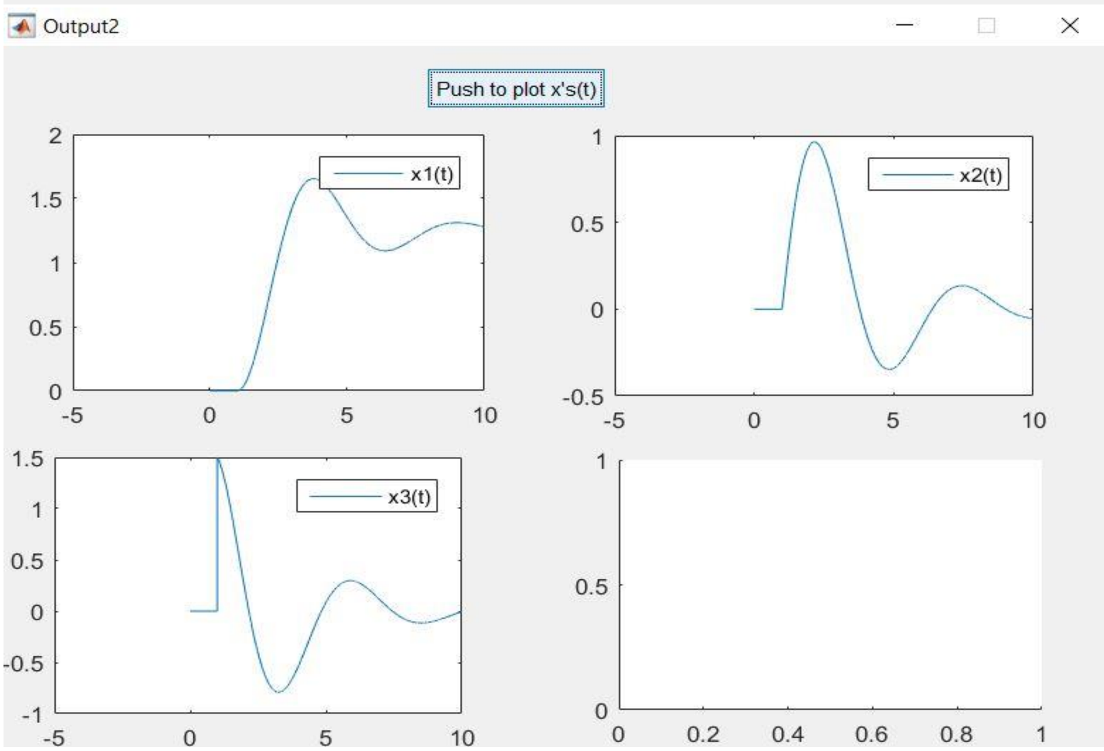


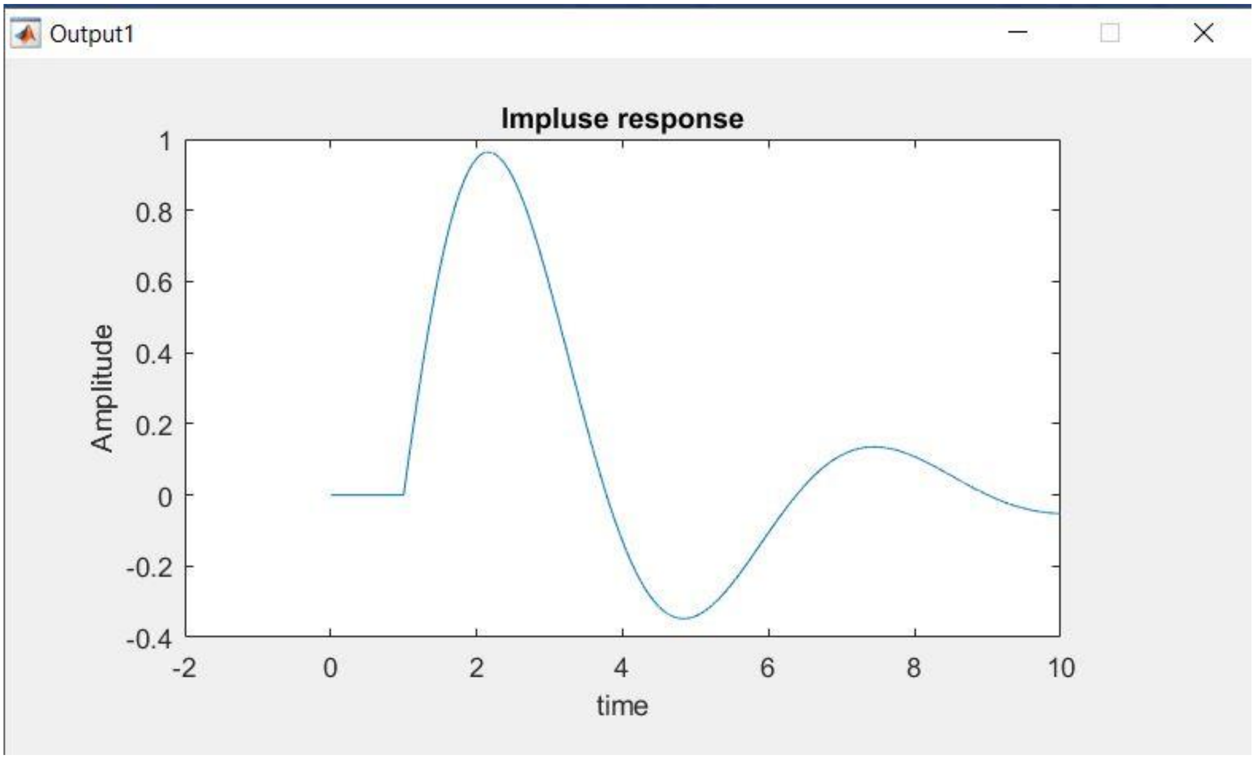


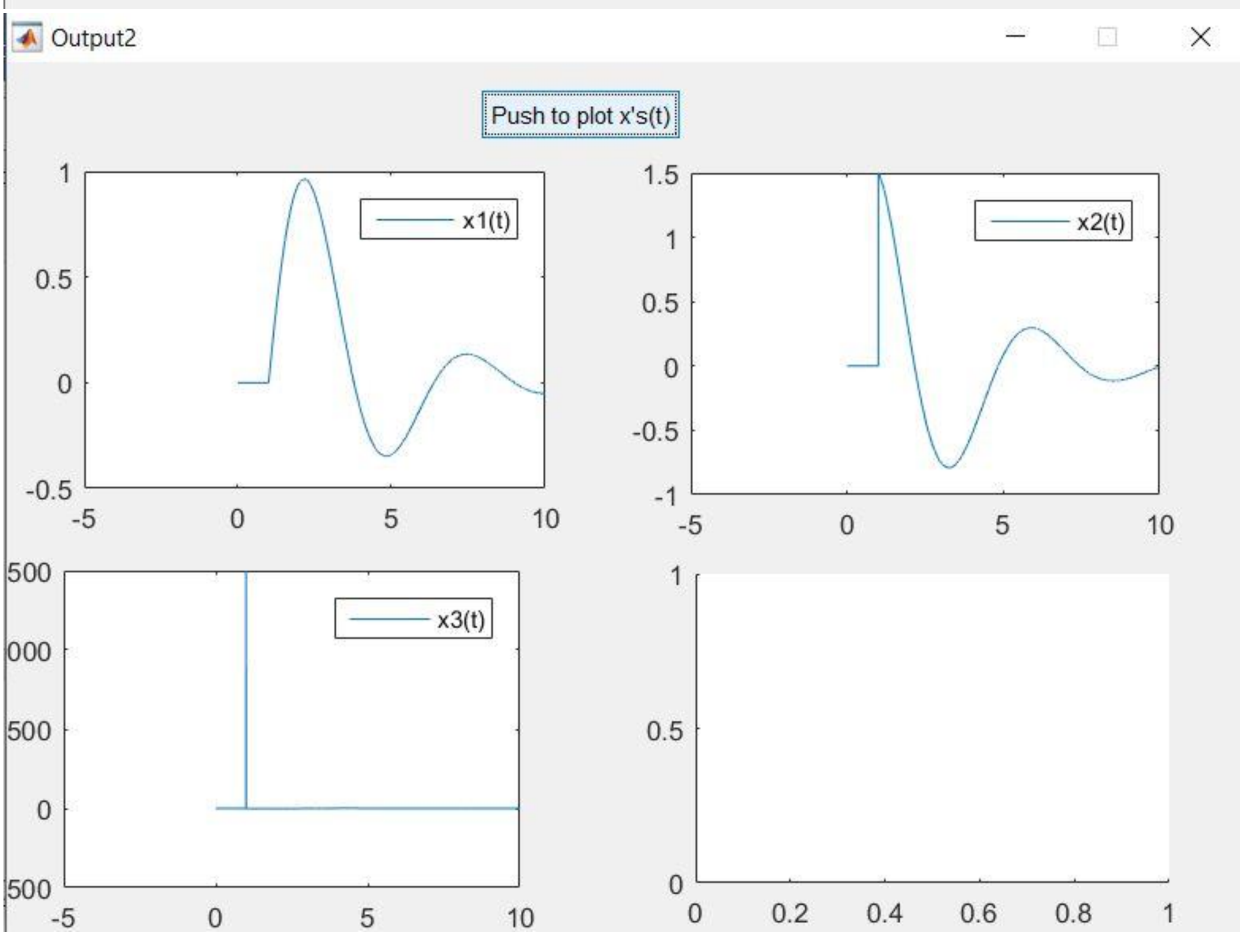

```
Command Window
A =

     0     1     0
     0     0     1
    -4    -5    -1


B =

     0
     0
     1


C =

     5     3     0


D =

     0

>>
```

$$2\frac{d^3y(t)}{dt} + 4\frac{d^2y(t)}{dt} + 5\frac{dy(t)}{dt} + 4y(t) = 5u(t) + 3\frac{du(t)}{dt}$$

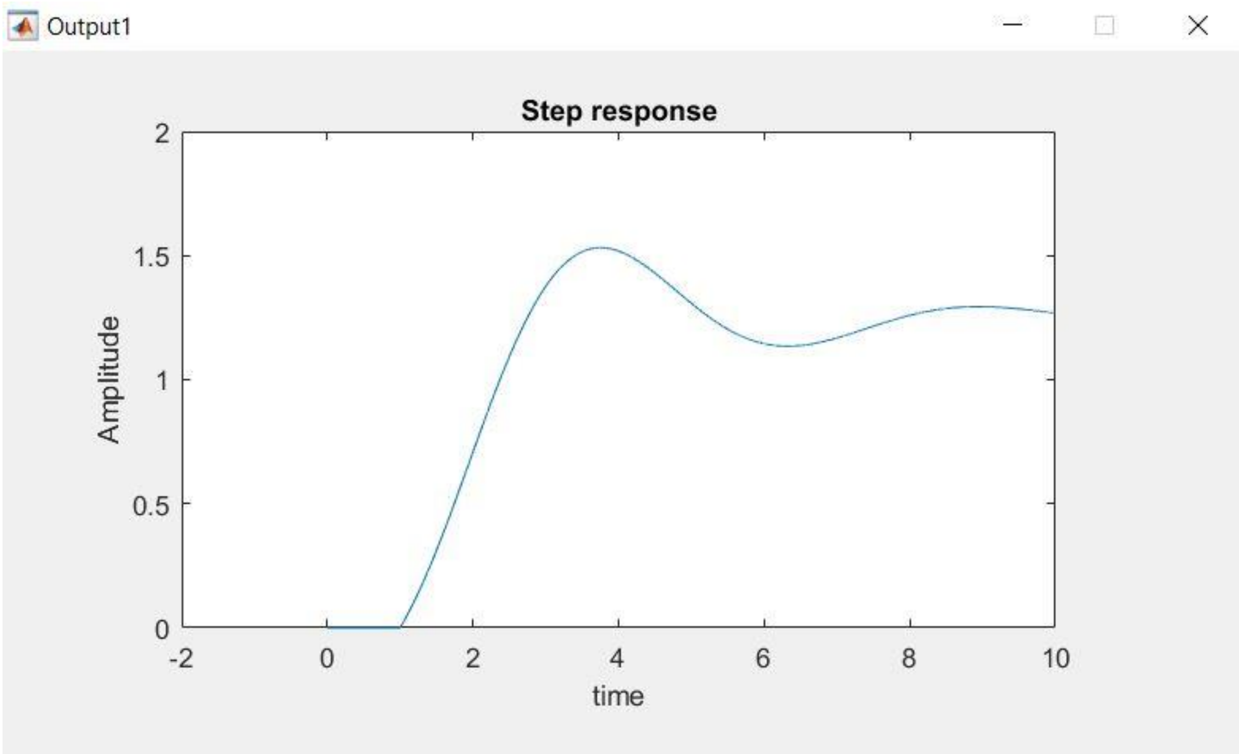


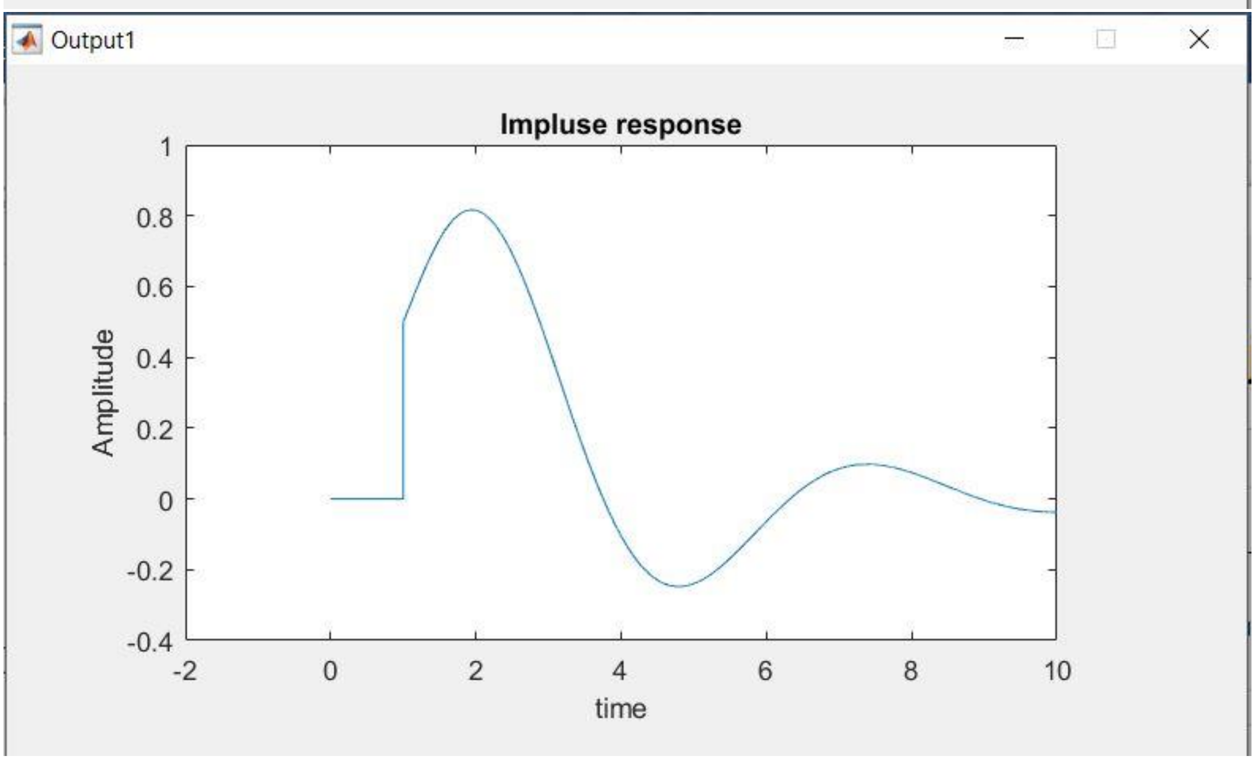



Output1
Impluse response
Amplitude
time

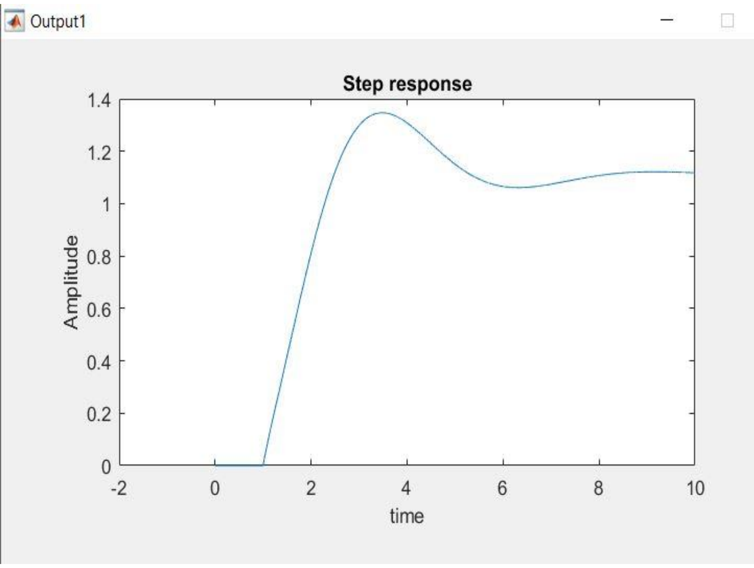

Output2
Push to plot x's(t)
x1(t)
x2(t)
x3(t)

