**Signals & Systems**

**EEE-223**

Lab # 06



.

|  |  |
| --- | --- |
| Name | Muhammad Haris Irfan |
| Registration Number | FA18-BCE-090 |
| Class | BCE-4A |
| Instructor’s Name | Muhammad Bilal Qasim |

**LAB # 06**

**Analysis of Discrete LTI Systems using Convolution Sum**

**Lab 06-** **Analysis of Discrete LTI Systems using Convolution Sum**

**Pre-Lab Tasks**

**6.1 Discrete Time Convolution:**

An LTI discrete-time system is (equivalently to the continuous-time case) completely describes by impulse response, which is usually denoted by. The impulse response of an LTI discrete-time system is the output of the system when the unit impulse sequence (or Kronecker delta function) is applied as input. Even though the unit impulse input consists of one no-zero term, the impulse response signal  usually consists of more than one non-zero elements. The explanation is that the system is dynamic (with memory); that is, the system responds over various time instances to the output applied at. The knowledge of the impulse responseof a system allows the computation of the response  of the system to any input signal . The output signal is computed by discrete time convolution. The mathematical expression (6.1) of discrete time convolution is



The convolution between two discrete time signals is computed by using the MATLAB command conv.

**6.1.1 The Command conv:**

In MATLAB, the command conv allows the direct computation of the convolution between two signals. In order to illustrate the conv command There are three rules that have to be applied for successful computation of the convolution between two continuous-time signals.

* First rule: Two signals (input and impulse response) should be defined in the same time interval. Hence both the signals are defined in the time interval.
* Second rule: When a signal consists of multiple parts, the time intervals in which each part is defined must not overlap. For example,



Note that the equality at  is placed only in the upper part.

Having defined the input and impulse response signals the response of the system can be computed by convoluting the two signals. The convolution is implemented by the command y=conv(x,h). The response of the system is now computed and the only thing left to do is to plot it. However, the number of elements of the output vectoris not equal to the number of elements of vectorsor .

The precise relationship is length()=length()+length()-1. To overcome this, the third rule must be applied.

* Third rule: The output of the system is plotted in the double time interval of the one in which the output and impulse response signals are defined.

**6.1.2 The Unit Impulse Sequence as Input to a System:**

In this section, the impulse response of a discrete time system is discussed in detail. More specifically, it is established that if the unit impulse sequence  is the input to a system, the output of the system is the impulse response  of the system.

**Example:**

Suppose that a discrete time system is described by the impulse response. Compute the response of the system to the input signal



The response of the system is computed by convoluting the impulse response  with the input signal. Recall that  must be defined in the same time interval to; that is, . For illustration purposes both signals are plotted.

|  |  |  |
| --- | --- | --- |
| Commands | Results | Comments |
| x=[1 0 0 0];  n=0:3;  stem(n,x,'fill','linewidth',2),grid on  axis([-0.2 3.2 -0.1 1.1])  legend('x[n]=\delta[n]') | x[n].bmp | Input signal plotted in . |
| h=[2 4 3 1];  n=0:3;  stem(n,h,'fill','linewidth',2),grid on  axis([-0.2 3.2 -0.1 4.1])  legend('h[n]') | h[n].bmp | Impulse response plotted in . |

The number of elements of the output vector  compared to the number of elements of the input and impulse response vectors. The exact relationship is length ()=length()+length()-1. Therefore the output of the system is plotted in the double time interval from input and impulse response signals.

|  |  |  |
| --- | --- | --- |
| Commands | Results | Comments |
| y=conv(x,h);  stem(0:6,y,'fill','linewidth',2),grid on  axis([-0.2 6.2 -0.1 4.1])  legend('y[n]=h[n]') | y[n].bmp | The outputis computed and plotted for. As expected the output and impulse response signals are the same. |

The output and impulse response signals are the same; thus the identity property  clearly stands. The system under consideration is a linear shift invariant system. If at a linear and shift invariant system, with impulse response, the input signal, i.e., a shifted unit impulse sequence is applied, then the response of the system is also shifted by units.

