

Experiment # 6

Convolution

Introduction to the convolution

Amongst the concepts that cause the most confusion to electrical engineering students, the Convolution Integral stands as a repeat offender. As such, the point of this experiment is to explain what a convolution integral is, why engineers need it, and the math behind it.

In essence, the “convolution” of two functions (over the same variable, e.g. $f_1(t)$ and $f_2(t)$) is an operation that produces a separate third function that describes how the first function “modifies” the second one. Conversely, the resulting function can be seen as how the second function “modifies” the first function. Sometimes the result is used to describe how much the first two functions “have in common.” In all honesty, the concept of the convolution of two functions is quite abstract, but the frequency at which it appears in nature grants its importance to scientists and engineers. Ultimately the aim here is to identify its use to electrical engineers – so for now do not dwell solely on its mathematical significance.

A convolution of two functions is denoted with the operator “*“, and is written as:

$$f_1(t) * f_2(t) = \int_{-\infty}^{\infty} f_1(\tau) * f_2(t - \tau) d\tau$$

Where τ is used as a “dummy variable.”

Before diving any further into the math, let us first discuss the relevance of this equation to the realm of electrical engineering.

Useful Properties

The following is a list of useful properties of the convolution integral that can help in developing an intuitive approach to solving problems:

1.) Commutative Property:

$$f_1(t) * f_2(t) = f_2(t) * f_1(t)$$

2.) Distributive Property:

$$f_1(t) * (f_2(t) + f_3(t)) = f_1(t) * f_2(t) + f_1(t) * f_3(t)$$

3.) Associative Property:

$$f_1(t) * (f_2(t) * f_3(t)) = (f_1(t) * f_2(t)) * f_3(t)$$

4.) Convolution with an Impulse results in the original function:

$$f(t) * \delta(t) = f(t)$$

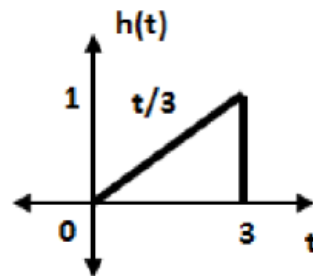
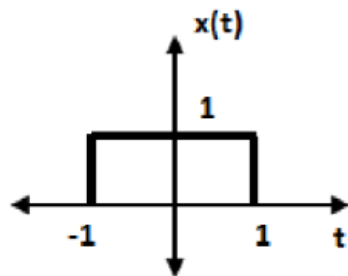
5.) Width Property:

The convolution of a signal of duration T_1 and a signal of duration T_2 will result in a signal of duration $T_3 = T_1 + T_2$

- **Continuous Convolution:**

Example:

In this example, use the function `conv` to compute the convolution of the continuous signals $(x(t), h(t))$ with interval of time between $-1 \leq t \leq 4$, where the both signal are shown below:



$$x(t) = u(t + 1) - u(t - 1).$$

$$h(t) = \left(\frac{1}{3}\right) t [u(t) - u(t - 3)].$$

```

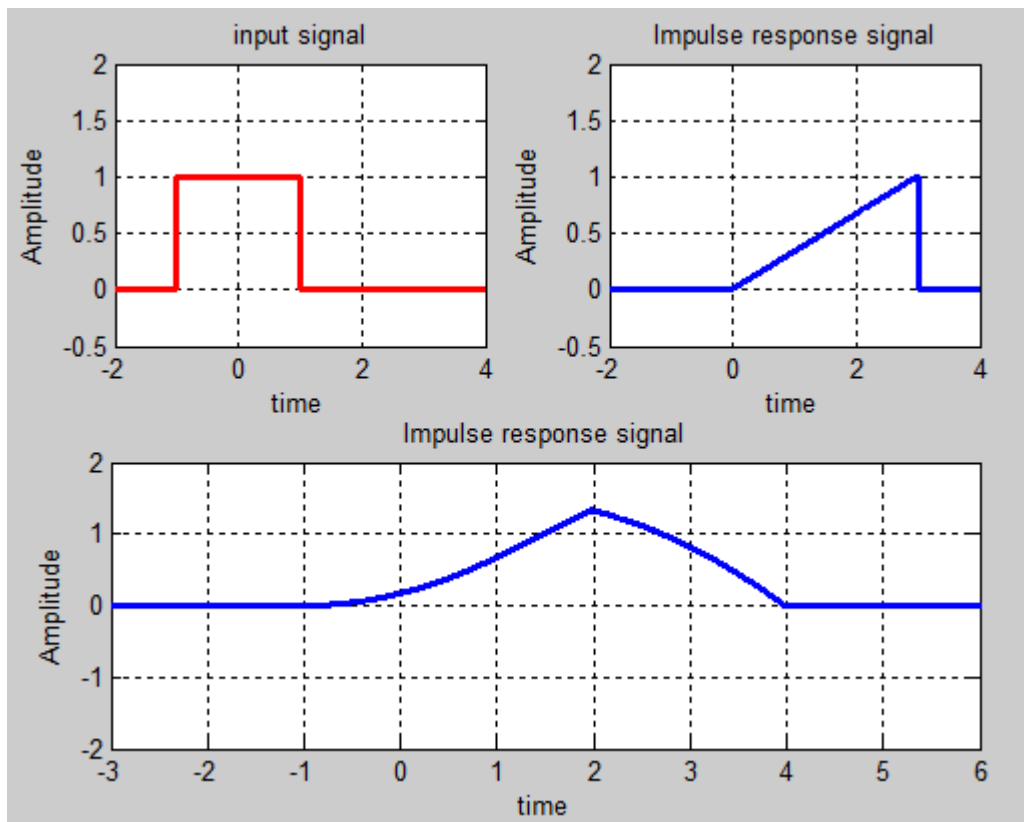
t=-2:0.001:4;
T=-4:0.001:8;
x=hardlim(t+1)-hardlim(t-1);
% x=((t>=-1)&(t<1));
h=(1/3)*(t).*(hardlim(t)-
hardlim(t-3));
y=(conv(x,h)).* 0.001;
subplot(2,2,1)
plot(t,x,'r','linewidth',2.5)
xlabel('time');
ylabel('Amplitude');
title('input signal');
grid on
axis([-2 4 -0.5 2])
subplot(2,2,2)
plot(t,h,'b','linewidth',2.5)
xlabel('time');
ylabel('Amplitude');
title('Impulse response
signal');
grid on
axis([-2 4 -0.5 2])
subplot(2,1,2)

```

```

plot(T,y,'b','linewidth',2.5)
xlabel('time');
ylabel('Amplitude');
title('Impulse response
signal');
grid on
axis([-3 6 -2 2])

```



- **Discrete convolution**

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

If arbitrary sequences are of infinite duration, then MATLAB cannot be used directly to compute the convolution. MATLAB does provide a built-in function called **conv** that computes the convolution between two finite-duration sequences. The conv function assumes that the two sequences begin at $n = 0$ and is invoked by :

» `y = conv(x,h);`

For example, given the following two sequences:

$$x(n) = [3, 11, 7, 0, -1, 4, 2], \quad -3 \leq n \leq 3; \quad h(n) = [2, 3, 0, -5, 2, 1], \quad -1 \leq n \leq 4$$

↑
↑

determine the convolution $y(n) = x(n) * h(n)$.

```
» x = [3, 11, 7, 0, -1, 4, 2].
» h = [2, 3, 0, -5, 2, 1];
» y = conv(x,h)
```

`y =`
`6 31 47 6 -51 -5 41 18 -22 -3 8 2`

However, the `conv` function neither provides nor accepts any timing information if the sequences have arbitrary support. What is needed is a beginning point and an end point of $y(n)$. Given finite duration $x(n)$ and $h(n)$, it is easy to determine these points. Let

$$\{x(n); n_{xb} \leq n \leq n_{xe}\} \text{ and } \{h(n); n_{hb} \leq n \leq n_{he}\} \Rightarrow$$

Then the beginning and end points of $y(n)$ are

$$\Rightarrow \text{for } y(n) \rightarrow n_{yb} = n_{xb} + n_{hb} \text{ and } n_{ye} = n_{xe} + n_{he}$$