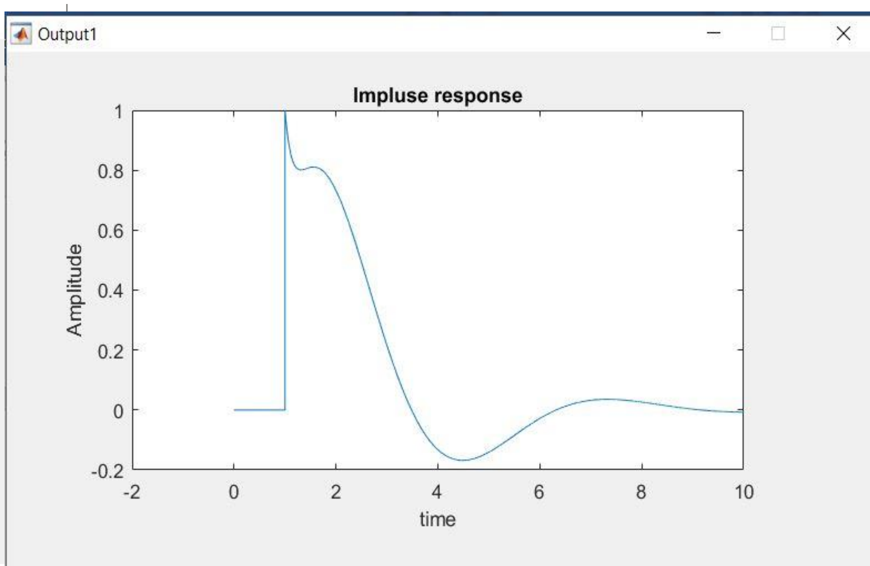
$$2\frac{d^3y(t)}{dt} + 4\frac{d^2y(t)}{dt} + 5\frac{dy(t)}{dt} + 4y(t) = 5\text{u(t)} + 3\frac{du(t)}{dt} + \frac{d^2u(t)}{dt}$$





$$\frac{d^3y(t)}{dt} + 7\,\frac{d^2y(t)}{dt} + 8\,\frac{dy(t)}{dt} + 9y(t) = 10\text{u(t)} + 5\,\frac{du(t)}{dt} + \frac{d^2u(t)}{dt}$$

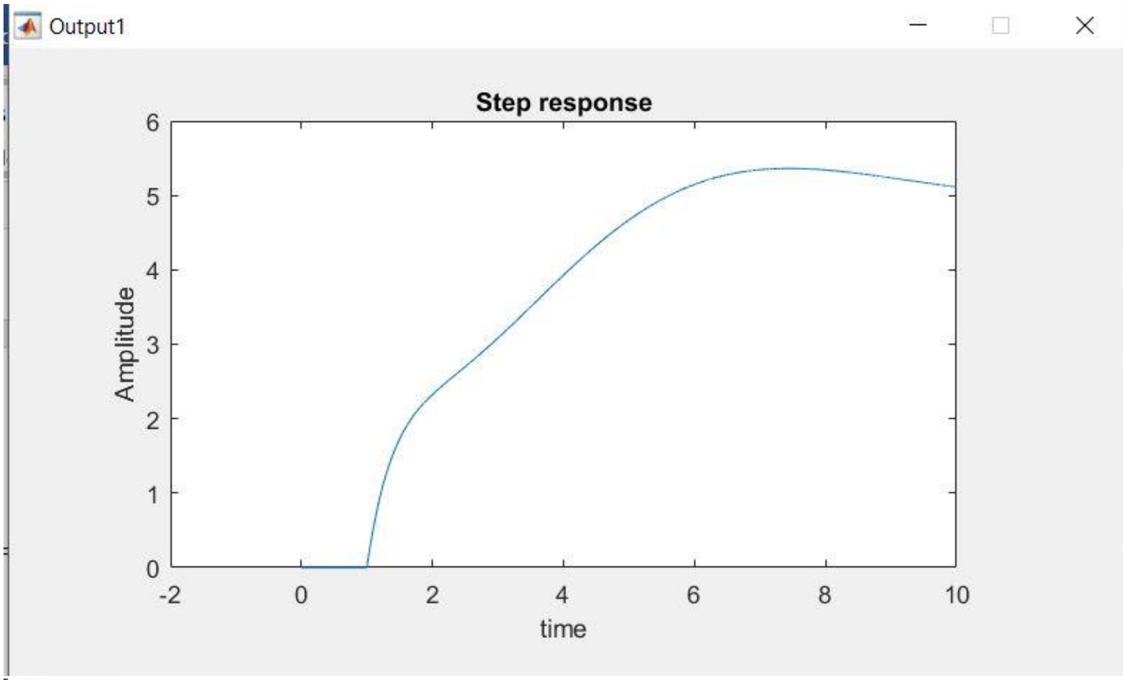


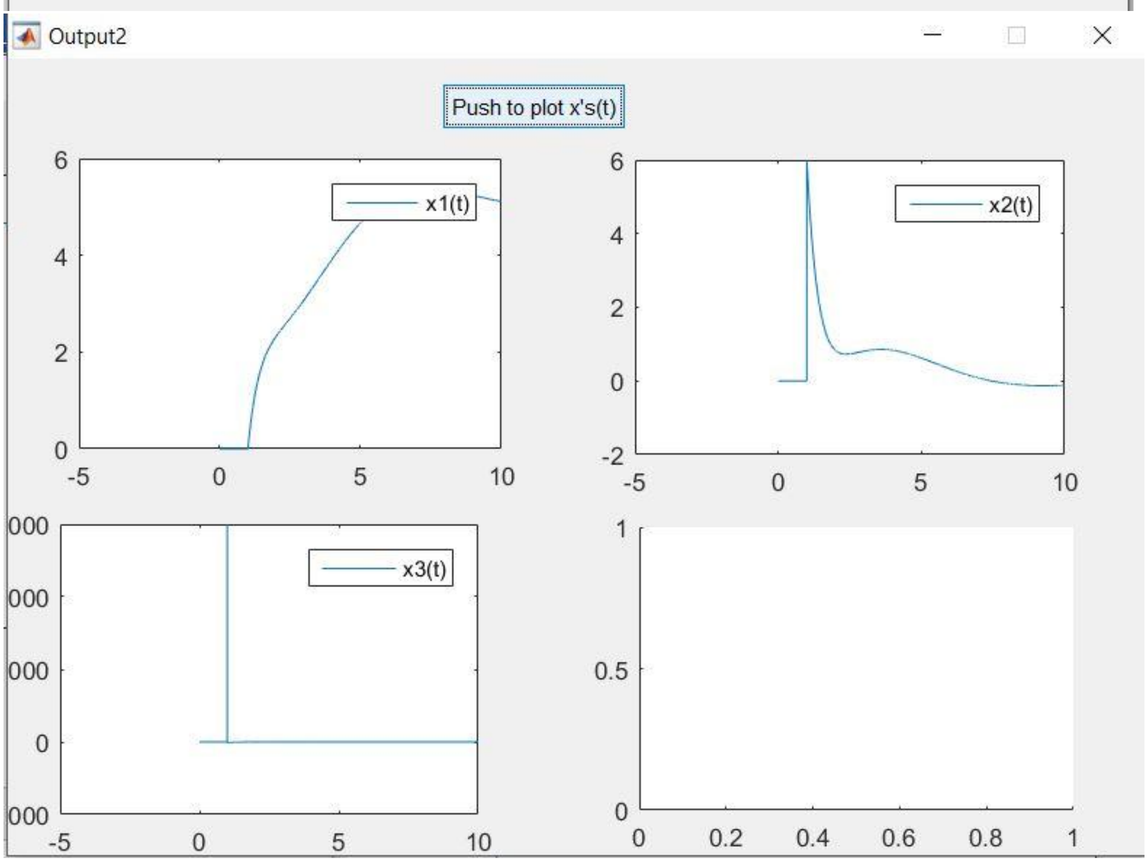


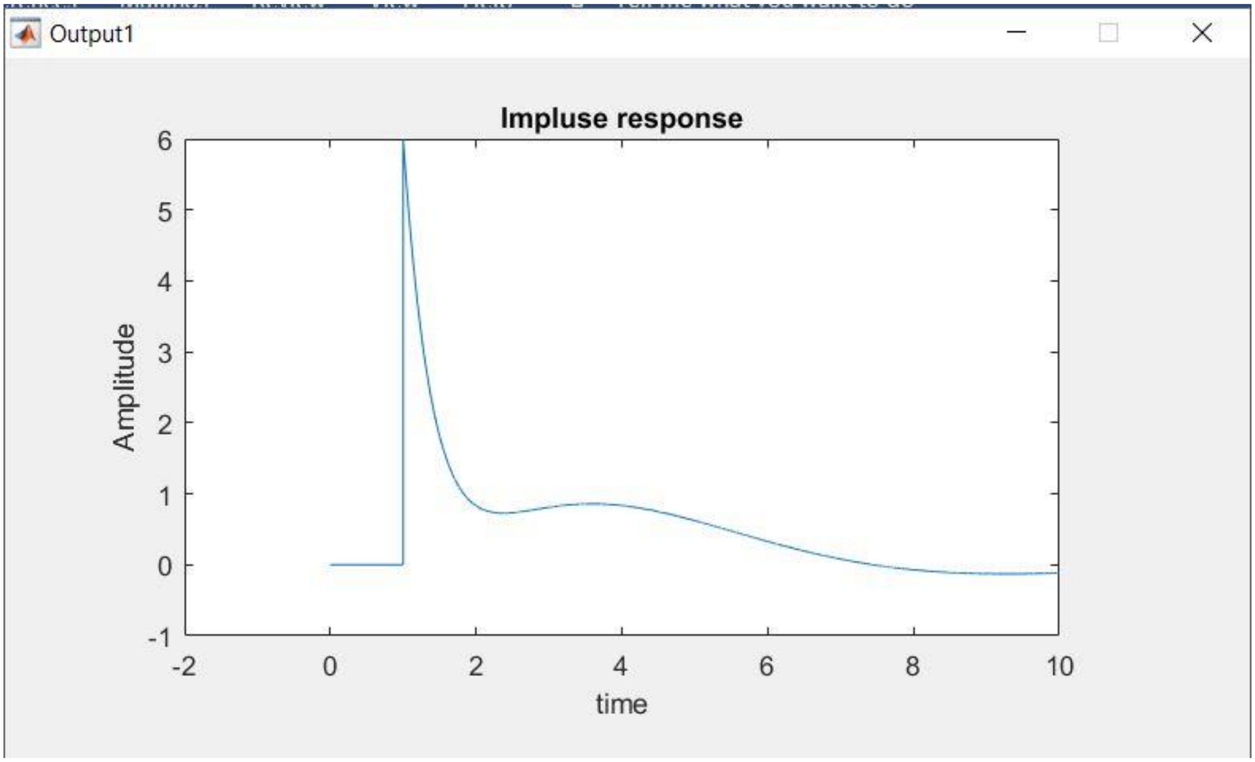
Command Window

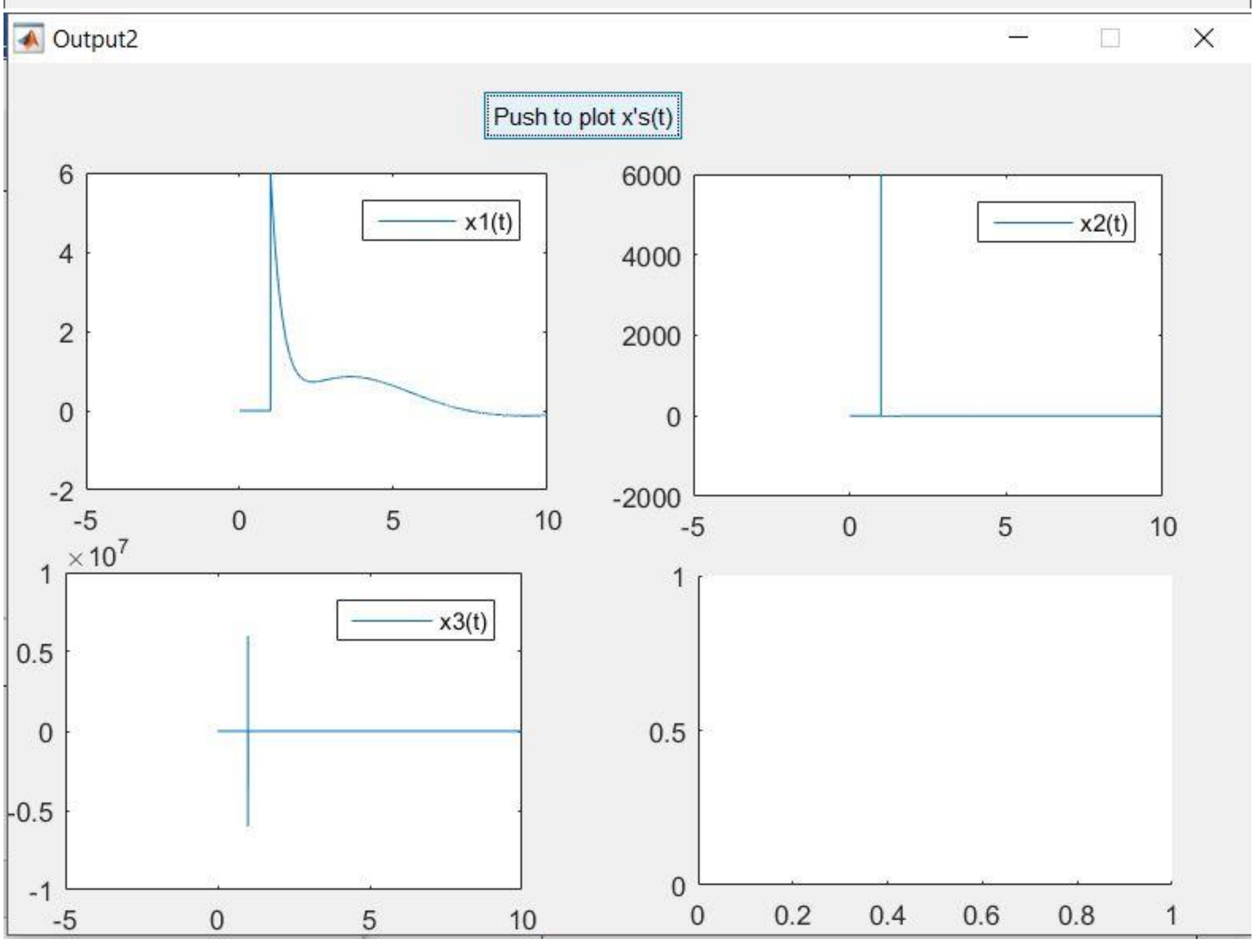
```
A =

     0     1     0
     0     0     1
    -9    -8    -7


B =

     0
     0
     1


C =

    10     5     1


D =

     0
```

$$\frac{d^3y(t)}{dt} + 3\frac{d^2y(t)}{dt} + 2\frac{dy(t)}{dt} + y(t) = 5\mathrm{u(t)} + 3\frac{du(t)}{dt} + 6\frac{d^2u(t)}{dt}$$

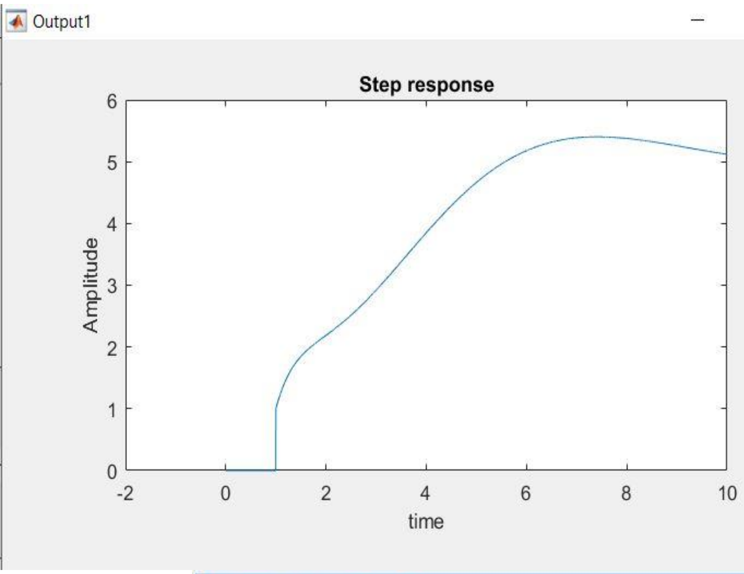


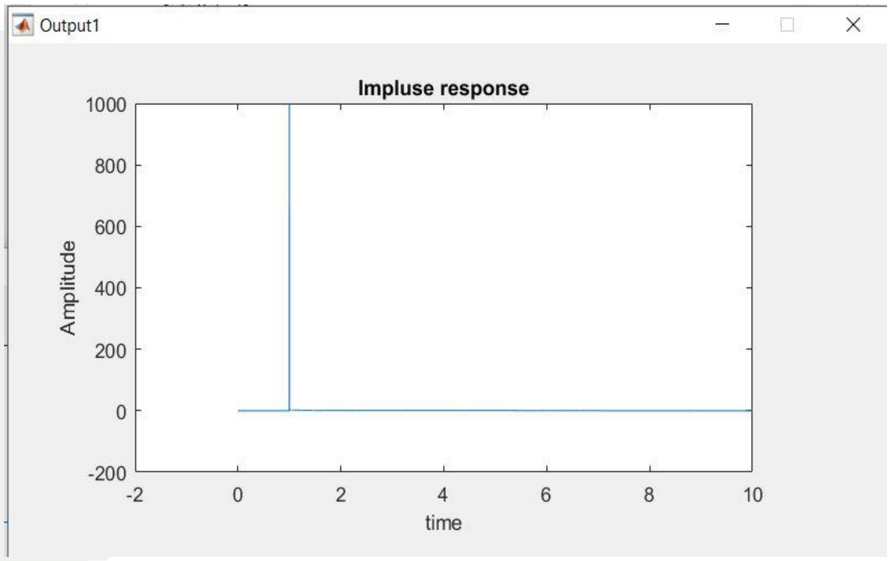



Output1
Impluse response
Amplitude
time
6
5
4
3
2
1
0
-1
-2
0
2
4
6
8
10


Output2
Push to plot x's(t)
x1(t)
x2(t)
x3(t)
6
4
2
0
-2
-5
0
5
10
6000
4000
2000
0
-2000
$\times 10^7$
1
0.5
0
-0.5
-1
0.2
0.4
0.6
0.8

$$\frac{d^3y(t)}{dt} + 3\frac{d^2y(t)}{dt} + 2\frac{dy(t)}{dt} + y(t) = 5u(t) + 3\frac{du(t)}{dt} + 6\frac{d^2u(t)}{dt} + \frac{d^3u(t)}{dt}$$