|  |  |  |
| --- | --- | --- |
| Commands | Results | Comments |
| x=[0 1 0 0];  n=0:3;  stem(n,x,'fill','linewidth',2),grid on  axis([-0.2 3.2 -0.1 1.1])  legend('x[n]=\delta[n-1]') | imshif.bmp | The shifted unit impulse sequence is the input to the system. |
| h=[2 4 3 1];  n=0:3;  stem(n,h,'fill','linewidth',2),grid on  axis([-0.2 3.2 -0.1 4.1])  legend('h[n]') | h[n].bmp | Impulse response, . |
| y=conv(x,h);  stem(0:6,y,'fill','linewidth',2),grid on  axis([-0.2 6.2 -0.1 4.1])  legend('y[n]=h[n-1]') | shifres.bmp | The output of the system is its impulse response shifted by 1 unit to the right. |

As expected, the response  of the system to the input signal  is . More generally, the response of a linear shift invariant system to a shifted unit impulse sequence shifted by  units version of impulse response, i.e., the output of the system is.

**6.1.3 Computation of Discrete Time Convolution:**

The process that has to be followed to analytically derive the convolution sum (equation 6.1) is as follows. First, the input and impulse response signals are plotted in the -axis. One of the two signals is reversed about the amplitude axis and its reflection is shifted from to by changing appropriately the value of. The output of the system is computed from the overlapping values of and  according to the convolution sum. Te convolution procedure for two discrete time signals is demonstrated by using the signals and.

Step 1: The two signals are plotted at the -axis.

|  |  |  |
| --- | --- | --- |
| Commands | Results | Comments |
| kx=[0 1];  x=[1 2];  stem(kx,x,'fill','linewidth',2),grid on  axis([-0.1 3.1 -0.1 2.1]) | kx.bmp | Input signal. |
| kh=0:3;  h=[2 1 1 1];  stem(kh,h,'fill','linewidth',2),grid on  axis([-0.1 3.1 -0.1 2.1])  legend('h[k]') | kh.bmp | Impulse response signal. |

Step 2: The reflected version of, namely, is plotted.

|  |  |  |
| --- | --- | --- |
| Commands | Results | Comments |
| stem(-kh,h,'fill','linewidth',2) ,grid on  axis([-3.1 0.1 -0.1 2.1])  legend('h[-k]') | kh1.bmp | The reflected signal. |

Step 3: The signal  is shifted from to  by changing appropriately the value of . The output if the system is computed at each shift through equation 6.1, in which only the values of overlapping parts of and  are considered. In discrete time case we will compute the sum of and. There are three stages, first there is zero overlap, next there is overlap and finally there is no overlap.

* First Stage: Zero Overlap. The stage occurs for.

|  |  |  |
| --- | --- | --- |
| Commands | Results | Comments |
| stem(kx,x,'fill','linewidth',2),grid on  axis([-5.1 3.1 -0.1 2.1])  legend('x[k]') | kx1.bmp | Input signal. |
| n=-2;  stem(-kh+n,h,'fill','linewidth',2), grid on  axis([-5.2 3.1 -0.1 2.1])  legend('h[-2-k]') | h[-2-k].bmp | Impulse response signal  for. |

Obviously the two signals do not overlap. Thus the output is.

* Second Stage: Overlap. The signals and overlap for.

In order to compute the output, the two signals are plotted for The output is computed through the convolution sum by taking into account only the overlapping samples.

For 

|  |  |
| --- | --- |
| Commands | Results |
| stem(kx,x,'fill','linewidth',2),grid on  axis([-5.1 3.1 -0.1 2.1])  legend('x[k]') | kx1.bmp |
| n=0;  stem(-kh+n,h,'fill','linewidth',2),grid on  axis([-5.2 3.1 -0.1 2.1])  legend('h[n-k]=h[0-k]') | h[0-k].bmp |

For the signals and have one overlapping sample at  The output of the system for; that is,, where 1 is the value of and 2 is the value of  at the overlapping sample. In order to understand better the output calculation, notice that according to the definition of the convolution sum, the overlapping samples of and are multiplied. The sum is applied when more than one samples overlap. This is the case for .

|  |  |
| --- | --- |
| Commands | Results |
| stem(kx,x,'fill','linewidth',2),grid on  axis([-5.1 3.1 -0.1 2.1])  legend('x[k]') | kx1.bmp |
| n=1;  stem(-kh+n,h,'fill','linewidth',2),grid on  axis([-5.2 3.1 -0.1 2.1])  legend('h[n-k]=h[0-k]') | h[1-k].bmp |

For, the signals and overlap with two samples, namely, at  and at. The output for, that is, is computed as, where 2 is the value of and at, while 1 is the value of and at. Hence.

