Signals and Systems Laboratory (EC2P002)

EXPERIMENT-3

Shorya Sharma

19EE01017

Aim of the experiment:

- Demonstrate the convolution of two signals and verify the properties.
- Demonstrate the auto-correlation and cross-correlation of two signals and verify the properties.

Waveforms to be Observed:

PART- A

1. Determine the output $y(t)$ if the input and impulse response of the linear time invariant are given as $x(t) = e^{-t}u(t)$ and $h(t) = e^{-2t}u(t)$
2. Calculate $y(t) = x(t) * h(t)$ where $x(t) = \sin(\pi t)(u(t) - u(t-1.5))$ and $h(t) = 1.5\big(u(t) - u(t-1.5)\big) - u(t-2) + u(t-2.5)$

PART- B

Perform the convolution operation for the following sets of input signals:

1. $x[n] = u[n] - u[n-8]$ $\quad h[n] = \sin\left(\frac{2\pi n}{8}\right)(u[n] - u[n-8])$
2. $x[n] = (0.8)^n u[n]$ $\quad h[n] = (0.3)^n u[n]$
3. $x[n] = (e)^{-n} u[n]$ $\quad h[n] = (2)^{-n} u[n]$

Theory: -

Convolution:

In continous time domain, convolutions of two functions $x(t)$ and $h(t)$ is defined as follows.

$$x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

In discrete time domain, convolution of two functions $x[n]$ and $h[n]$ is defined as follows.

$$x[n] * h[n] = \sum_{n=-\infty}^{\infty} x[k]h[n-k]$$

Properties of the Convolution:

1. **Commutative Property:**

$$x_1(t) * x_2(t) = x_2(t) * x_1(t)$$
$$x_1[n] * x_2[n] = x_2[n] * x_1[n]$$

2. **Distributive Property:**

$$x_1(t) * [x_2(t) + x_3(t)] = x_1(t) * x_2(t) + x_1(t) * x_3(t)$$
$$x_1[n] * [x_2[n] + x_3[n]] = x_1[n] * x_2[n] + x_1[n] * x_3[n]$$

3. **Associative Property:**

$$x_1(t) * [x_2(t) * x_3(t)] = [x_1(t) * x_2(t)] * x_3(t)$$
$$x_1[n] * [x_2[n] * x_3[n]] = [x_1[n] * x_2[n]] * x_3[n]$$

4. **Shift Property:**

If $x_1(t) * x_2(t) = y(t)$ then $x_1(t - T_1) * x_2(t - T_1) = y(t - T_1 - T_2)$
If $x_1[n] * x_2[n] = y[n]$ then $x_1[n - N_1] * x_2[n - N_2] = y[n - N_1 - N_2]$

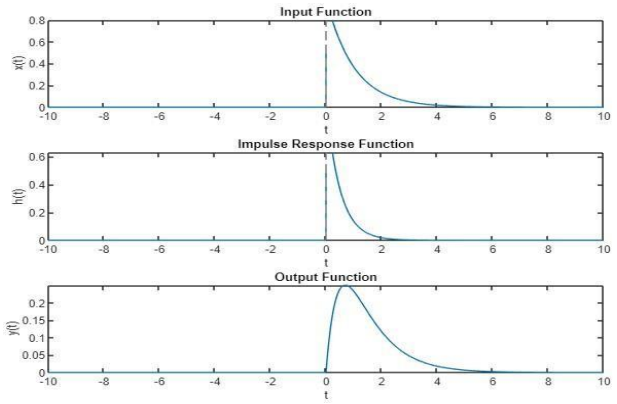
5. **Scaling Property:**

If $x_1(t) * x_2(t) = y(t)$ then $y(at) = |a|x_1(at) * x_2(at)$
If $x_1[n] * x_2[n] = y[n]$ then $y[an] = |a|x_1[an] * x_2[an]$

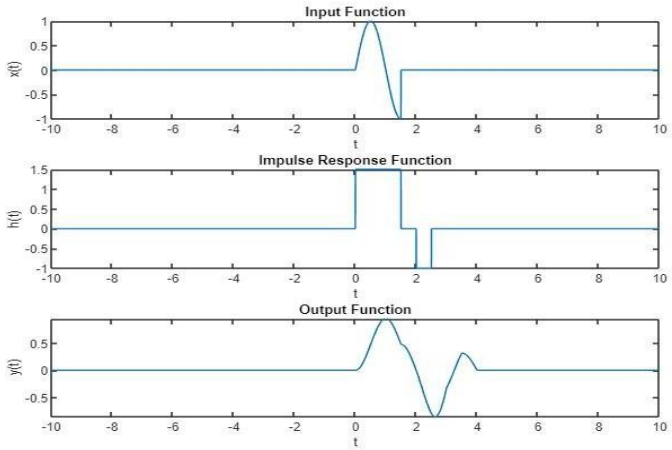
Results:

PART-A

1) $x(t)=e^{-t}u(t)$, $h(t)=e^{-2t}u(t)$

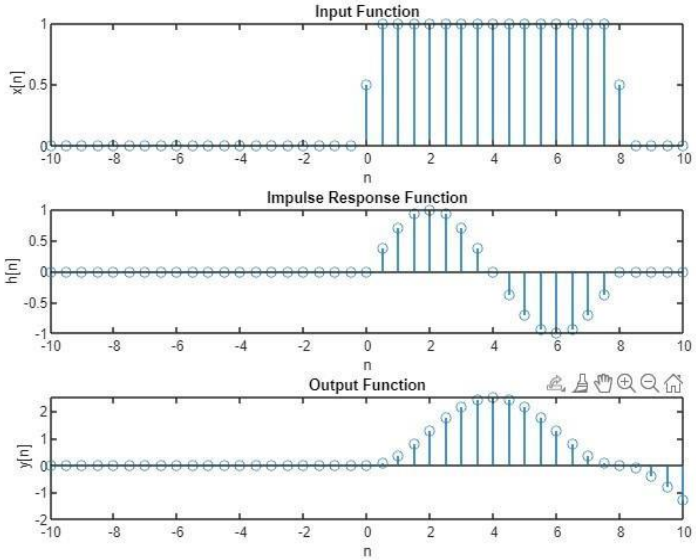


2) $x(t)=\sin(\pi t)(u(t)-u(t-1.5))$,$h(t)=1.5(u(t)-u(t-1.5))-u(t-2)+u(t-2.5)$

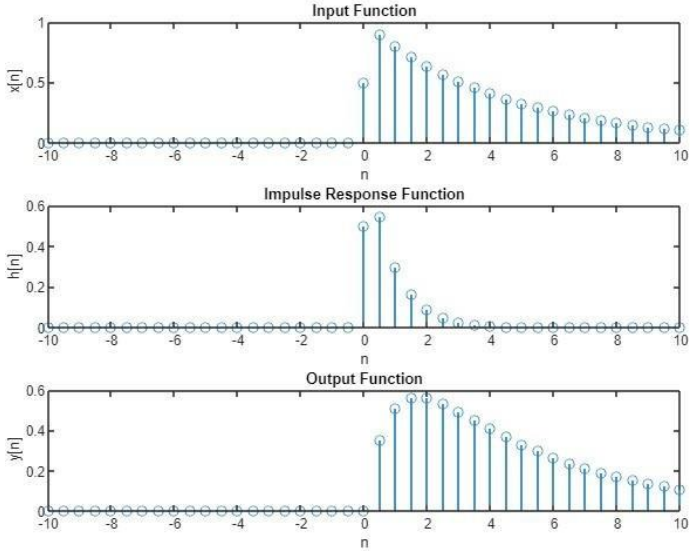


PART-B

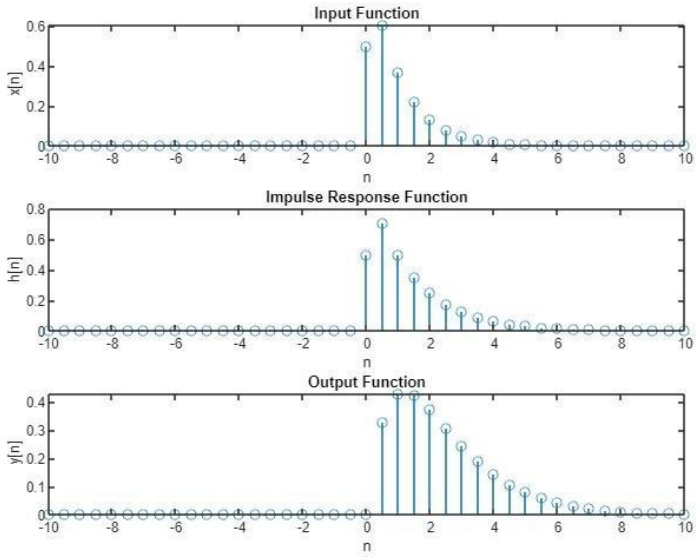
1) $x[n] = u[n] - u[n-8]$, $h[n] = \sin(2\pi n/8)(u[n] - u[n-8])$



2) $x[n] = (0.8)^n u[n]$ $h[n] = (0.3)^n u[n]$



3) $x[n] = (e)^{-n}u[n]$, $h[n] = (2)^{-n}u[n]$



Discussion:

We can see many real-life examples of convolution. In acoustics, reverberation is the convolution of original sound with the echoes. In digital image processing, convolution filtering plays an important role in many algorithms in edge detection and related processes.