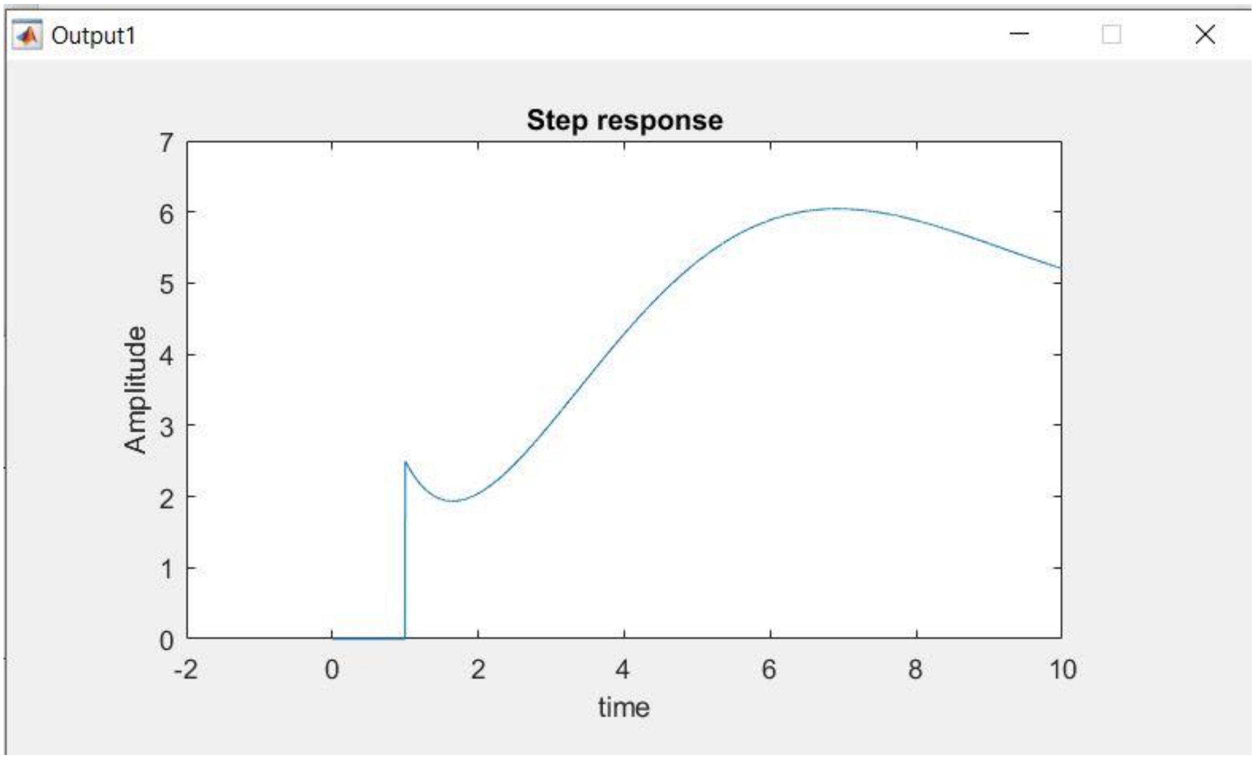


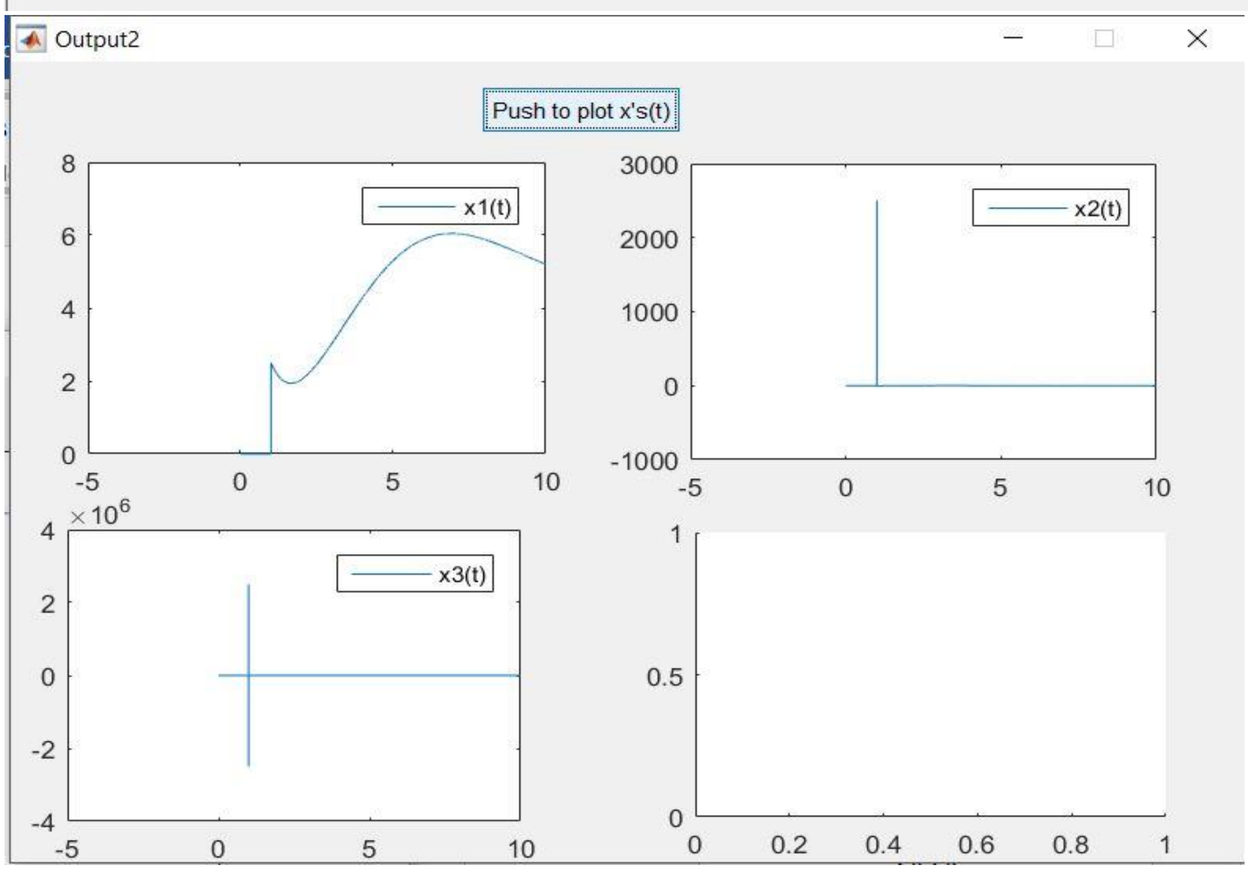


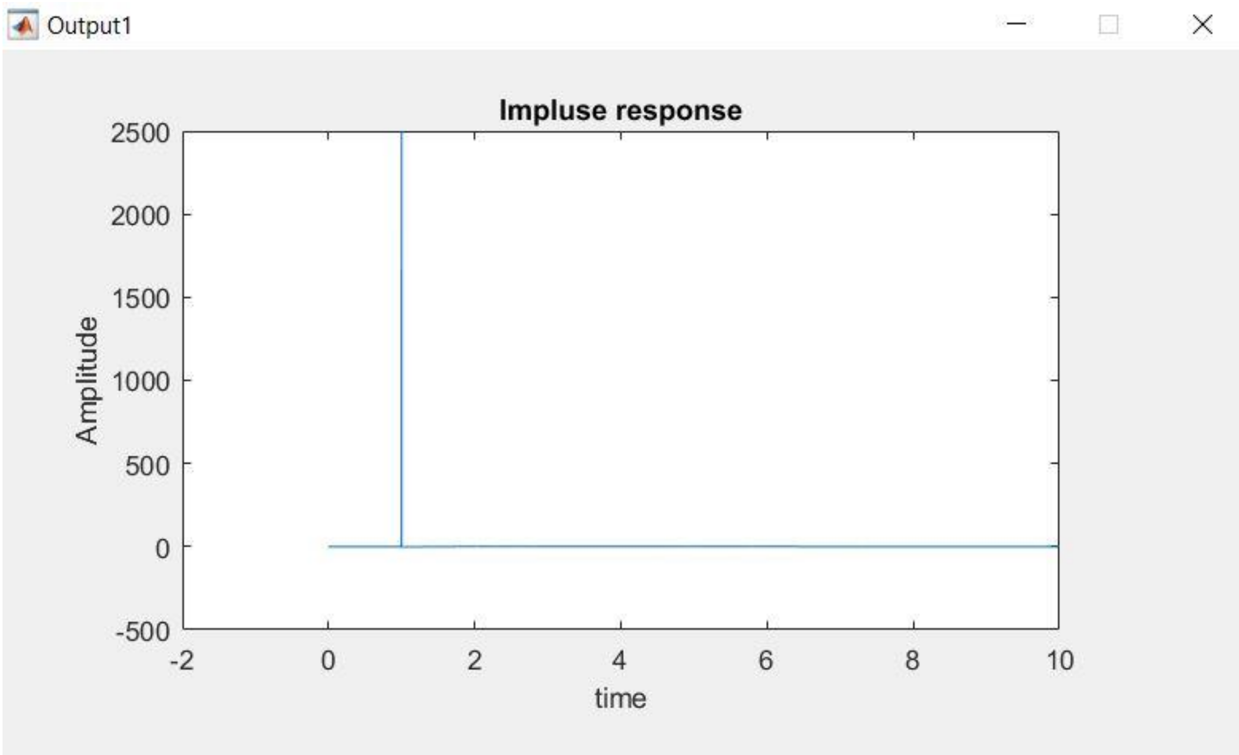
Command Window

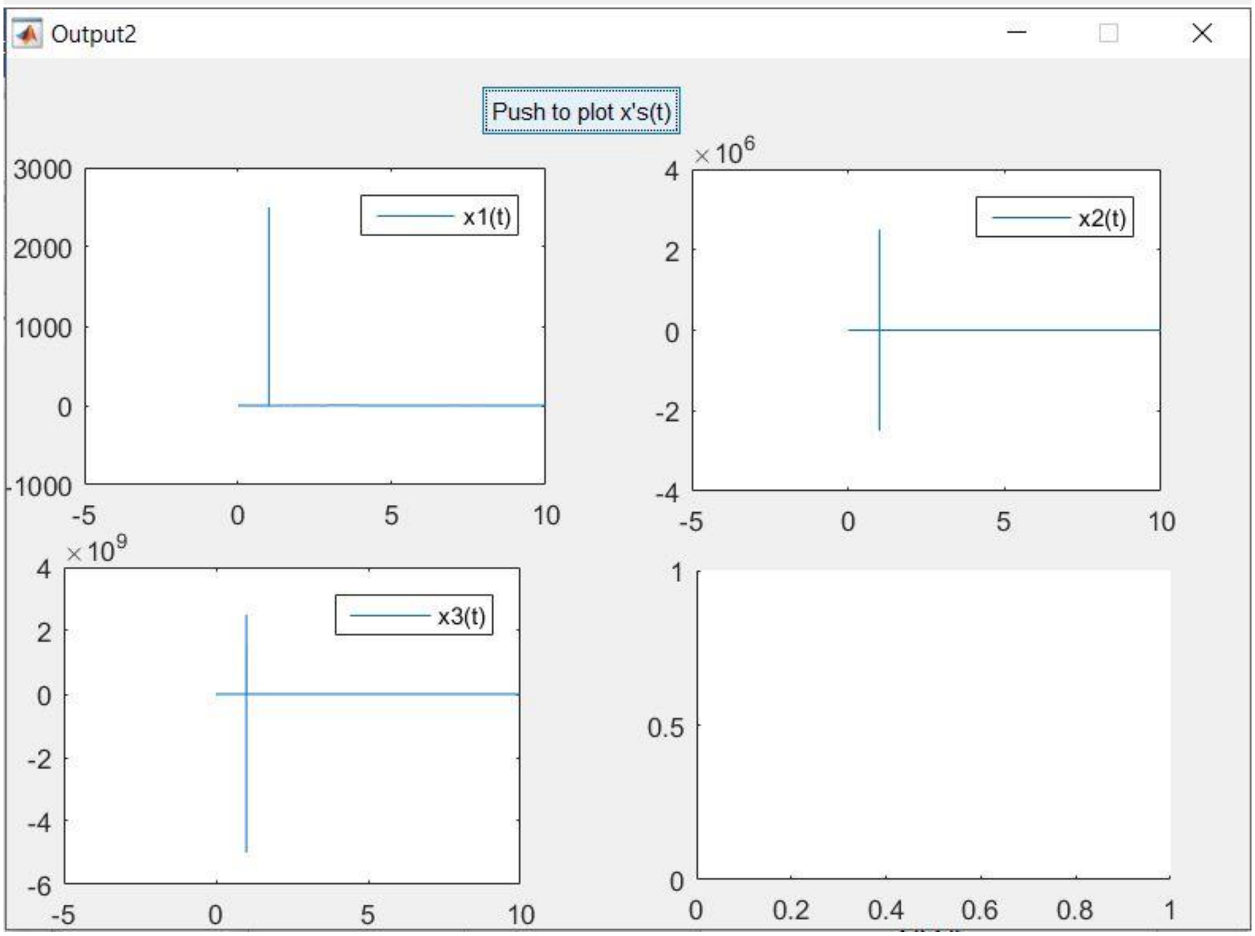
```
A =

     0     1     0
     0     0     1
    -1    -2    -3


B =

     0
     0
     1


C =

     4     1     3


D =

     1
```

$$2\frac{d^3y(t)}{dt}+4\frac{d^2y(t)}{dt}+2\frac{dy(t)}{dt}+y(t)=5\mathrm{u(t)}+6\frac{du(t)}{dt}+\ 6\frac{d^2u(t)}{dt}+5\frac{d^3u(t)}{dt}$$



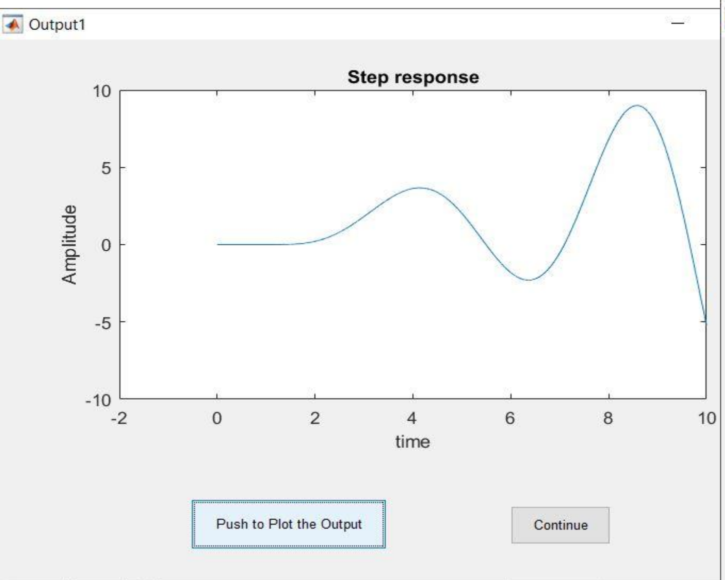


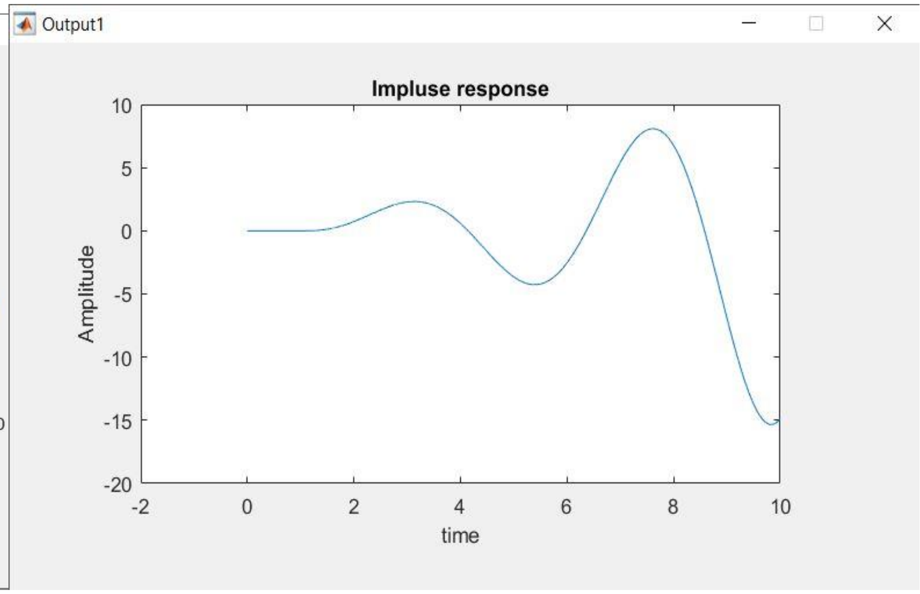

Output1
Impluse response
Amplitude
time
2500
2000
1500
1000
500
0
-500
-2
0
2
4
6
8
10


Output2
Push to plot x's(t)
x1(t)
x2(t)
x3(t)
×10^6
×10^9
3000
2000
1000
0
-1000
-5
0
5
10
4
2
0
-2
-4
-6
1
0.5
0.2
0.4
0.6
0.8