For, we have

|  |  |
| --- | --- |
| Commands | Results |
| stem(kx,x,'fill','linewidth',2),grid on  axis([-5.1 3.1 -0.1 2.1])  legend('x[k]') | kx1.bmp |
| n=2;  stem(-kh+n,h,'fill','linewidth',2),grid on  axis([-5.2 3.1 -0.1 2.1])  legend('h[n-k]=h[2-k]') | h[2-k].bmp |

For, the signals and overlap at two points, at  and at . The output for, that is,  is computed as , where 2 is the value of and 1 is the value of at , while 1 is the value of and at . Hence 

For, we have

|  |  |
| --- | --- |
| Commands | Results |
| stem(kx,x,'fill','linewidth',2),grid on  axis([-5.1 3.1 -0.1 2.1])  legend('x[k]') | kx1.bmp |
| n=3;  stem(-kh+n,h,'fill','linewidth',2),grid on  axis([-5.2 3.1 -0.1 2.1])  legend('h[n-k]=h[3-k]') | h[3-k].bmp |

The output for, i.e., is computed as 

For,

|  |  |
| --- | --- |
| Commands | Results |
| stem(kx,x,'fill','linewidth',2),grid on  axis([-2.1 6.1 -0.1 2.1])  legend('x[k]') | xk1.bmp |
| n=4;  stem(-kh+n,h,'fill','linewidth',2),grid on  axis([-2.1 6.1 -0.1 2.1])  legend('h[n-k]=h[4-k]') | h[4-k].bmp |

For, there is only one overlap at. Thus the output for is 

* Third Stage: Zero Overlap. For, the input and impulse response signals do not overlap, hence.

For, we have

|  |  |
| --- | --- |
| Commands | Results |
| stem(kx,x,'fill','linewidth',2),grid on  axis([-2.1 6.1 -0.1 2.1])  legend('x[k]') | xk1.bmp |
| n=5;  stem(-kh+n,h,'fill','linewidth',2),grid on  axis([-2.1 6.1 -0.1 2.1])  legend('h[n-k]=h[5-k]') | h[5-k].bmp |

Combining the derived results we conclude that the response of the system with impulse response to the input signal  is. The output signal is plotted in figure below.

|  |  |
| --- | --- |
| Commands | Results |
| n=0:4;  y=[2 5 3 3 2];  stem(n,y,'fill','linewidth',2),grid on  axis([-0.1 4.1 -0.1 5.1])  legend('y[n]') | y1.bmp |

There are two basic principles that should be considered while computing the convolution between two discrete time signals  and using conv command.

1. Suppose that the length of vector is and the length of vector of is . The outcome of command y=conv(x,h) is a vector of length . In other words, length()=length()+length()-1.
2. If non zeros values of  are in the interval and non zero values of  are in the interval then the non zero values of the output are in the interval .

**In-Lab Tasks**

**Task 01: Compute and plot the convolution by any of the two procedures, where**

** and **

|  |
| --- |
| n = -5:5;  u = (n<=1);  x = 0.3.^-n.\*u;  subplot(3,1,1)  stem(n,x,'LineWidth',2)  axis([-5.5 5.5 -0.5 4])  legend('x[n]','Location','NorthEastOutside')    h=zeros(1,11);  h(7:11)=1;  subplot(3,1,2)  stem(n,h,'LineWidth',2)  axis([-5.5 5.5 -0.5 1.5])  legend('h[n]','Location','NorthEastOutside')    y =conv(h,x);  subplot(3,1,3)  stem(-5:15,y,'LineWidth',2)  legend('y[n]=x[n]\*h[n]','Location','NorthEastOutside')  **Graphical user interface, diagram  Description automatically generated** |