Conclusion:

Hence by using MATLAB we find the input-output response of a system. Convolution is a mathematical way of combining two signals to form a third signal. It is the single most important technique in Digital Signal Processing. Using the strategy of impulse decomposition, systems are described by a signal called the impulse response. Convolution is important because it relates the three signals of interest: the input signal, the output signal, and the impulse response. Convolution can be presented from two different viewpoints, called the input side algorithm and the output side algorithm. Convolution provides the mathematical framework for DSP

Appendix

Part-A

1) $x(t)=e^{-t}u(t)$, $h(t)=e^{-2t}u(t)$

```
clc;
close all;
%  ?(?) = ?^??.*?(?) and ?(?) = ?^?2?.*?(?)
syms t k ;
x(t)=exp(-t).*heaviside(t);
h(t)=exp(2*(-t)).*heaviside(t);
y(t)= int(x(k)*h(t-k),k,-inf,inf); %convoluting based on formula
subplot(3,1,1);
fplot(t,x),title("input function"),xlim([-5 5]),grid on;
subplot(3,1,2);
fplot(t,h),title("impulse function");
subplot(3,1,3);
fplot(t,y), title("convution of x and h")

% properties of convolution
% commutative ?1(?)??2(?) = ?2(?)?x1(?)
if (int(x(k)*h(t-k),k,-inf,inf)==int(x(t-k)*h(k),k,-inf,inf))
    disp("it is commutative");
end
% distributive ?1(?)?[?2(?)+?3(?)] = ?1(?)??2(?)+?1(?)??3(?)
s(t)=heaviside(t);%test signal for properties checking
y1=int(x(k)*(s(t-k)+h(t-k)),k,-inf,inf);
y2=int(x(k)*s(t-k)+x(k)*h(t-k),k,-inf,inf);
if (y1==y1)
   disp("it is distributive");
end
%associative
y3=int(x(k)*(s(t-k)*h(t-k)),k,-inf,inf);
y4=int(x(t-k)*(s(t-k)*h(k)),k,-inf,inf);
if (y3==y4)
    disp("it is associative");
```

```
end
%shifting
g=-5:0.1:5;
y5(t)=int(x(k-0.2)*h(t-k-0.1),k,-inf,inf);
y6(t)=subs(y,t,t-0.3);
d1=zeros(1,length(g)); d2=zeros(1,length(g));
for i=1:length(g)
    d1(i)=d1(i)+y5(i);
    d2(i)=d2(i)+y6(i);
end
if (d1==d2)
    disp("It follows shifting");
end
% scaling
y7(t)=2*int(x(2*k)*h(2*(t-k)),k,-inf,inf);
y8(t)=subs(y,t,2*t);
d3=zeros(1,length(g)); d4=zeros(1,length(g));
for i=1:length(g)
    d3(i)=d3(i)+y5(i);
    d4(i)=d4(i)+y6(i);
end
if (d3==d4)
    disp("It follows scaling");
end
```