Fourth Order:

$$\frac{d^4y(t)}{dt} + 2\frac{d^3y(t)}{dt} + 3\frac{d^2y(t)}{dt} + 4\frac{dy(t)}{dt} + 5y(t) = 8\text{u(t)}$$

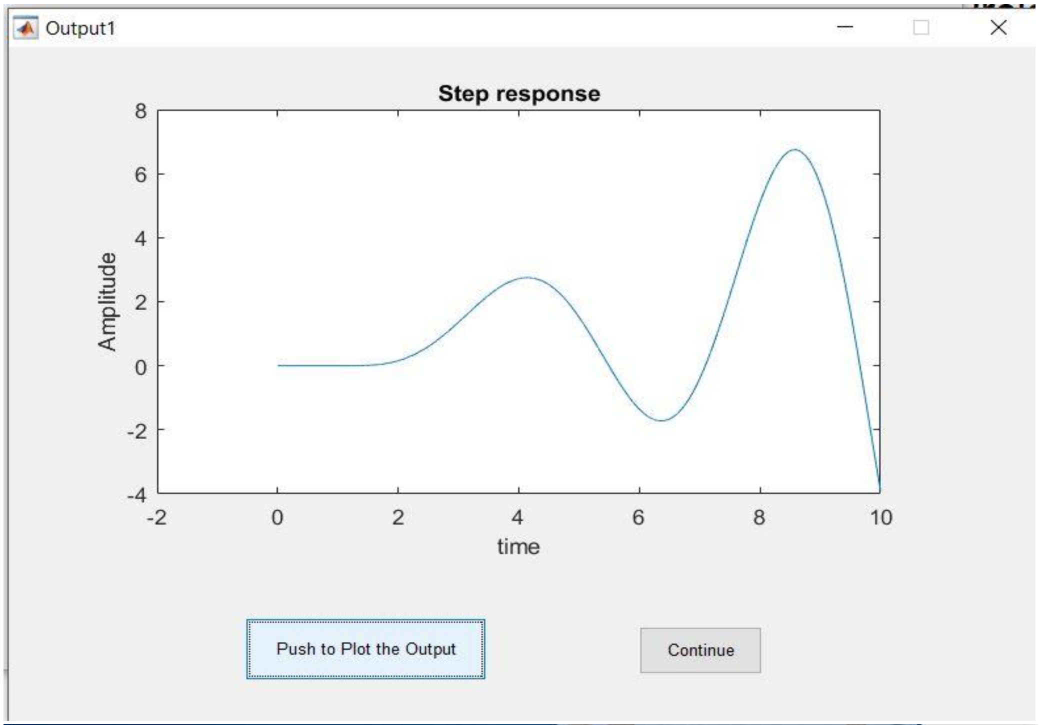


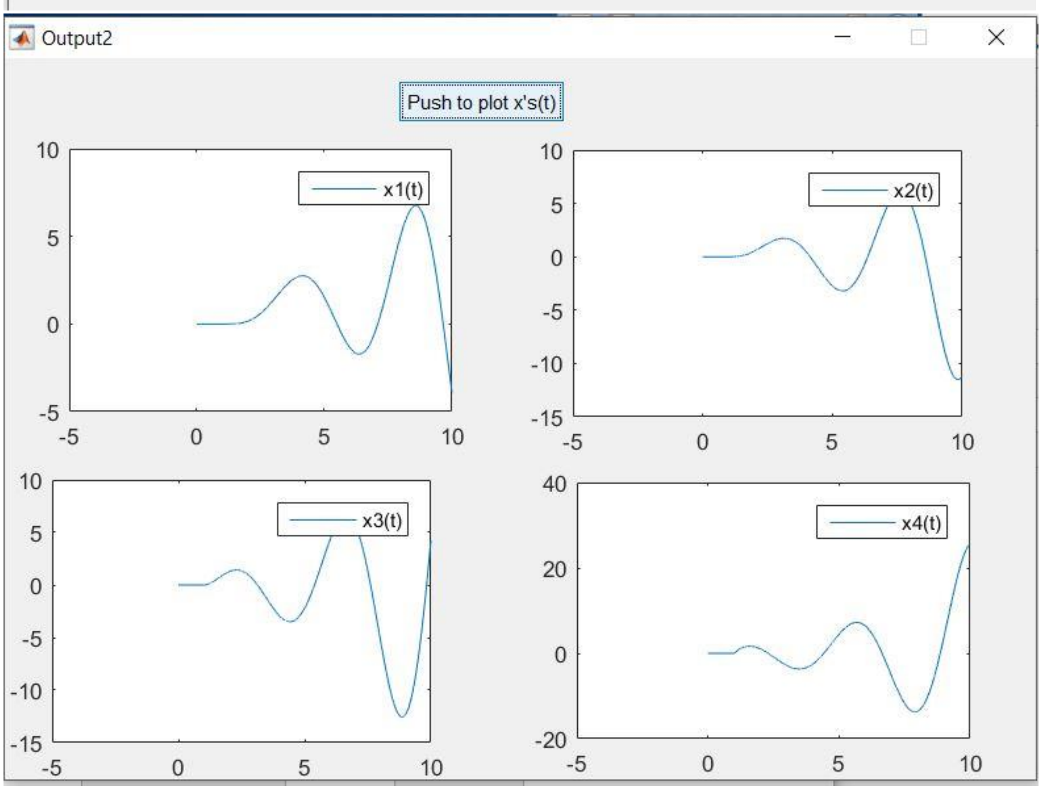


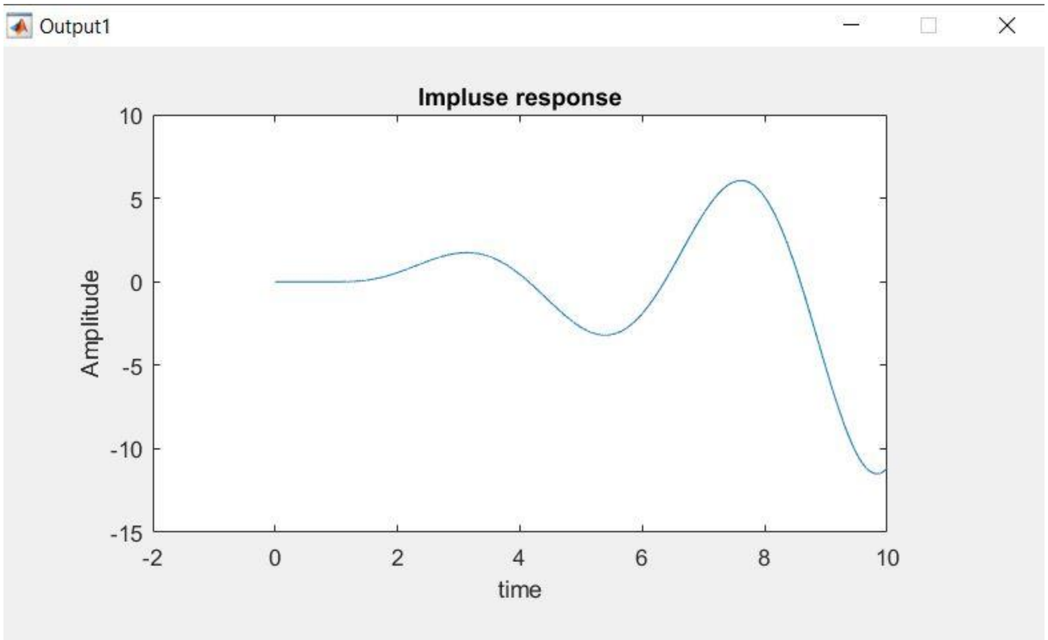
Command Window

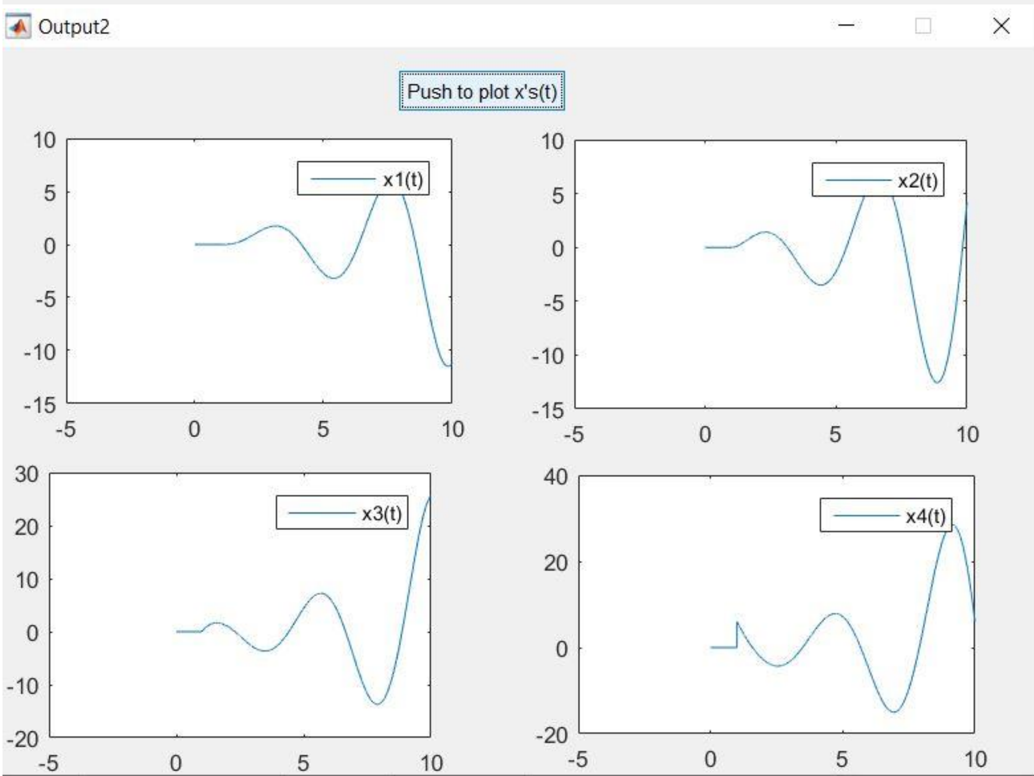
```
A =

     0     1     0     0
     0     0     1     0
     0     0     0     1
    -5    -4    -3    -2


B =

     0
     0
     0
     1


C =

     8     0     0     0


D =

     0
```

$$\frac{d^4y(t)}{dt} + 2\frac{d^3y(t)}{dt} + 3\frac{d^2y(t)}{dt} + 4\frac{dy(t)}{dt} + 5y(t) = 6\mathrm{u(t)}$$

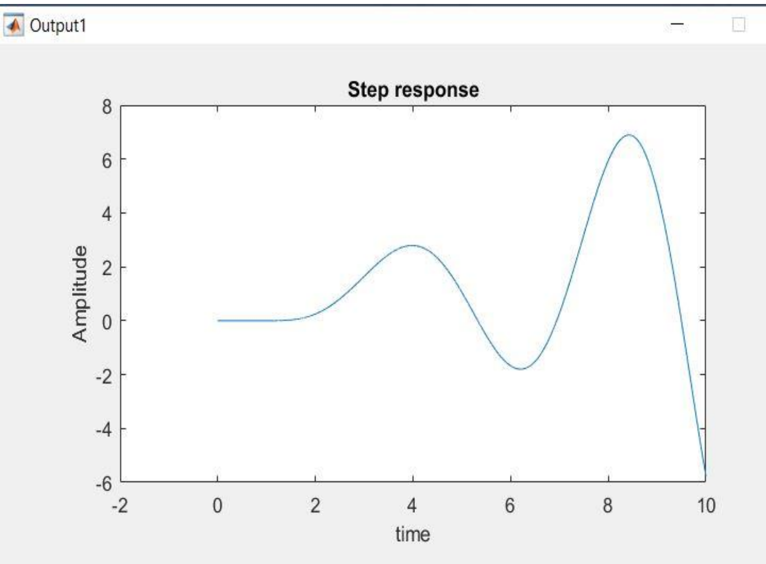


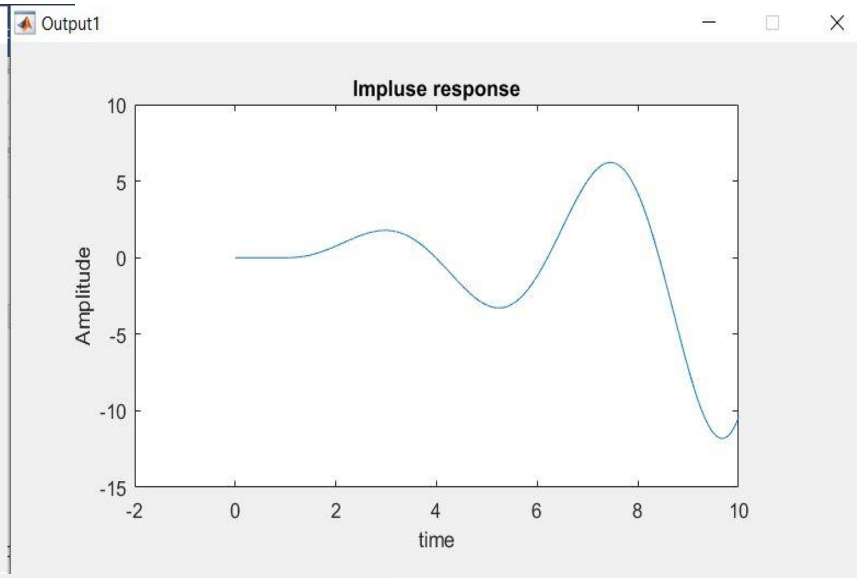



Output1
Impluse response
Amplitude
time
10
5
0
-5
-10
-15
-2
0
2
4
6
8
10


Output2
Push to plot x's(t)
x1(t)
x2(t)
x3(t)
x4(t)
10
5
0
-5
-10
-15
30
20
-20
40
-5
0
5
10

$$\frac{d^4y(t)}{dt} + 2\frac{d^3y(t)}{dt} + 3\frac{d^2y(t)}{dt} + 4\frac{dy(t)}{dt} + 5y(t) = 6\mathrm{u(t)} + \frac{du(t)}{dt}$$