**Task 02: Compute and plot the convolution of following signals (by both procedures)**



and



|  |
| --- |
| n = -5:10;  x = [0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0];    subplot(3,1,1)  stem(n,x,'LineWidth',2)  legend('x[n]')    subplot(3,1,2)  a1 = (n>=0)&(n<=6);  a2 = 1.5.^n;  h = a1.\*a2;  stem(n,h,'LineWidth',2)  legend('h[n]')    y = conv(x,h);  subplot(3,1,3)  stem(-1:29,y,'LineWidth',2)  legend('y[n]=x[n]\*h[n]')  Graphical user interface  Description automatically generated  **Other Method:**  n = -5:10;  x = [0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0];  a1 = 1.5.^n;  a2 = (n>=0)&(n<=6);  h = a1.\*a2;    subplot(3,1,1)  stem(n,x,'LineWidth',2)  legend('x[n]')    subplot(3,1,2)  stem(n,h,'LineWidth',2)  legend('h[n]')    subplot(3,1,3)  stem(-n,h,'LineWidth',2)  legend('h[-n]')  Graphical user interface  Description automatically generated with medium confidence  k = -5:10; % changing axis  n = 0; % delay  subplot(12,2,1)  stem(k,x,'LineWidth',2)  legend('x[k]')  subplot(12,2,2)  stem(-k+n,h,'LineWidth',2)  legend('h[0-k]')  n=2; % delay  subplot(12,2,2)  stem(-k+n,h,'LineWidth',2)  legend('h[2-k]')    n=3; % delay  subplot(12,2,3)  stem(-k+n,h,'LineWidth',2)  legend('h[3-k]')    n=4; % delay  subplot(12,2,4)  stem(-k+n,h,'LineWidth',2)  legend('h[4-k]')    n=5; % delay  subplot(12,2,5)  stem(-k+n,h,'LineWidth',2)  legend('h[5-k]')    n=6; % delay  subplot(12,2,6)  stem(-k+n,h,'LineWidth',2)  legend('h[6-k]')    n=7; % delay  subplot(12,2,7)  stem(-k+n,h,'LineWidth',2)  legend('h[7-k]')    n=8; % delay  subplot(12,2,8)  stem(-k+n,h,'LineWidth',2)  legend('h[8-k]')    n=9; % delay  subplot(12,2,9)  stem(-k+n,h,'LineWidth',2)  legend('h[9-k]')    n=10; % delay  subplot(12,2,10)  stem(-k+n,h,'LineWidth',2)  legend('h[10-k]')    n=11; % delay  subplot(12,2,11)  stem(-k+n,h,'LineWidth',2)  legend('h[11-k]')  Diagram  Description automatically generated  Convolution  %convolution  y = [1 2.5 4.75 8.13 13.19 19.78 29.67 27.42 24.05 18.98 11.39];  yn = 0:10;  stem(yn,y,'LineWidth',2)  title('y[n]=x[n]\*h[n]')  Chart  Description automatically generated |

**Task 03: Consider and LTI system with input  and unit impulse response  specified as**

****



**Compute the response of the system (by both methods) where we have .**

|  |
| --- |
| n = -6:6;  h = [0 0 0 0 0 0 1 1 1 1 1 1 1];  x = 2.^n.\*(n<=0);  subplot(3,1,1)  stem(n,x,'LineWidth',2)  axis([-6.2 6.2 -0.2 1.2])  xlabel('n')  ylabel('x[n]')    subplot(3,1,2)  stem(n,h,'LineWidth',2)  axis([-6.2 6.2 -0.2 1.2])  xlabel('n')  ylabel('h[n]')    y = conv(h,x);  yn = -12:12;  subplot(3,1,3)  stem(yn,y,'LineWidth',2)  xlabel('n')  ylabel('y[n]')  legend('y[n]=x[n]\*h[n]')  Graphical user interface, diagram  Description automatically generated  Other Method:  k = -6:6; % changing axis  x = 2.^k.\*(k<=0);  subplot(3,1,1)  stem(k,x,'LineWidth',2)  legend('x[k]')    h = [0 0 0 0 0 0 1 1 1 1 1 1 1];  subplot(3,1,2)  stem(k,h,'LineWidth',2)  legend('h[k]')    subplot(3,1,3)  stem(-k,h,'LineWidth',2)  legend('h[-k]')  shift=0; % No shift  subplot(6,2,1)  stem(-k+shift,h,'LineWidth',2)  title('h[0-k]')  shift=1; % 1 unit shift  subplot(6,2,2)  stem(-k+shift,h,'LineWidth',2)  title('h[1-k]')    shift=2; % 2 unit shift  subplot(6,2,3)  stem(-k+shift,h,'LineWidth',2)  title('h[2-k]')  shift=3; % 3 unit shift  subplot(6,2,4)  stem(-k+shift,h,'LineWidth',2)  title('h[3-k]')  shift=4; % 4 unit shift  subplot(6,2,5)  stem(-k+shift,h,'LineWidth',2)  title('h[4-k]')    shift=5; % 5 unit shift  subplot(6,2,6)  stem(-k+shift,h,'LineWidth',2)  title('h[5-k]')  shift=6; % 6 unit shift  subplot(6,2,7)  stem(-k+shift,h,'LineWidth',2)  title('h[6-k]')  shift=7; % 7 unit shift  subplot(6,2,8)  stem(-k+shift,h,'LineWidth',2)  title('h[7-k]')  shift=8; % 8 unit shift  subplot(6,2,9)  stem(-k+shift,h,'LineWidth',2)  title('h[8-k]')  shift=9; % 9 unit shift  subplot(6,2,10)  stem(-k+shift,h,'LineWidth',2)  title('h[9-k]')  shift=10; % 10 unit shift  subplot(6,2,11)  stem(-k+shift,h,'LineWidth',2)  title('h[10-k]')    shift=11; % 11 unit shift  subplot(6,2,12)  stem(-k+shift,h,'LineWidth',2)  title('h[11-k]')  Diagram, bar chart  Description automatically generated  %Convolution  y=[ 0.0156 0.05 0.11 0.23 0.48 0.98 1.98 1.97 1.94 1.87 1.75 1.5 1.0];  yn=-6:6;  stem(yn,y,'LineWidth',2)  title('Convolution')  legend('y[n]=x[n]\*h[n]')  grid on  Chart  Description automatically generated |

