Roll No: 5117060

EXPERIMENT 03

Aim: To implement Discrete Convolution.

Theory:

Convolution, one of the most important concepts in electrical engineering, can be used to determine the output a system produces for a given input signal. It can be shown that a linear time invariant system is completely characterized by its impulse response. The sifting property of the discrete time impulse function tells us that the input signal to a system can be represented as a sum of scaled and shifted unit impulses. Thus, by linearity, it would seem reasonable to compute of the output signal as the sum of scaled and shifted unit impulse responses. That is exactly what the operation of convolution accomplishes. Hence, convolution can be used to determine a linear time invariant system's output from knowledge of the input and the impulse response.

In mathematics (in particular, functional analysis), **convolution** is a mathematical operation on

two functions (*f* and *g*) that produces a third function ( ) that expresses how the shape of one is modified by the other. The term *convolution* refers to both the result function and to the process of computing it. It is defined as the integral of the product of the two functions after one is reversed and shifted. And the integral is evaluated for all values of shift, producing the convolution function.

Some features of convolution are similar to cross-correlation: for real-valued functions, of a continuous or discrete variable, it differs from cross-correlation ( ) only in that either *f*(*x*) or *g*(*x*) is reflected about the y-axis; thus it is a cross-correlation of *f*(*x*) and *g*(−*x*), or *f*(−*x*) and *g*(*x*). For complex-valued functions, the cross-correlation operator is the adjoint of the convolution operator.

Convolution has applications that include probability, statistics, computer vision, natural language processing, image and signal processing, engineering, and differential equations.

The convolution can be defined for functions on Euclidean space and other groups. For example, periodic functions, such as the discrete-time Fourier transform, can be defined on a circle and convolved by *periodic convolution*. (See row 18 at DTFT § Properties.) A *discrete convolution* can be defined for functions on the set of integers.

Generalizations of convolution have applications in the field of numerical analysis and numerical linear algebra, and in the design and implementation of finite impulse response filters in signal processing.

Computing the inverse of the convolution operation is known as deconvolution. **Convolution and Circular Convolution**

Convolution

Operation Definition

Discrete time convolution is an operation on two discrete time signals defined by the integral
(f*g)[n]=∑k=−∞∞f[k]g[n−k](f*g)[n]=∑k=-∞∞f[k]g[n-k]

for all signals f,gf,g defined on ZZ. It is important to note that the operation of convolution is commutative, meaning that

f*g=g*ff*g=g*f

for all signals f,gf,g defined on ZZ. Thus, the convolution operation could have been just as easily stated using the equivalent definition

(f*g)[n]=∑k=−∞∞f[n−k]g[k](f*g)[n]=∑k=-∞∞f[n-k]g[k]

for all signals f,gf,g defined on ZZ. Convolution has several other important properties not listed here but explained and derived in a later module.

Circular Convolution

Discrete time circular convolution is an operation on two finite length or periodic discrete time signals defined by the sum

(f⊛g)[n]=∑k=0N−1f^[k]g^[n−k](f ⊛g)[n]=∑k=0N-1f^[k]g^[n-k]

for all signals f,gf,g defined on Z[0,N−1]Z[0,N-1] where f^,g^f^,g^ are periodic extensions of ff and gg. It is important to note that the operation of circular convolution is commutative, meaning that

f⊛g=g⊛ff⊛g=g⊛f

for all signals f,gf,g defined on Z[0,N−1]Z[0,N-1]. Thus, the circular convolution operation could have been just as easily stated using the equivalent definition

(f⊛g)[n]=∑k=0N−1f^[n−k]g^[k](f⊛g)[n]=∑k=0N-1f^[n-k]g^[k]

for all signals f,gf,g defined on Z[0,N−1]Z[0,N-1] where f^,g^f^,g^ are periodic extensions of ff and gg. Circular convolution has several other important properties not listed here but explained and derived in a later module.

Alternatively, discrete time circular convolution can be expressed as the sum of two summations given by

(f⊛g)[n]=∑k=0nf[k]g[n−k]+∑k=n+1N−1f[k]g[n−k+N](f⊛g)[n]=∑k=0nf[k]g[n-k]+∑k=n+1N 1f[k]g[n-k+N]

for all signals f,gf,g defined on Z[0,N−1]Z[0,N-1].
Meaningful examples of computing discrete time circular convolutions in the time domain would involve complicated algebraic manipulations dealing with the wrap around behavior, which would ultimately be more confusing than helpful. Thus, none will be provided in this section. Of course, example computations in the time domain are easy to program and demonstrate. However, discrete time circular convolutions are more easily computed using frequency domain tools as will be shown in the discrete time Fourier series section.