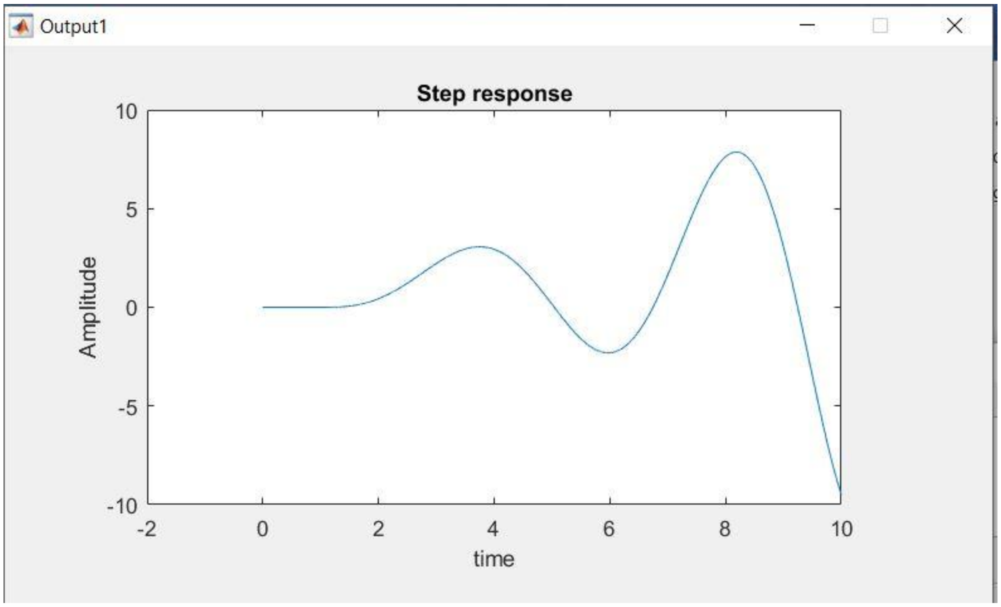


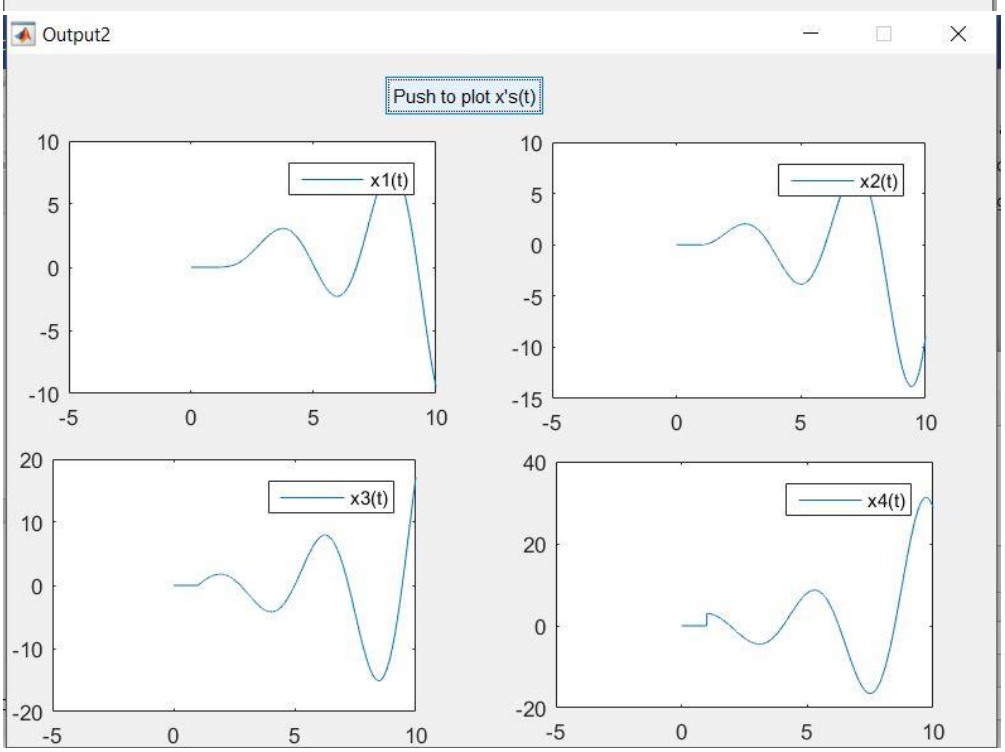


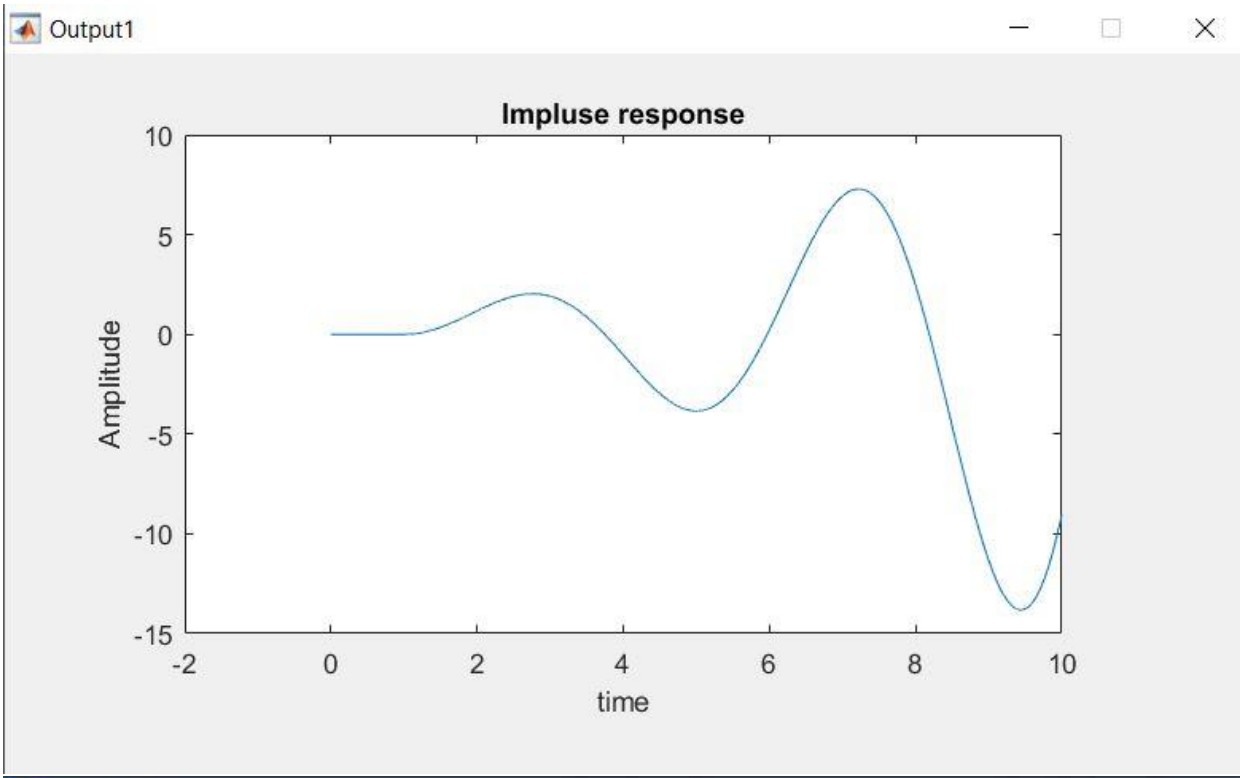
Command Window

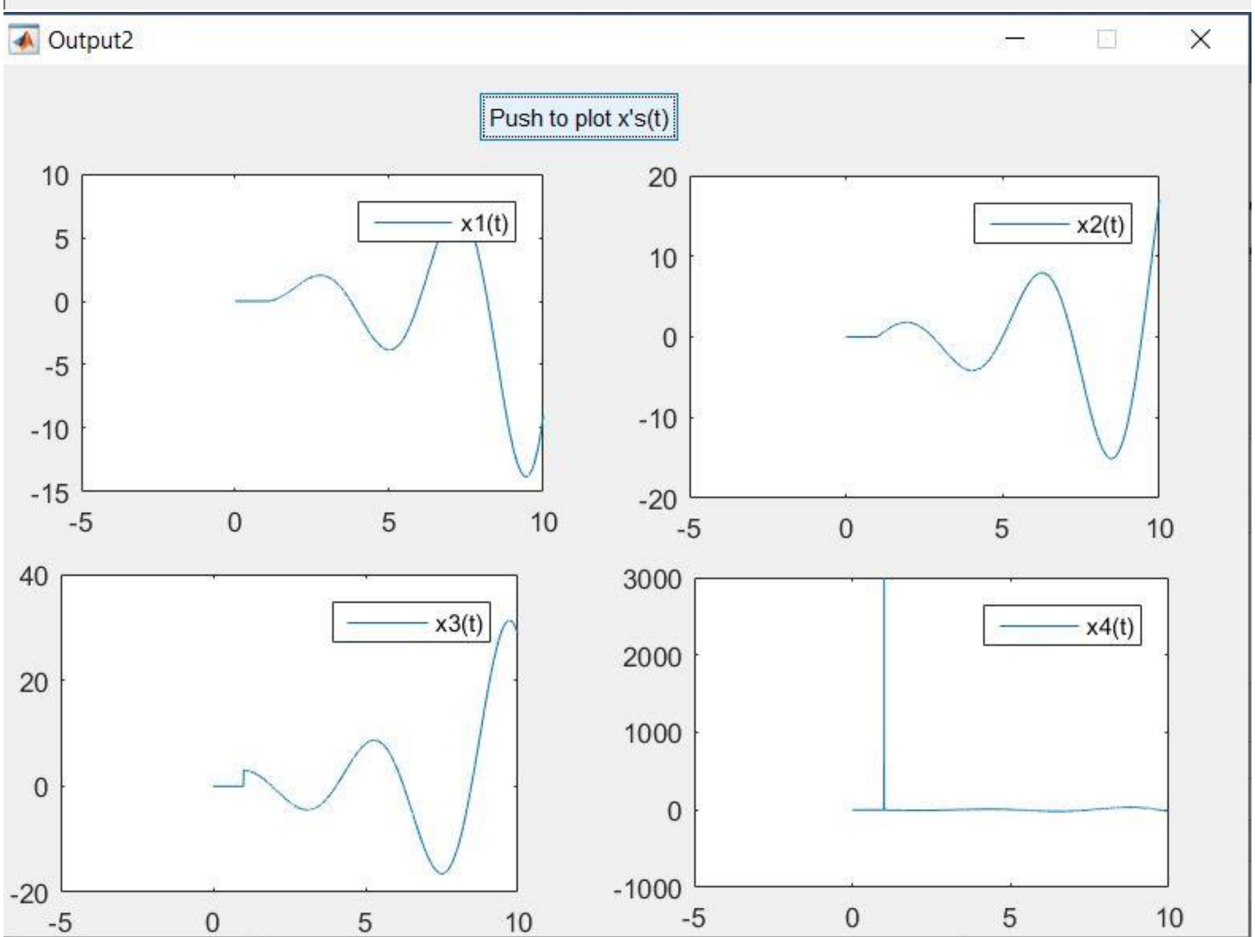
```
A =

     0     1     0     0
     0     0     1     0
     0     0     0     1
    -5    -4    -3    -2


B =

     0
     0
     0
     1


C =

     6     1     0     0


D =

     0
```

$$\frac{d^4y(t)}{dt} + 2\frac{d^3y(t)}{dt} + 3\frac{d^2y(t)}{dt} + 4\frac{dy(t)}{dt} + 5y(t) = 6u(t) + 3\frac{du(t)}{dt}$$

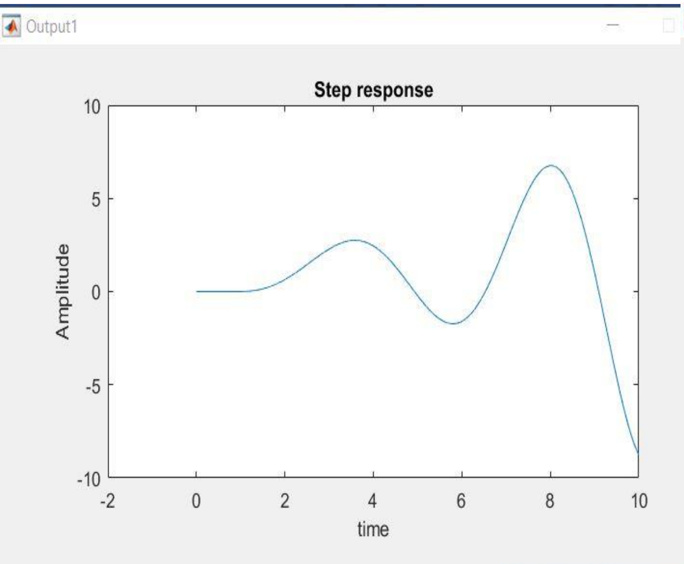


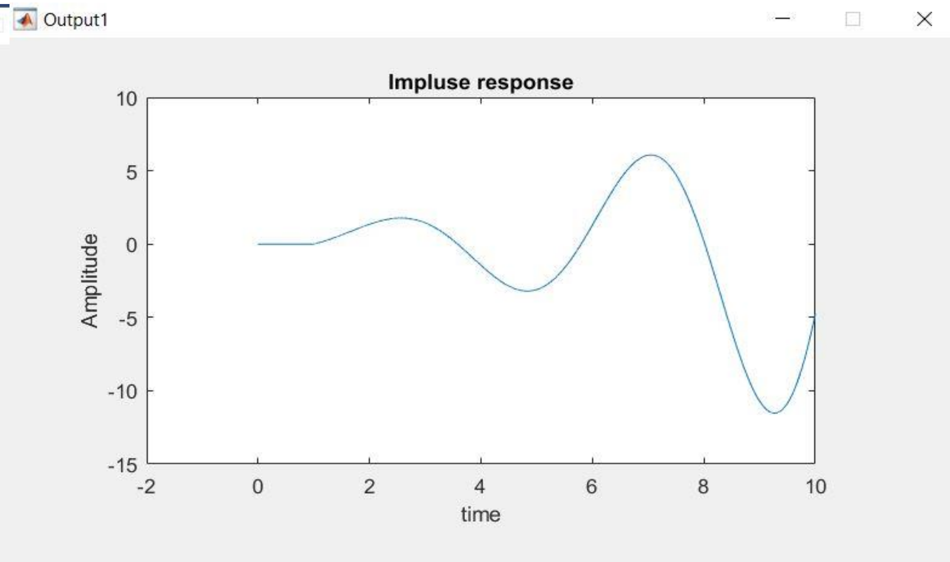



Output1
Impluse response
Amplitude
time
10
5
0
-5
-10
-15
-2
0
2
4
6
8
10


Output2
Push to plot x's(t)
x1(t)
x2(t)
x3(t)
x4(t)
3000
2000
1000
0
-1000

$$\frac{d^4y(t)}{dt} + 2\frac{d^3y(t)}{dt} + 3\frac{d^2y(t)}{dt} + 4\frac{dy(t)}{dt} + 5y(t) = 6\mathrm{u(t)} + 3\frac{du(t)}{dt} + \frac{d^2u(t)}{dt}$$