2) x(t)=sin(πt)(u(t)−u(t−1.5)),h(t)=1.5(u(t)−u(t − 1.5))−u(t − 2)+u(t−2.5)

```
clc;
close all;
%  ?(?) = sin(??)(?(?) ? ?(? ? 1.5))
%?(?) = 1.5(?(?) ? ?(? ? 1.5)) ? ?(? ? 2) + ?(? ? 2.5
syms t k;
x(t)=sin(pi*t).*(heaviside(t)-heaviside(t-1.5));
h(t)=1.5.*(heaviside(t)-heaviside(t-1.5))-heaviside(t-2)+heaviside(t-2.5);
y=int(x(k)*h(t-k),k,-inf,inf); %convoluting based on formula
subplot(3,1,1);
fplot(t,x),title("input function");
subplot(3,1,2);
fplot(t,h),title("impulse function");
subplot(3,1,3);
fplot(t,y), title("convution of x and h")

% properties of convolution
% commutative ?1(?)??2(?) = ?2(?)?x1(?)
if (int(x(k)*h(t-k),k,-inf,inf)==int(x(t-k)*h(k),k,-inf,inf))
    disp("it is commutative");
end
% distributive ?1(?)?[?2(?)+?3(?)] = ?1(?)??2(?)+?1(?)??3(?)
s(t)=heaviside(t);%test signal for properties checking
y1=int(x(k)*(s(t-k)+h(t-k)),k,-inf,inf);
y2=int(x(k)*s(t-k)+x(k)*h(t-k),k,-inf,inf);
if (y1==y1)
   disp("it is distributive");
end
%associative
y3=int(x(k)*(s(t-k)*h(t-k)),k,-inf,inf);
y4=int(x(t-k)*(s(t-k)*h(k)),k,-inf,inf);
if (y3==y4)
    disp("it is associative");
end
%shifting
g=-5:0.1:5;
y5(t)=int(x(k-0.2)*h(t-k-0.1),k,-inf,inf);
y6(t)=subs(y,t,t-0.3);
d1=zeros(1,length(g)); d2=zeros(1,length(g));
for i=1:length(g)
    d1(i)=d1(i)+y5(i);
    d2(i)=d2(i)+y6(i);
end
if (d1==d2)
    disp("It follows shifting");
end
% scaling
y7(t)=2*int(x(2*k)*h(2*(t-k)),k,-inf,inf);
y8(t)=subs(y,t,2*t);
d3=zeros(1,length(g)); d4=zeros(1,length(g));
for i=1:length(g)
    d3(i)=d3(i)+y5(i);
    d4(i)=d4(i)+y6(i);
end
if (d3==d4)
    disp("It follows scaling");
end
```

Part-B

Function for part-B

```
function z = discrete (x,h)
syms tau;
syms n;
k = -10:0.5:10;
subplot(3,1,1);
stem(k,x(k));
xlabel('n');ylabel('x[n]');title('Input Function vs t');
```

```
subplot(3,1,2);
stem(k,h(k));
xlabel('n');ylabel('h[n]');title('Impulse Response Function vs t');
subplot(3,1,3);
y1 = int(x(tau)*h(n-tau),tau,-inf,inf);
y = zeros(size(k));
for i =1:length(k)
y(i)= subs(y1,n,k(i));
end
stem(k,y);
xlabel('n');ylabel('y[n]'); title('Output Function vs t');
% Commutative Property Verification
y1_= int(x(tau)*h(n-tau),tau,-inf,inf);
y2 = int(h(n-tau)*x(tau),tau,-inf,inf);
if y1_ == y2
disp('Commutative');
else
disp('Not Commutative');
end
% Distributive Property Verification
h1(n) = heaviside(n);
y3 = int((x(tau)+h1(tau))*h(n-tau),tau,-inf,inf);
y4 = int(h1(tau)*h(n-tau)+x(tau)*h(n-tau),tau,-inf,inf);
if y3 == y4
disp('Distributive');
else
disp('Not Distributive')
end
% Associative Property Verification

y5 = int(x(tau)*(h(n-tau)*h1(n-tau)),tau,-inf,inf);
y6 = int((x(tau)*h(n-tau))*h1(n-tau),tau,-inf,inf);
if y5 == y6
disp('Associative');
else
disp('Not Associative');
end
% Scaling Property Verifcation
y7 = int(x(tau)*h((n)-tau),tau,-inf,inf);
y8 = 2*int(x(2*tau)*h(2*(n-tau)),tau,-inf,inf);
if y7 == y8
disp('Scaling Property Satisfies')
disp(' ');
else
disp('Scaling Property Donot Satisfy')
disp(' ');
end
```

Code

```
clc
clear all
close all
syms n; %Creating variable n
x(n) = heaviside(n)-heaviside(n-8); %Generating Input Signal-1
h(n) = sin((2*pi*n)/8).*x; %Generating Input Signal-2
figure(1);
discrete(x,h); %Passing x,h to function
x(n)= ((0.8)^n).*heaviside(n); %Generating Input Signal-1
h(n)= ((0.3)^n).*heaviside(n); %Generating Input Signal-2
figure(2);
discrete(x,h); %Passing x,h to function
x(n) = (exp(-n)).*heaviside(n); %Generating Input Signal-1
h(n) = ((2)^(-n)).*heaviside(n); %Generating Input Signal-2
figure(3);
discrete(x,h); %Passing x,h to function
```