A simple extension of the `conv` function, called `conv_m`, which performs the convolution of arbitrary support sequences can be designed.

```
function [y,ny] = conv_m(x,nx,h,nh)
nyb = nx(1)+nh(1);
nye = nx(length(x)) + nh(length(h));
ny = [nyb:nye];
y = conv(x,h);
end
```

now resolve the previous convolution using the conv_m function.

```
x=[3, 11, 7, 0, -1, 4, 2];
nx =[-3:3];
h =[2, 3, 0, -5, 2, 1];
nh = [-1:4];
[y,ny] =conv_m(x,nx,h,nh)
subplot(3,1,1)
stem(nx,x)
xlabel('n')
ylabel('x[n]')
subplot(3,1,2)
stem(nh,h)
xlabel('n')
ylabel('h[n]')
subplot(3,1,3)
stem(ny,y)
xlabel('n')
ylabel('y[n]')
```

y = 6 31 47 6 -51 -5 41 18 -22 -3 8 2

ny= -4 -3 -2 -1 0 1 2 3 4 5 6 7

Hence

$y(n) = \{6, 31, 47, 6, -51, -5, 41, 18, -22, -3, 8, 2\}$

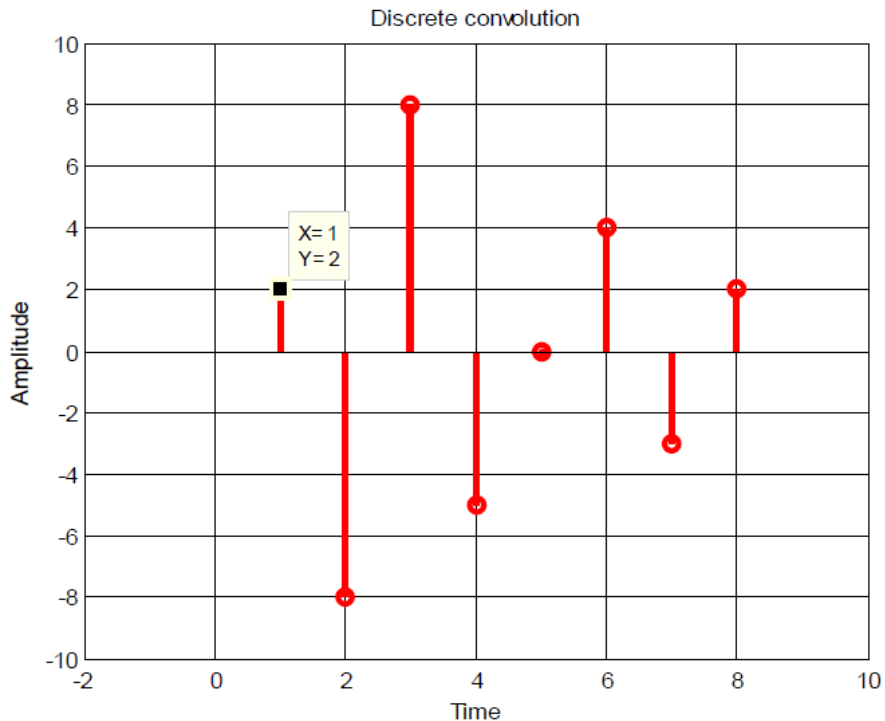


Exercise:

$x[n] = \{-1, 3, -1, -2\}$ and $h[n] = \{-2, 2, 0, -1, 1\}$ to find $y[n]$

Without using function

```
x=[-1 3 -1 2];
h=[-2 2 0 -1 1];
y=conv(x,h);
stem(y,'r','linewidth',2.5)
axis([-2 10 -10 10])
grid on
xlabel('Time')
ylabel('Amplitude')
title('Discrete convolution')
```



Using function

function

```
[y,yn]=convolution(x,xn,h,hn)
y=conv(x,h);
y ns=xn(1)+hn(1);
y ne=xn(length(x))+hn(length(h));
yn=[y ns:y ne];
```

after modified function

```
x=[-1 3 -1 2];
xn=[-1:2];
h=[-2 2 0 -1 1];
hn=[-1:3];
[y,yn]=convolution(x,xn,h,hn)
stem(yn,y,'r','linewidth',2.5)
axis([-4 10 -10 10])
grid on
xlabel('Time')
ylabel('Amplitude')
title('Discrete convolution')
```