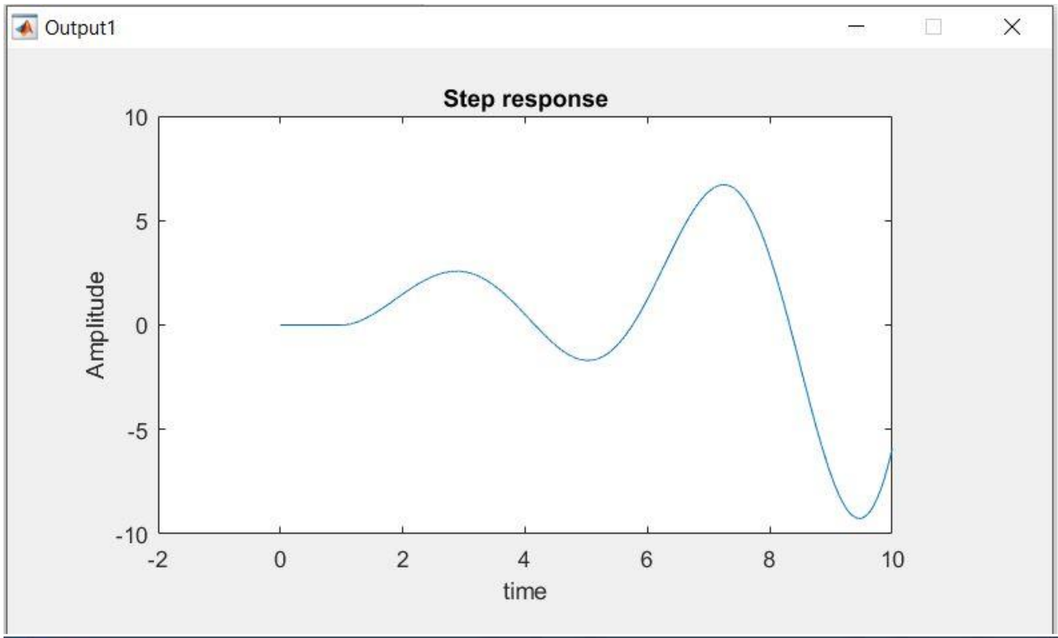


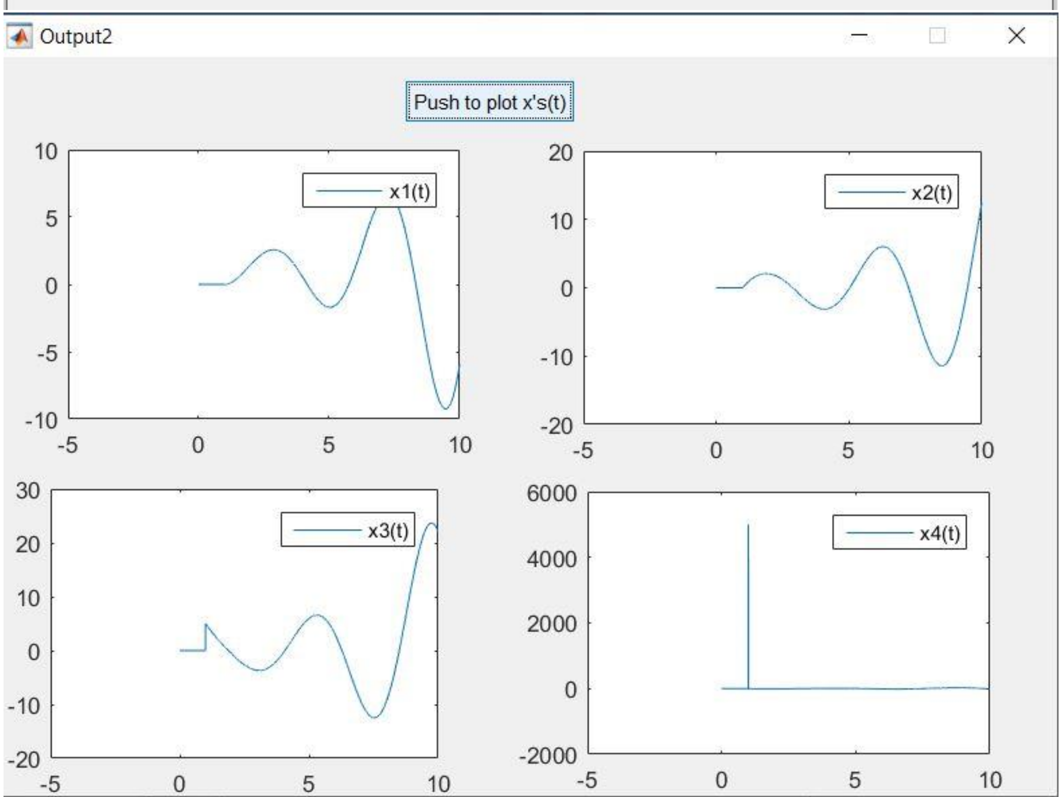


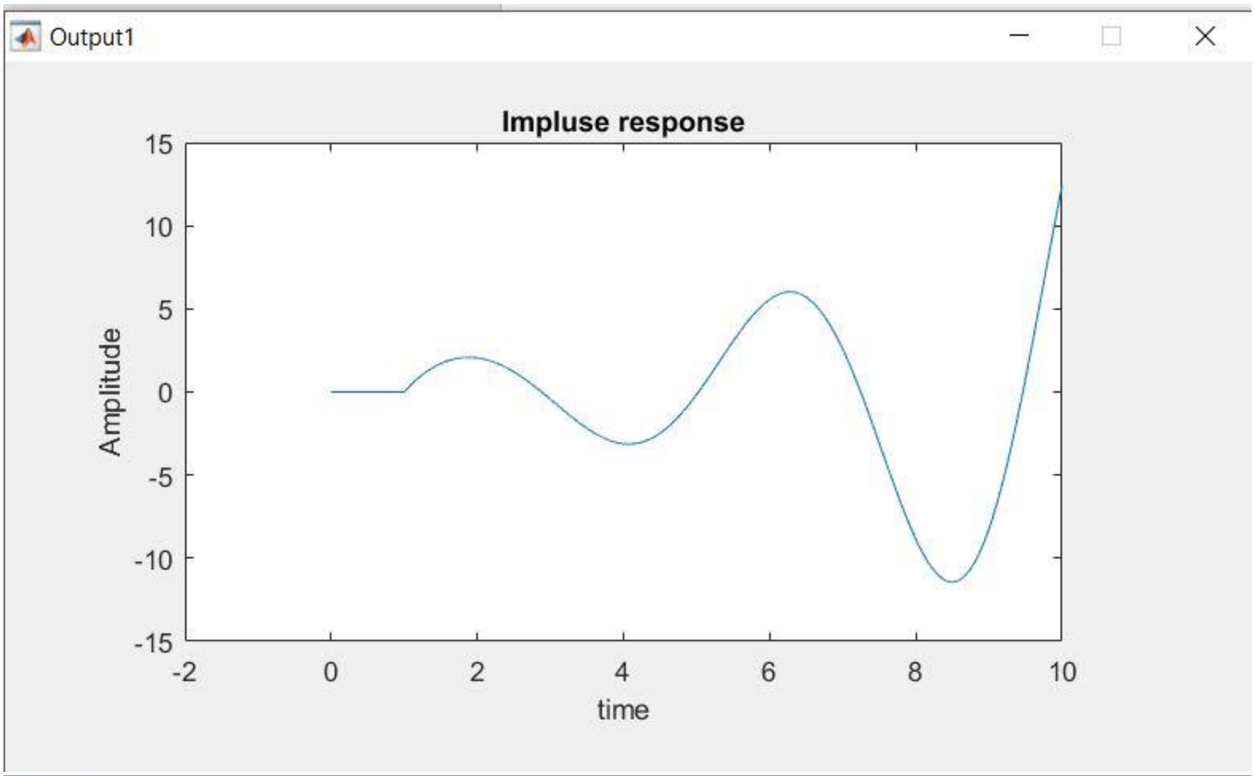
Command Window

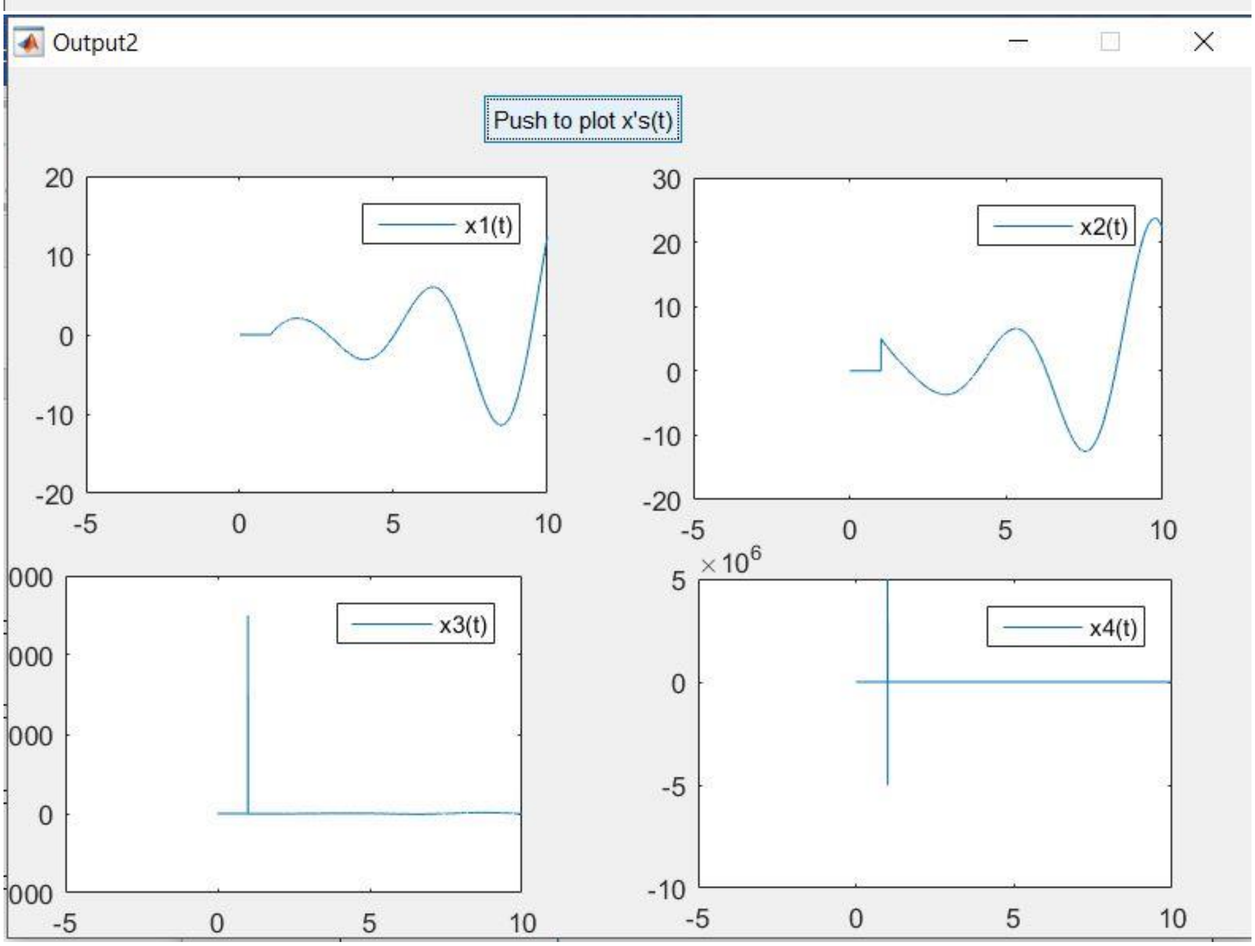
```
A =

     0     1     0     0
     0     0     1     0
     0     0     0     1
    -5    -4    -3    -2


B =

     0
     0
     0
     1


C =

     6     3     1     0


D =

     0
```

$$\frac{d^4y(t)}{dt} + 2\,\frac{d^3y(t)}{dt} + 3\,\frac{d^2y(t)}{dt} + 4\,\frac{dy(t)}{dt} + 5y(t) = 6u(t) + 3\,\frac{du(t)}{dt} + 5\,\frac{d^2u(t)}{dt}$$

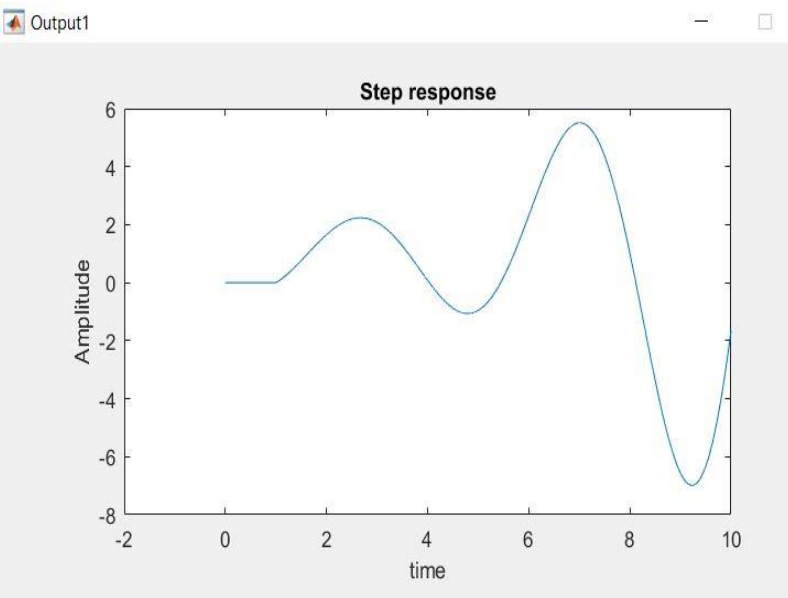


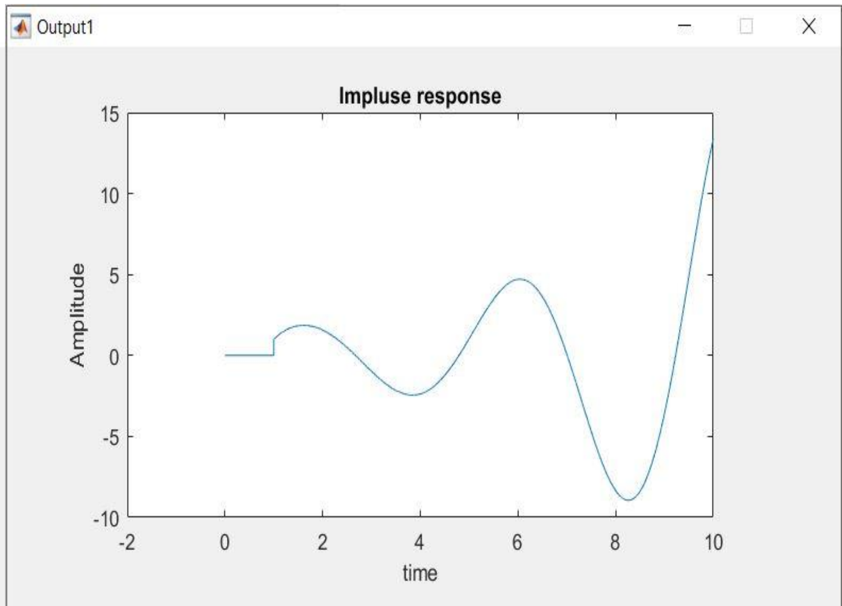



Output1
Impluse response
Amplitude
time


Output2
Push to plot x's(t)
x1(t)
x2(t)
x3(t)
x4(t)

$$\frac{d^4y(t)}{dt} + 2\,\frac{d^3y(t)}{dt} + 3\,\frac{d^2y(t)}{dt} + 4\,\frac{dy(t)}{dt} + 5y(t) = 6\mathrm{u(t)} + 3\,\frac{du(t)}{dt} + 5\,\frac{d^2u(t)}{dt} + \frac{d^3u(t)}{dt}$$