**Task 04: Compute (by both procedures) and graph the convolution, where  and .**

|  |
| --- |
| n = 0:6;  h = [1 0 -1 0 0 0 0];  x = [1 1 1 1 0 0 0];  subplot(3,1,1)  stem(n,x,'LineWidth',2)  legend('x[n]')    subplot(3,1,2)  stem(n,h,'LineWidth',2)  legend('h[n]')    y = conv(x,h);  subplot(3,1,3)  stem(0:12,y,'LineWidth',2)  legend('y[n]=x[n]\*y[n]')  Diagram  Description automatically generated  **Other Method**  k = 0:6; %changing axis  x = [1 1 1 1 0 0 0];  h = [1 0 -1 0 0 0 0];  subplot(3,1,1)  stem(k,x,'LineWidth',2)  legend('x[k]')  axis([-4 4 -1.2 1.2])    subplot(3,1,2)  stem(k,h,'LineWidth',2)  legend('h[k]')  axis([-4 4 -1.2 1.2])    subplot(3,1,3)  stem(-k,h,'LineWidth',2)  legend('h[-k]')  axis([-4 4 -1.2 1.2])  Diagram  Description automatically generated with low confidence  n=0; %delay  subplot(3,2,1)  stem(-k+n,h,'LineWidth',2)  legend('h[0-k]')  xlim([-3 6])    n=1; %delay  subplot(3,2,2)  stem(-k+n,h,'LineWidth',2)  legend('h[1-k]')  xlim([-3 6])    n=2; %delay  subplot(3,2,3)  stem(-k+n,h,'LineWidth',2)  legend('h[2-k]')  xlim([-3 6])    n=3; %delay  subplot(3,2,4)  stem(-k+n,h,'LineWidth',2)  legend('h[3-k]')  xlim([-3 6])    n=4; %delay  subplot(3,2,5)  stem(-k+n,h,'LineWidth',2)  legend('h[4-k]')  xlim([-3 6])    n=5; %delay  subplot(3,2,6)  stem(-k+n,h,'LineWidth',2)  legend('h[5-k]')  xlim([-3 6])  Graphical user interface, chart  Description automatically generated  %Convolution  y = [1 1 0 0 -1 -1 0];  yn = 0:6;  stem(yn,y,'LineWidth',2)  title('Convolution')  legend('y[n]=x[n]\*h[n]')  Chart  Description automatically generated |

**Task 05: Compute (by both procedures) and graph the convolution, where**

1. 
2. 