Conclusion:

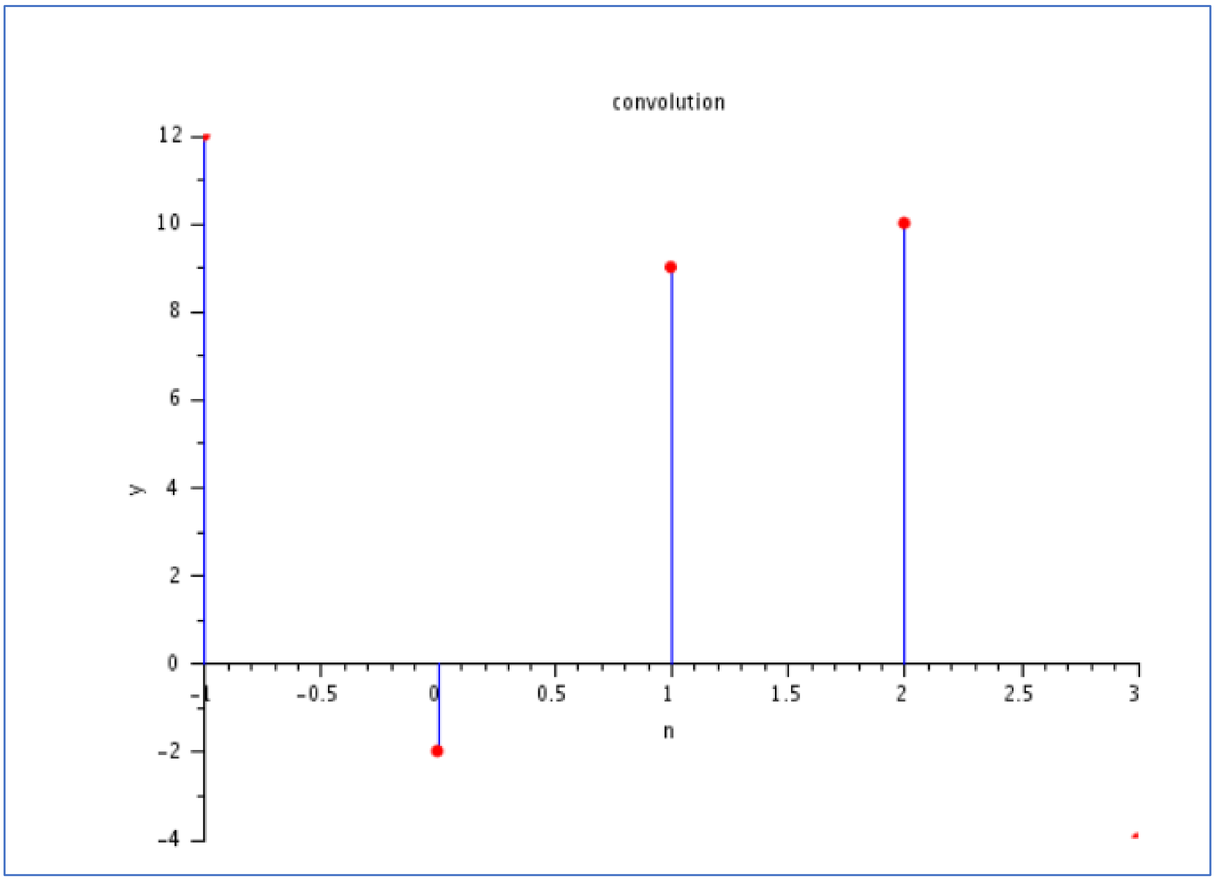
Thus we have successfully implemented Convolution.

Code:

```
clear;
clc;
g=[3 -2 4];//originating at n=-1
h=[4 2 -1];//originating at n=0
q=length(g);
w=length(h);
z=q+w-1;
y0=0;
for i=1:z;
    y(i)=0;
    for k=1:i;
        if k>q
            g(k)=0;
        else
            if (i-k+1)>w
                h(i-k+1)=0;
            else
             y(i)= y(i) + g(k)*h(i-k+1);
            end
        end
    end
end
n=-1:z-2;
disp(y,'The Convolution of the two sequences is =')
clf();
a=gca();
figure(0);
a.x_location="origin";
plot2d3(n,y,2);
plot(n,y,'r.');
xtitle('convolution','n','y');
```

Output:

```
The Convolution of the two sequences is =
    12.
  - 2.
    9.
    10.
  - 4.
```



Code:

```
//EXAMPLE 2.26,convolution of x & h
x=[-2 0 1 -1 3];
disp(x,'x = ');
h=[1 2 0 -1];
disp(h,'h = ');
n=0:7;
y=convol(x,h);
disp(y,'The convolution of the two inputs is :')
```

Output:

```
x =    - 2.    0.    1.  - 1.    3.
 h =    1.    2.    0.  - 1.
 The convolution of the two inputs is :
  - 2.  - 4.    1.    3.    1.    5.    1.  - 3.
```