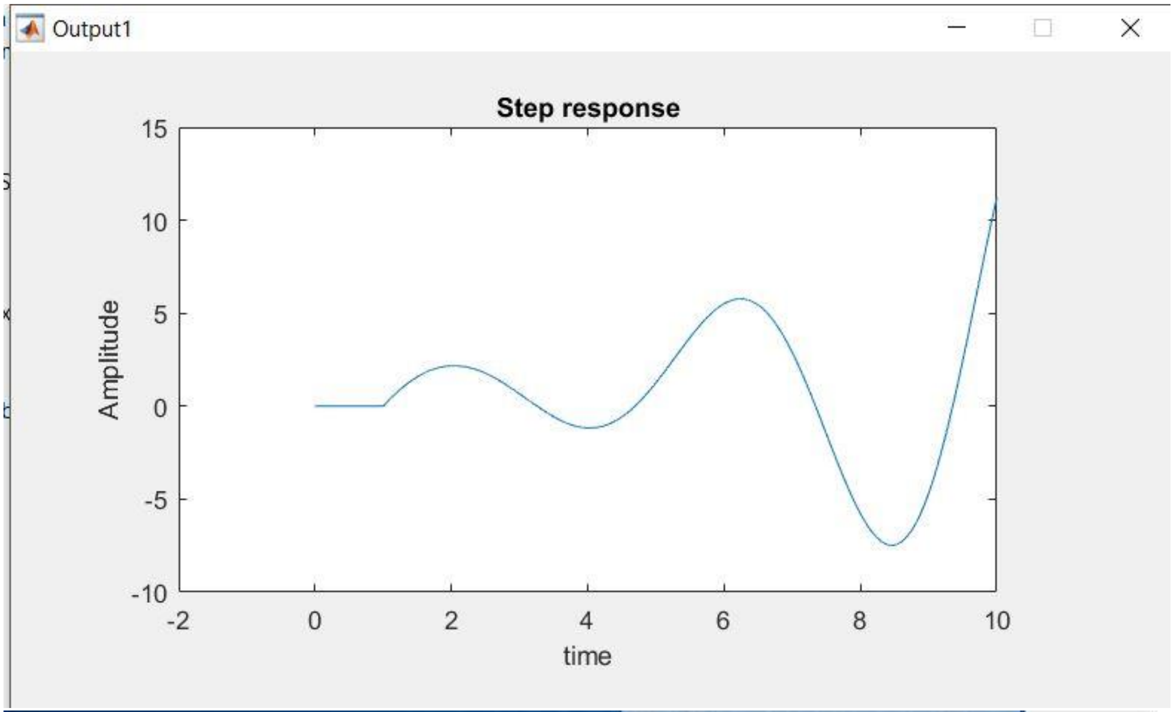


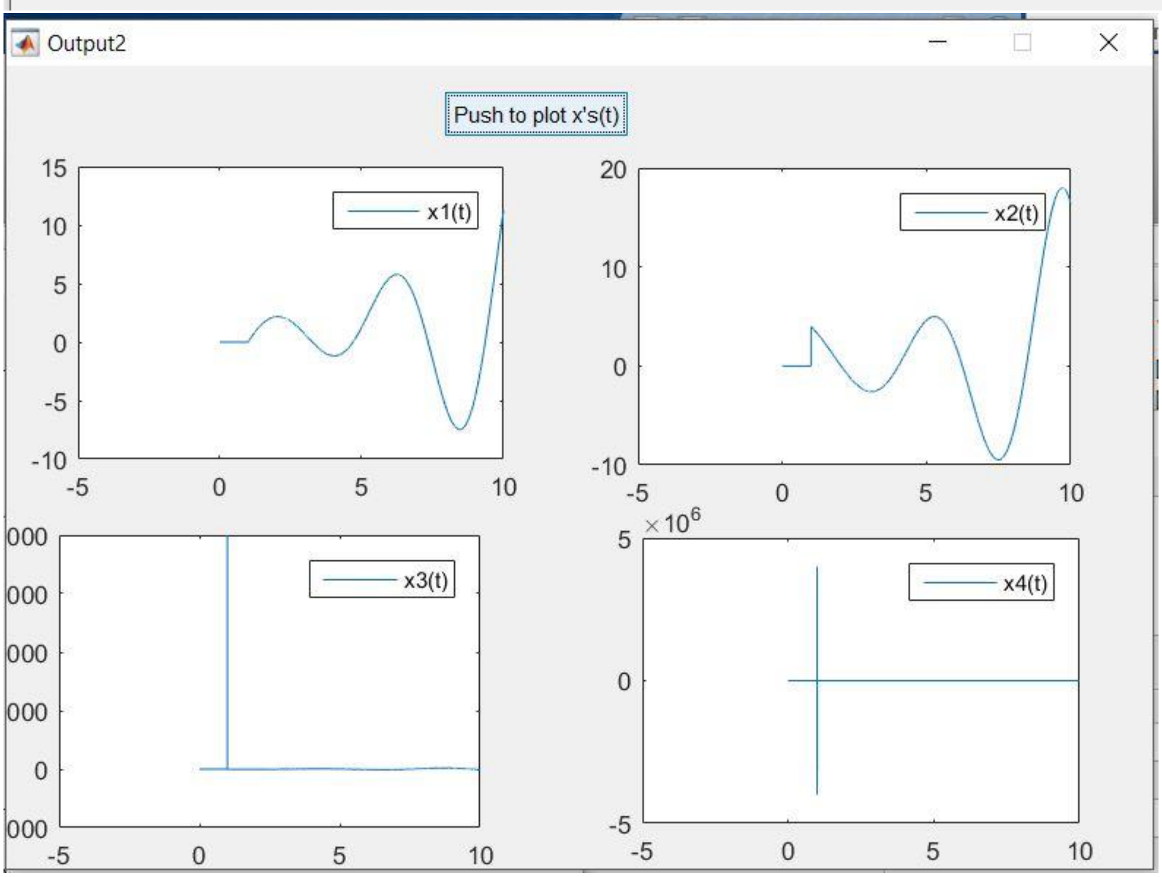


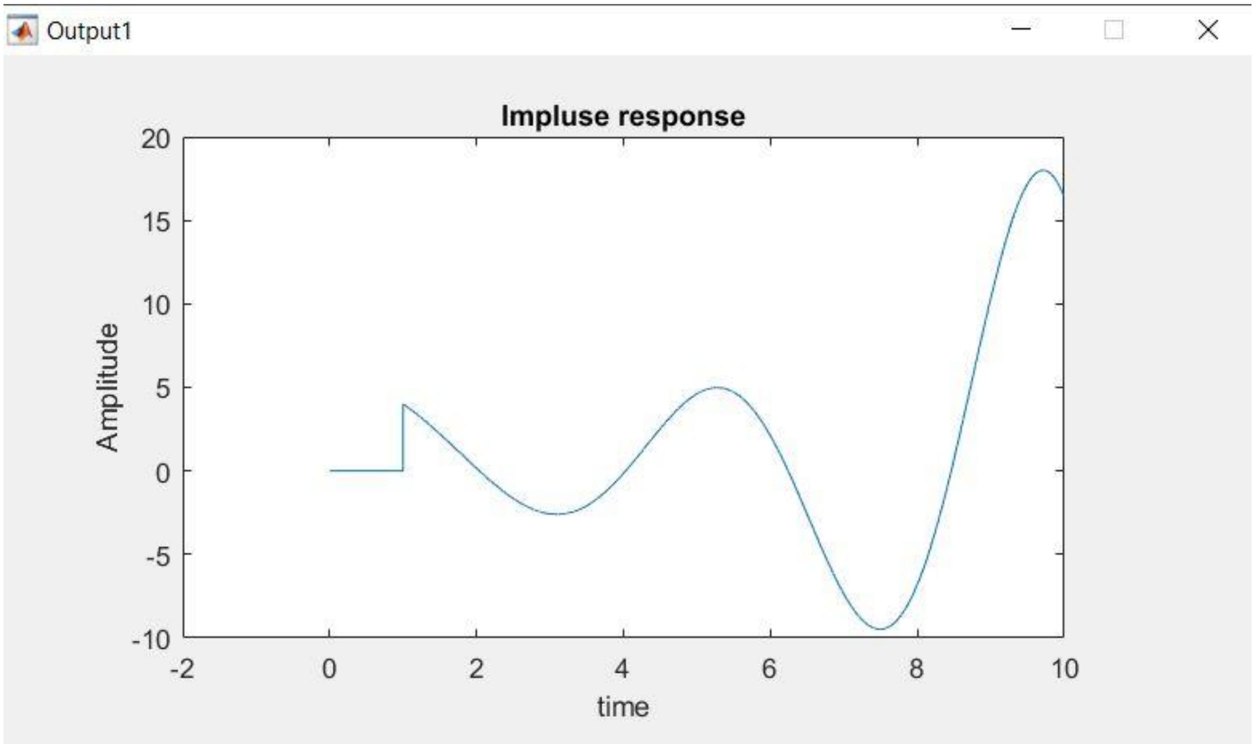
Command Window

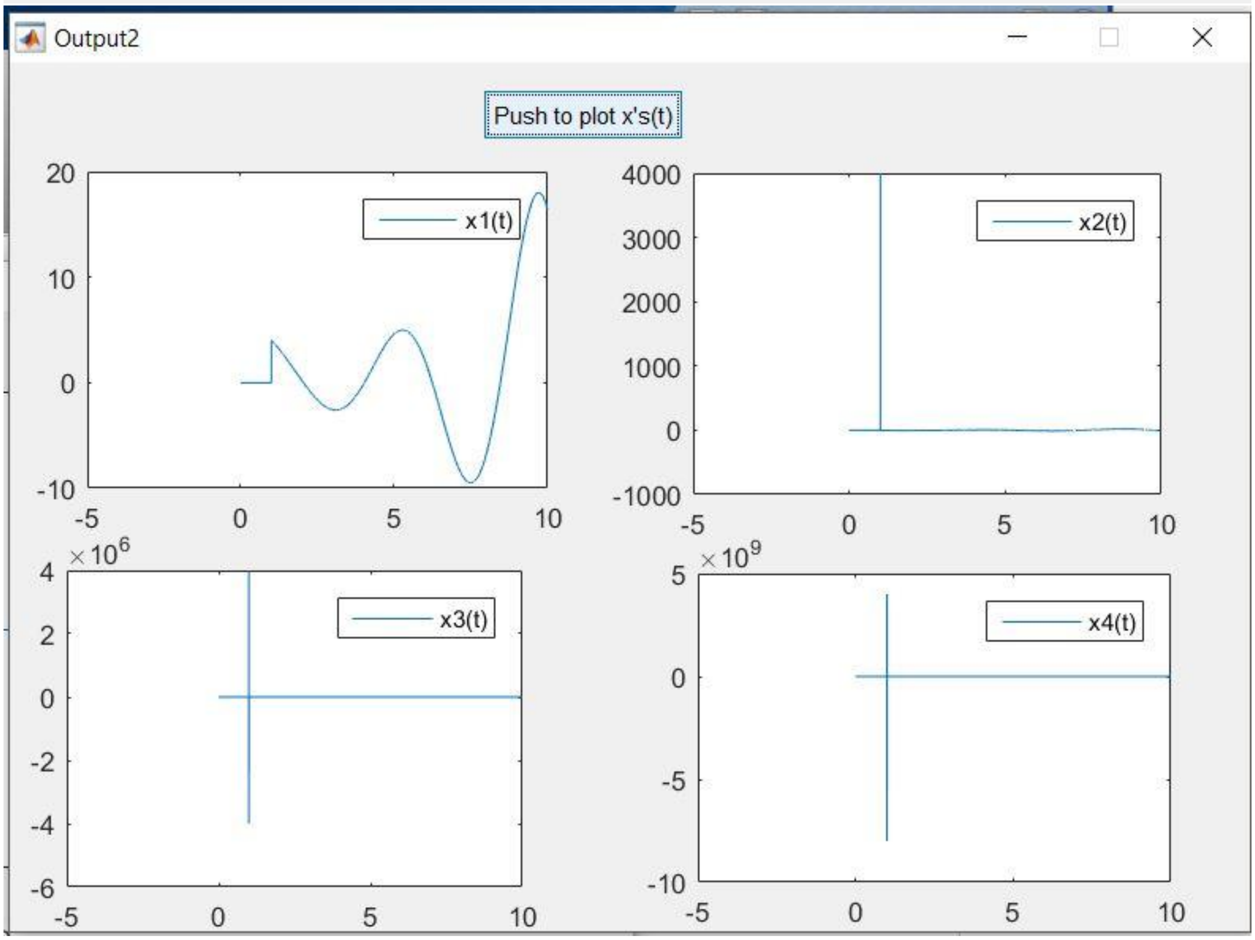
```
A =

     0     1     0     0
     0     0     1     0
     0     0     0     1
    -5    -4    -3    -2


B =

     0
     0
     0
     1


C =

     6     3     5     1


D =

     0
```

$$\frac{d^4y(t)}{dt} + 2\,\frac{d^3y(t)}{dt} + 3\,\frac{d^2y(t)}{dt} + 4\,\frac{dy(t)}{dt} + 5y(t) = 6u(t) + 3\,\frac{du(t)}{dt} + 5\,\frac{d^2u(t)}{dt} + 4\,\frac{d^3u(t)}{dt}$$

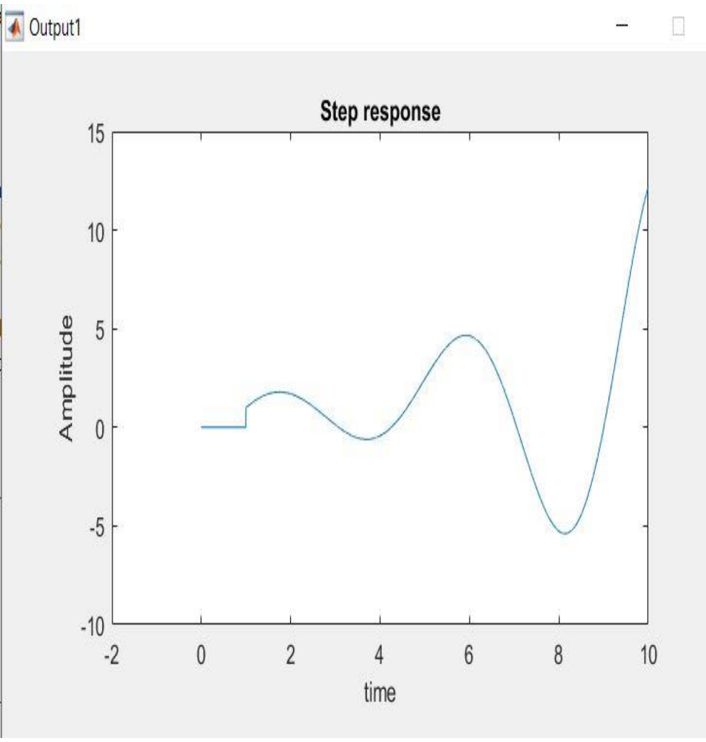


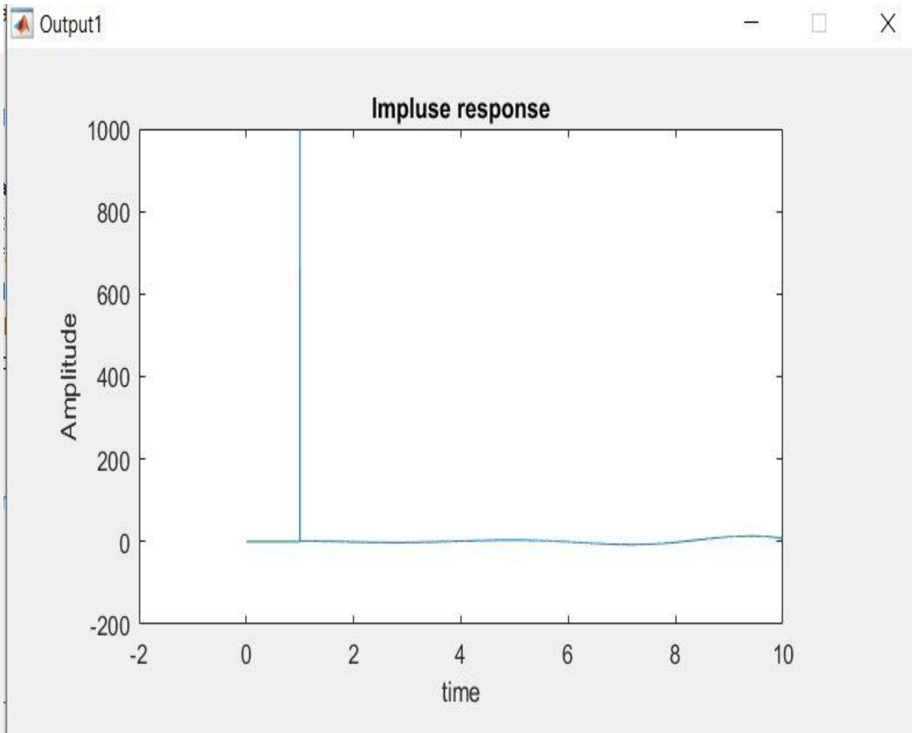


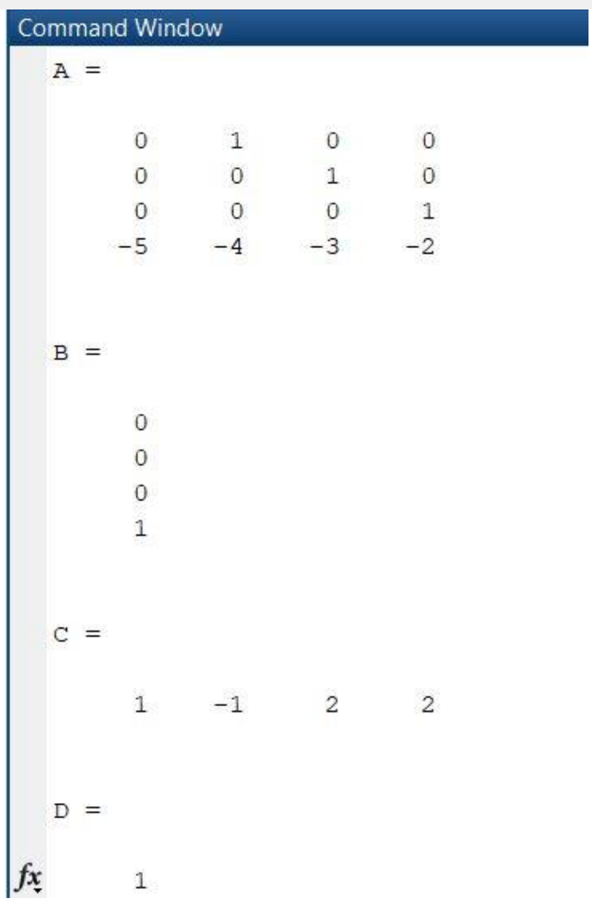

Output1
Impluse response
Amplitude
time
20
15
10
5
0
-5
-10
-2
0
2
4
6
8
10


Output2
Push to plot x's(t)
x1(t)
x2(t)
x3(t)
x4(t)
× 10^6
× 10^9

$$\frac{d^4y(t)}{dt} + 2\frac{d^3y(t)}{dt} + 3\frac{d^2y(t)}{dt} + 4\frac{dy(t)}{dt} + 5y(t) = 6\text{u(t)} + 3\frac{du(t)}{dt} + 5\frac{d^2u(t)}{dt} + 4\frac{d^3u(t)}{dt} + \frac{d^4u(t)}{dt}$$