|  |
| --- |
| **A)**  n = 0:6;  u = ones(size(n));    subplot(2,1,1)  stem(n,u,'linewidth',2)  title('u[n] = x[n] = h[n]')  xlabel('n')  ylabel('u[n]')    y = conv(u,u);  subplot(2,1,2)  stem(0:12,y,'LineWidth',2)  title('Convolution y[n] = u[n]\*u[n] = x[n]\*h[n]')  xlabel('n')  ylabel('y[n]')  Graphical user interface  Description automatically generated  **Other Method**  k = 0:6;  u = ones(size(k));  subplot(2,1,1)  stem(k,u,'LineWidth',2)  legend('u[k]')  xlim([-7 7]);    subplot(2,1,2)  stem(-k,u,'LineWidth',2)  legend('u[-k]')  xlim([-7 7]);  Chart  Description automatically generated  %Convolution  y = [1 2 3 4 5 6 7 6 5 4 3 2 1];  yn = 0:12;  stem(yn,y,'LineWidth',2)  legend('y[n]')  title('Convolution y[n] = u[n] \* u[n]')  xlabel('n')  ylabel('y[n]')  Chart  Description automatically generated  **B)**  n = 0:5;  h = (3./4).^n.\*(n>=0);  x = [0 0 0 0 3 0];  subplot(3,1,1)  stem(n,x,'LineWidth',2)  legend('x[n]')  axis([-0.2 5.2 -0.2 3.2])    subplot(3,1,2)  stem(n,h,'LineWidth',2)  legend('h[n]')  axis([-0.2 5.2 -0.2 1.2])    y = conv(x,h);  subplot(3,1,3)  stem(0:10,y,'LineWidth',2)  title('Convolution')  ylabel('y[n]')  xlabel('n')  **Timeline  Description automatically generated**  **Other Method:**  k = 0:5;  x = [0 0 0 0 3 0];  h = (3./4).^k.\*(k>=0);  subplot(3,1,1)  stem(k,x,'LineWidth',2)  legend('x[k]')  xlim([-6 6])    subplot(3,1,2)  stem(k,h,'LineWidth',2)  legend('h[k]')  xlim([-6 6])    subplot(3,1,3)  stem(-k,h,'LineWidth',2)  legend('h[-k]')  xlim([-6 6])  **Chart, box and whisker chart  Description automatically generated**  n=0; %delay  subplot(6,2,1)  stem(-k+n,h,'LineWidth',2)  title('h[0-k]')  xlim([-6 12])    n=1; %delay  subplot(6,2,2)  stem(-k+n,h,'LineWidth',2)  title('h[1-k]')  xlim([-6 12])    n=2; %delay  subplot(6,2,3)  stem(-k+n,h,'LineWidth',2)  title('h[2-k]')  xlim([-6 12])    n=3; %delay  subplot(6,2,4)  stem(-k+n,h,'LineWidth',2)  title('h[3-k]')  xlim([-6 12])    n=4; %delay  subplot(6,2,4)  stem(-k+n,h,'LineWidth',2)  title('h[4-k]')  xlim([-6 12])    n=5; %delay  subplot(6,2,5)  stem(-k+n,h,'LineWidth',2)  title('h[5-k]')  xlim([-6 12])    n=6; %delay  subplot(6,2,6)  stem(-k+n,h,'LineWidth',2)  title('h[6-k]')  xlim([-6 12])    n=7; %delay  subplot(6,2,7)  stem(-k+n,h,'LineWidth',2)  title('h[7-k]')  xlim([-6 12])    n=8; %delay  subplot(6,2,8)  stem(-k+n,h,'LineWidth',2)  title('h[8-k]')  xlim([-6 12])    n=9; %delay  subplot(6,2,9)  stem(-k+n,h,'LineWidth',2)  title('h[9-k]')  xlim([-6 12])    n=10; %delay  subplot(6,2,10)  stem(-k+n,h,'LineWidth',2)  title('h[10-k]')  xlim([-6 12])    n=11; %delay  subplot(6,2,11)  stem(-k+n,h,'LineWidth',2)  title('h[11-k]')  xlim([-6 12])  **Chart, bar chart  Description automatically generated**  %Convolution  y = [0 0 0 0 3 2.2 1.8 1.2 1 0.8 0];  yn = 0:10;  stem(yn,y,'LineWidth',2),grid on  legend('y[n]')  title('Convolution')  **Chart  Description automatically generated** |

**Post-Lab Tasks**

## Critical Analysis / Conclusion

|  |
| --- |
| In this lab we learnt how to perform convolution in Matlab in two different ways. Firstly, we used matlab built-in function conv() which takes two argument i.e. the input signal and Impulse response of the signal .  Secondly, we use the conventional method to convolve the two signals in which we flip one of the signals in first step then we shift the flipped signal until there is an overlap between the input and impulse response. |

|  |  |  |
| --- | --- | --- |
| **Lab Assessment** | | |
| **Pre-Lab** | **/1** | **/10** |
| **In-Lab** | **/5** |
| **Critical Analysis** | **/4** |
| **Instructor Signature and Comments** | | |