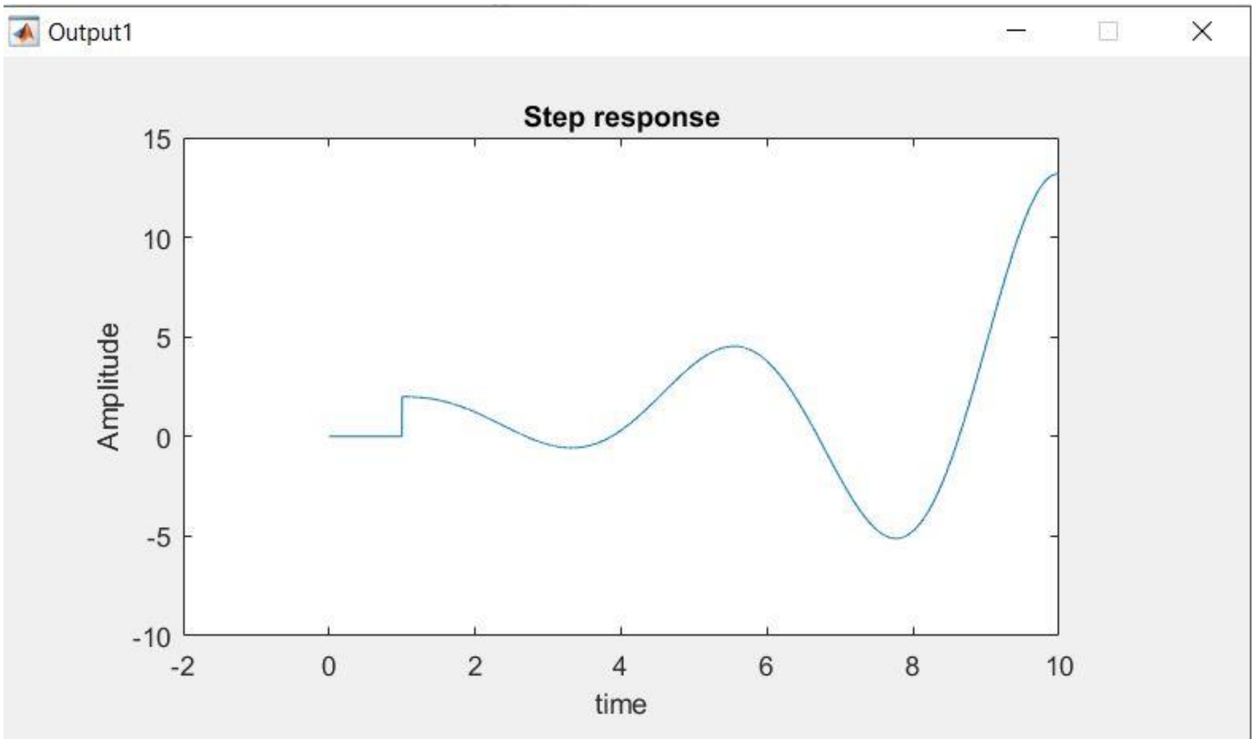


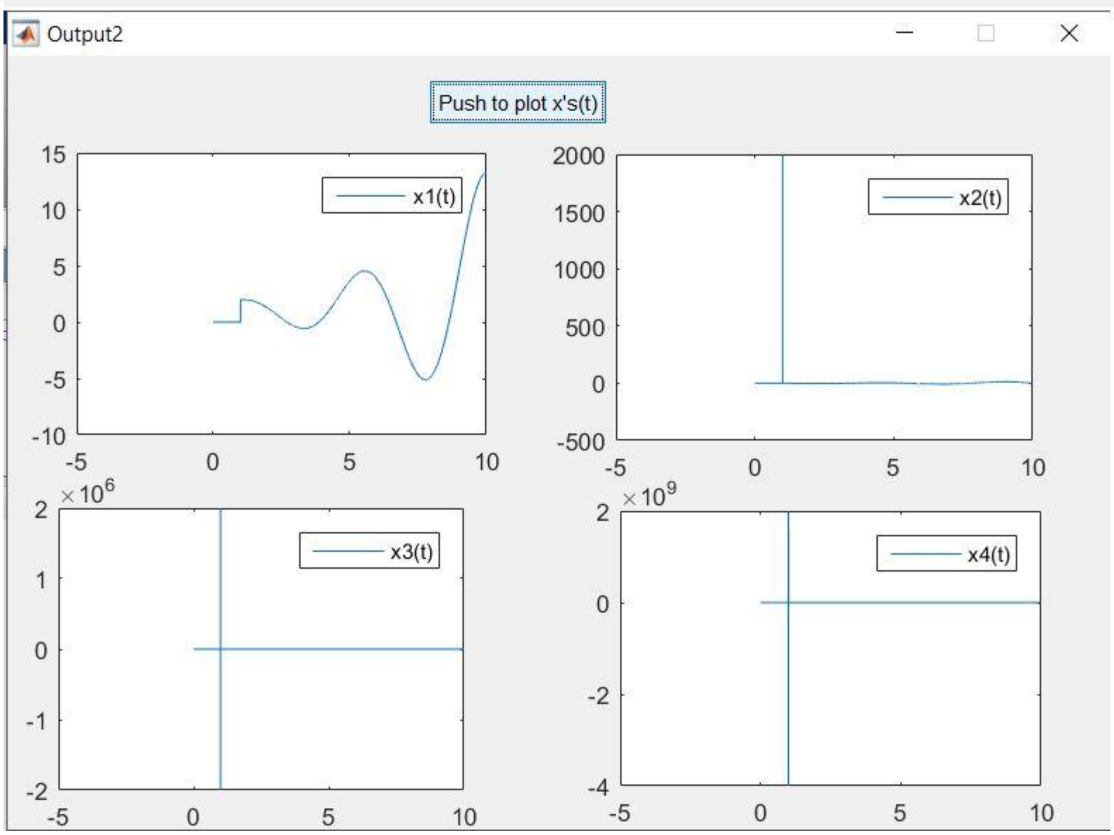


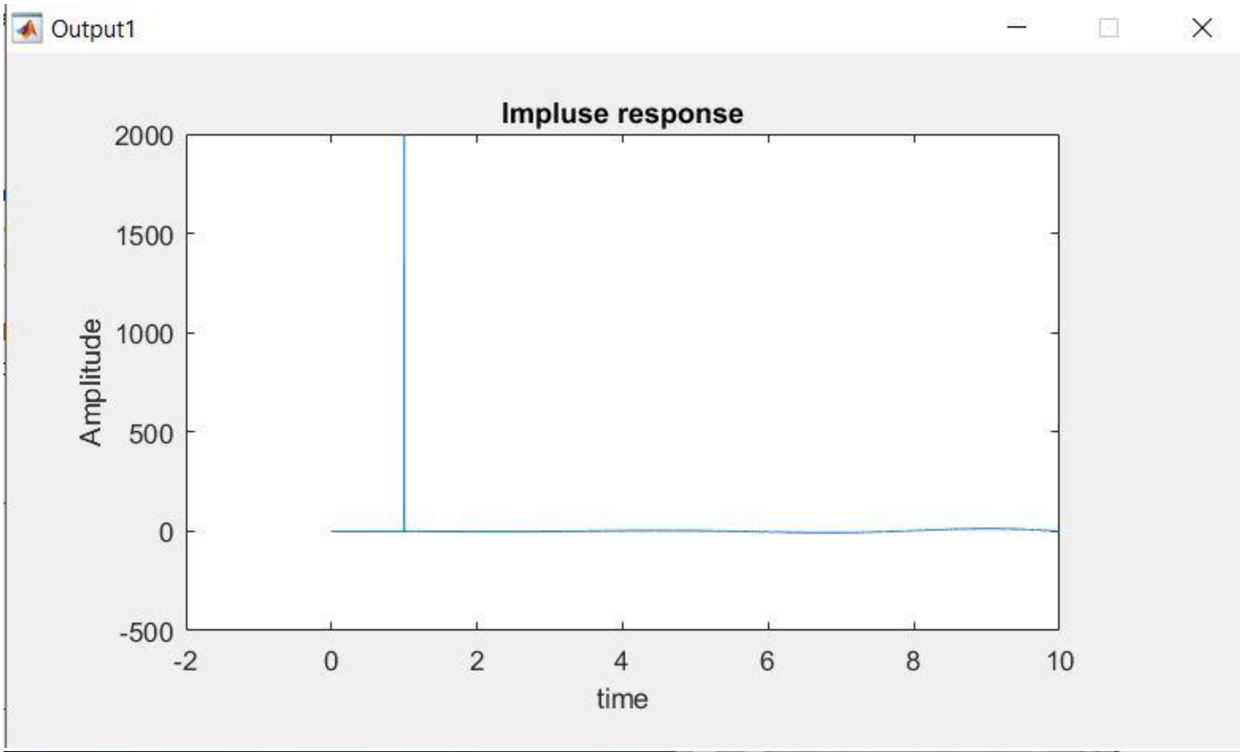


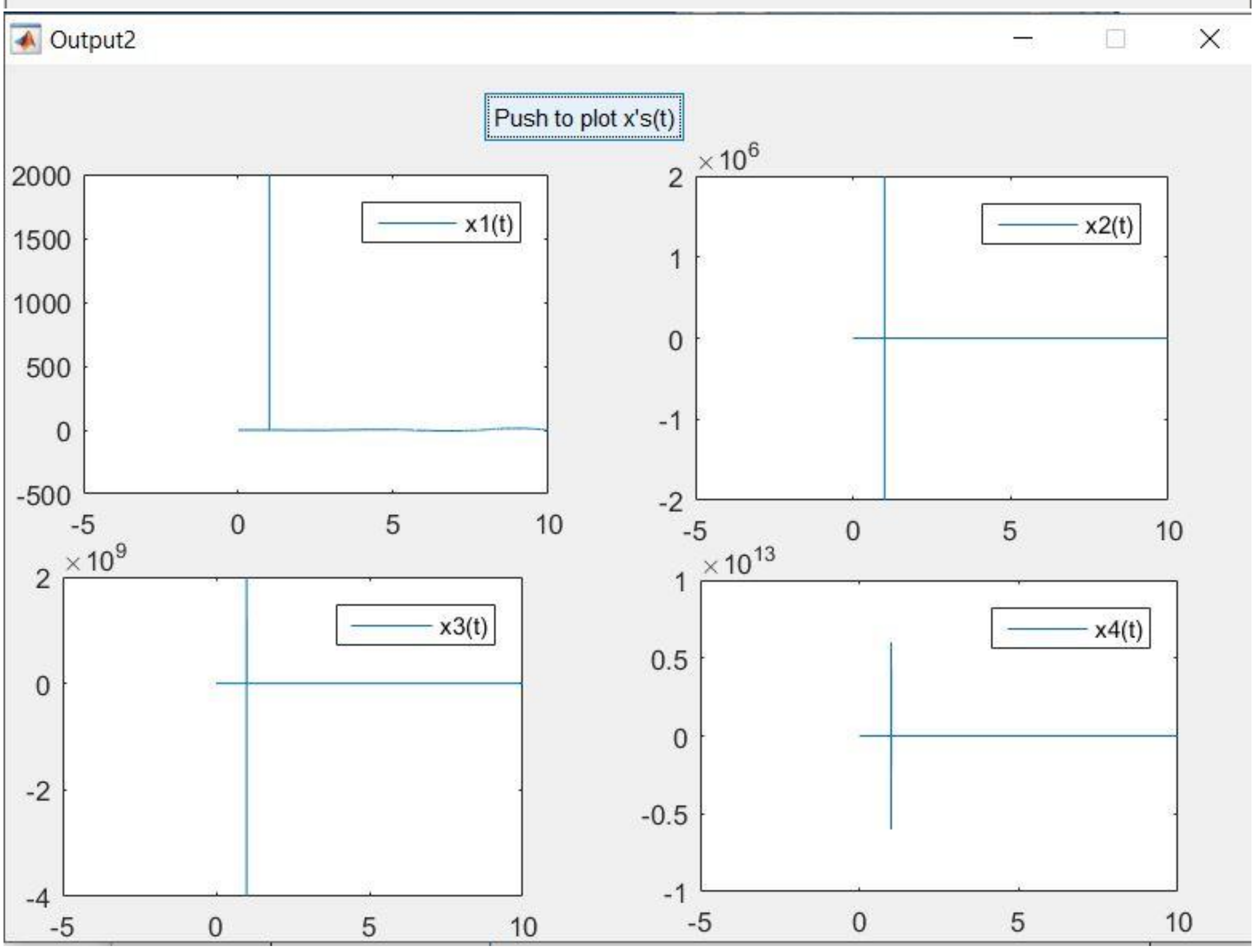
```
Command Window
A =

     0     1     0     0
     0     0     1     0
     0     0     0     1
    -5    -4    -3    -2


B =

     0
     0
     0
     1


C =

     1    -1     2     2


D =

     1
```

$$\frac{d^4y(t)}{dt} + 2\,\frac{d^3y(t)}{dt} + 3\,\frac{d^2y(t)}{dt} + 4\,\frac{dy(t)}{dt} + 5y(t) = 6u(t) + 3\,\frac{du(t)}{dt} + 5\,\frac{d^2u(t)}{dt} + 4\,\frac{d^3u(t)}{dt} + 2\,\frac{d^4u(t)}{dt}$$






Output1
Impluse response
Amplitude
time
2000
1500
1000
500
0
-500
-2
0
2
4
6
8
10


Output2
Push to plot x's(t)
x1(t)
x2(t)
x3(t)
x4(t)
$\times 10^6$
$\times 10^9$
$\times 10^{13}$

list Of References:

1. Cannon, R.H., *Dynamics of Physical Systems*, McGraw-Hill, 1967.
2. Close, C.M., Frederick, D.K., & Newell, J.C., *Modeling and Analysis of Dynamic Systems*, Wiley, 2001.
3. Cochin, I., & Cadwallender, W., *Analysis and Design of Dynamic Systems,* Addison-Wesley, 1997.
4. Ogata, K., *System Dynamics*, Prentice-Hall, 2004.
5. Bracewell, R.N., *The Fourier Transform and Its Application* McGraw-Hill, 1986.

appendix:

Final.m file:

```
function varargout = final(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @final_OpeningFcn, ...
                   'gui_OutputFcn',  @final_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 final is made visible.
function final_OpeningFcn(hObject, eventdata, handles, varargin)
```

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

% Update handles structure
guidata(hObject, handles);




% --- Outputs from this function are returned to the command
line.
function varargout = final_OutputFcn(hObject, eventdata,
handles)
% Get default command line output from handles structure
varargout{1} = handles.output;



function title_Callback(hObject, eventdata, handles)



% --- Executes during object creation, after setting all
properties.
function title_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all
properties.
function edit2_CreateFcn(hObject, eventdata, handles)

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)

a=getappdata(0,'a');
b=getappdata(0,'b');
n=getappdata(0,'n');
m=getappdata(0,'m');
unit = get(handles.unit,'Value');
impluse = get(handles.impluse,'Value');
% we will set 1 to refer to unit step and 0 to refer to unit
impluse
if (unit ==1)
    type =1;
    assignin('base','type',type);
end
if(impluse == 1)
     type =0;
    assignin('base','type',type);
end
if(n<m)
    msgbox('m must be less or equal to n');
elseif((n+1)~=length(a))
    msgbox('a values is not equal to n');
elseif((m+1)~=length(b))
    msgbox('b values is not equal to m');
else
   assignin('base','a',a);
   assignin('base','b',b);
   assignin('base','n',n);
   assignin('base','m',m);
   input;
end




function a_values_Callback(hObject, eventdata, handles)

A_values = get(handles.a_values, 'String');
flag = 0;
a = double.empty;
for i = 1 : length(A_values)

    if (A_values(i)<0 & A_values(i)>9 & A_values(i) ~= ',' &
A_values(i) ~= '.')
        flag = flag +1;
```

```
    end
    if (i == length(A_values))
        continue;
    end
    if(A_values(i) == ',' & A_values(i+1) == ',')
       flag = flag +1;
    end
    if(A_values(i) == '.' & A_values(i+1) == '.')
       flag = flag +1;
    end
end
if (flag ~= 0)
    msgbox('Invalid syntax');
else
    i =1;
    while ( i < length(A_values))

        s = A_values(i);
        if(i ~= length(A_values))
           i = i+1;
        else
            break;
        end
        while(A_values(i)~= ',')
             s = strcat(s,A_values(i));
             if(i ~= length(A_values))
                 i = i+1;
             else
                break;
             end
        end
        s = str2double(s);
        a = [a, s];
    end
end
setappdata(0,'a',a);

% --- Executes during object creation, after setting all
properties.
function a_values_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function b_values_Callback(hObject, eventdata, handles)
m=getappdata(0,'m');
B_values = get(handles.b_values, 'String');
flag = 0;
b = double.empty;
for i = 1 : length(B_values)
    if(B_values(i) == ',')
            continue;
    end
    if (B_values(i)<0 & B_values(i)>9 & B_values(i) ~= ',')
        flag = flag +1;
    end
    if (i == length(B_values))
        continue;
    end
    if(B_values(i) == ',' & B_values(i+1) == ',')
       flag = flag +1;
    end
end
if (flag ~= 0)
    msgbox('Invalid syntax');
else
    i =1;
    if(length(B_values)==1)
       s = B_values(i);
       s = str2double(s);
       b = [b, s];
    else
    while ( i < length(B_values))
        s = B_values(i);
        if(i ~= length(B_values))
           i = i+1;
        else
             s = str2double(s);
             b = [b, s];
            break;
        end
        while(B_values(i)~= ',')
             s = strcat(s,B_values(i));
             if(i ~= length(B_values))
                 i = i+1;
             else
                break;
             end
        end
        s = str2double(s);
```

```
        b = [b, s];
    end
    end
end
setappdata(0,'b',b);

function b_values_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function valueof_n_Callback(hObject, eventdata, handles)

n = str2double(get(handles.valueof_n, 'String'));
if(n<=0 | n>4)
    msgbox('We only solve to the Forth order')
else
setappdata(0,'n',n);
end

function valueof_n_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function valueof_m_Callback(hObject, eventdata, handles)

m = str2double(get(handles.valueof_m, 'String'));
if(m<0 | m>4)
    msgbox('We only solve to the Forth order')
else
setappdata(0,'m',m);
end

function valueof_m_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end
```

Input.m file:

```
function varargout = input(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @input_OpeningFcn, ...
                   'gui_OutputFcn',  @input_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



function input_OpeningFcn(hObject, eventdata, handles, varargin)

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

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command
line.
function varargout = input_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;
```

```matlab
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
type=evalin('base','type');
a=evalin('base','a');
b=evalin('base','b');
m=evalin('base','m');
h = 0.001 ; %time step
t = 0-h : h : 10;
diff1 = zeros(size(t));
diff2 = zeros(size(t));
diff3 = zeros(size(t));
diff4 = zeros(size(t));

if(type == 1) %Step
   % code to generate the step function
   step = ones(size(t));
   step(1:1000) = 0;

   if(m>=1)
     for i = 1 : length(t)-1
       diff1(i) = (step(i+1) - step(i))/h;
     end
   end
else
    %code to generate the unit impluse
    impluse = zeros(size(t));
    impluse(1001) = 1/h;
    if(m>=1)
     for i = 1 : length(t)-1
       diff1(i) = (impluse(i+1) - impluse(i))/h;
     end
    end
end
   if(m>=2)
       for i = 1 : length(t)-1
           diff2(i) = (diff1(i+1) - diff1(i))/h;
       end
   end

   if(m>=3)
       for i = 1 : length(t)-1
         diff3(i) = (diff2(i+1) - diff2(i))/h;
       end
   end
```

```
   if(m>=4)
       for i = 1 : length(t)-1
         diff4(i) = (diff3(i+1) - diff3(i))/h;
       end
   end



Btmp = b/ a(end); %coefficients of the input
if (length(Btmp) ~= 5)
   for i = (length(Btmp)+1) : 5
        Btmp = [Btmp, 0];
    end
end

if(type == 1)
    input = Btmp(1)*step + Btmp(2)*diff1 + Btmp(3)*diff2 +
Btmp(4)*diff3 + Btmp(5)*diff4;
    plot(t,step);
else
    input = Btmp(1)*impluse + Btmp(2)*diff1 + Btmp(3)*diff2 +
Btmp(4)*diff3 + Btmp(5)*diff4;
    plot(t,impluse);
end


assignin('base','input',input);
xlabel('time');
ylabel('input');
```

Output1.m file:

```
function varargout = Output1(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Output1_OpeningFcn, ...
                   'gui_OutputFcn',  @Output1_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 Output1 is made visible.
function Output1_OpeningFcn(hObject, eventdata, handles,
varargin)
% Choose default command line output for Output1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);



% --- Outputs from this function are returned to the command
line.
function varargout = Output1_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
```

```matlab
a=evalin('base','a');
b=evalin('base','b');
n=evalin('base','n');
m=evalin('base','m');
type=evalin('base','type');
input=evalin('base','input');
% We use Controllable Canonical Form (CCF)
Atmp = a/ -a(end);
Btmp = b/ a(end);


%A Matrix :
    tmp1 = Atmp(2:end-1);
    tmp2 = eye(n-1);
    tmp3 = [tmp2;tmp1];
    tmp4 = zeros(n,1);
    tmp4(n,1)= Atmp(1);
    A = [tmp4,tmp3]

%B Matrix :
    B = zeros(n,1);
    B(n,1) = 1


%C & D Matrix :
    if(n==m)
        C = zeros(1,n);
        Atmp = Atmp * -1;
        for i= 1 : n
            C(i) = Btmp(i)-(Atmp(i)*Btmp(end));
        end
        C
        D = Btmp(end)
    end

    if(n>m)
       C =  zeros(1,n);
      for i = 1 : length(Btmp)
          C(i) = Btmp(i);
      end
      C
      D = 0
    end

h = 0.001 ; %time step
t =  0-h : h : 10;
    q = zeros(1,n);
```

```
    y_values = zeros(n,length(t));
    y_values(:,1) = q;
    for i = 1 : (length(t)-1)
              k = A*y_values(:,i) + B*input(i);
              y_values(:,i+1) = y_values(:,i) + k*h;
    end
    assignin('base','y_values',y_values);
    plot(t,y_values(1,:));
    if(type==1)
        title('Step response');
    else
        title('Impluse response');
    end
    xlabel('time');
    ylabel('Amplitude');



function pushbutton2_Callback(hObject, eventdata, handles)

Output2
```

Output2.m file:

```
function varargout = Output2(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Output2_OpeningFcn, ...
                   'gui_OutputFcn',  @Output2_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 Output2 is made visible.
function Output2_OpeningFcn(hObject, eventdata, handles,
varargin)

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command
line.
function varargout = Output2_OutputFcn(hObject, eventdata,
handles)

varargout{1} = handles.output;
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)

y_values=evalin('base','y_values');
n = evalin('base','n');
h = 0.001 ; %time step
t = 0-h : h : 10;

if(n==1)
    axes(handles.axes1);
    plot(t,y_values(1,:));
    legend('x1(t)');
elseif(n==2)
  axes(handles.axes1);
  plot(t,y_values(1,:));
  legend('x1(t)');
  axes(handles.axes2);
  plot(t,y_values(2,:));
  legend('x2(t)');
elseif(n==3)
   axes(handles.axes1);
  plot(t,y_values(1,:));
  legend('x1(t)');
  axes(handles.axes2);
  plot(t,y_values(2,:));
  legend('x2(t)');
  axes(handles.axes3);
  plot(t,y_values(3,:));
  legend('x3(t)');
elseif(n==4)
  axes(handles.axes1);
  plot(t,y_values(1,:));
  legend('x1(t)');
  axes(handles.axes2);
  plot(t,y_values(2,:));
  legend('x2(t)');
  axes(handles.axes3);
  plot(t,y_values(3,:));
  legend('x3(t)');
  axes(handles.axes4);
  plot(t,y_values(4,:));
  legend('x4(t)');
end
```