San Francisco State University

Engineering 315

Laboratory #4 - Convolution

1. Purpose

In this lab, you will investigate the properties of convolution, and use convolution to understand what happens when systems process signals.

Background reading includes:

- Lathi, Chapter2
- Holton notes, Unit 3

2. Background

Convolution

Convolution is an operation on two functions, $x(t)$ and $h(t)$, producing an output function, $y(t)$, defined by

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \tag{L4.1}$$

We want to use Matlab to approximate the convolution of continuous-time systems, even though Matlab inherently quantizes the time axis. What this means is that instead of allowing us to manipulate a continuous function, for example $x(t)$, Matlab creates an array of points, $x[n]$, that correspond to $x(t)$ sampled at multiples of a fixed period, Δ. That is, $x[n] = x(n\Delta)$. To get Matlab to approximate a continuous-time convolution, we start by noting that a continuous time function such as $x(t)$ is defined as

$$x(t) = \int_{-\infty}^{\infty} x(\tau)\delta(t-\tau)d\tau\ . \tag{L4.2}$$

This can be approximated by

$$x_{\Delta}(t) = \sum_{k=-\infty}^{\infty} x(k\Delta)\,p_{\Delta}(t-k\Delta)\,, \tag{L4.3}$$

where $p_{\Delta}(t)$ is a pulse function of width Δ:

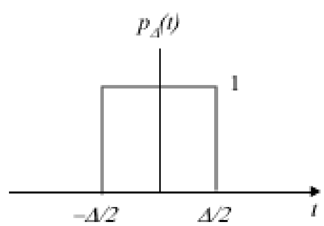

Figure 1: Pulse function

Equation (L4.3) says that *x(t)* can be approximated the sum of pulse functions, as shown in the top panel of Figure 2.

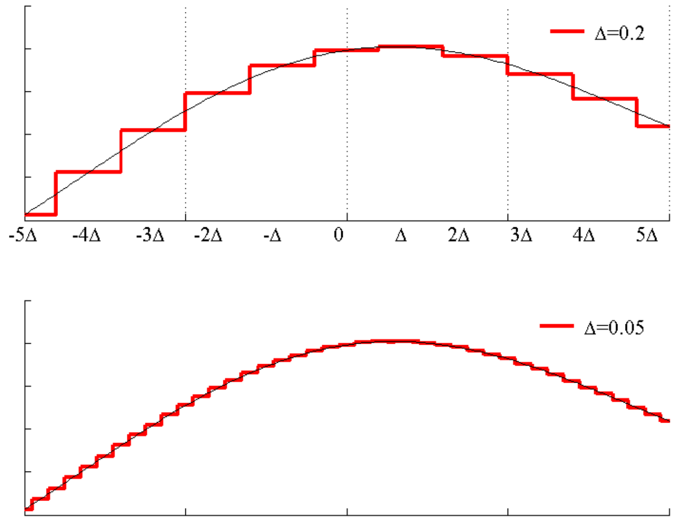

Figure 2: Approximation of continuous function

Each pulse is shifted over so that it is centered on a time point $t = n\Delta$ and then scaled by $x(n\Delta)$, producing a "staircase approximation" of $x(t)$. As Δ gets smaller, the staircase approximation, $x_\Delta(t)$, matches $x(t)$ more and more exactly, as shown in the lower panel of Figure 2; that is

$$\lim_{\Delta \to 0} x_\Delta(t) = x(t). \tag{L4.4}$$

The convolution of Equation (L4.1) can therefore be approximated by

$$y_\Delta(t) = \int_{-\infty}^{\infty} x_\Delta(\tau) h_\Delta(t-\tau) d\tau \tag{L4.5}$$

where

$$h_\Delta(t) = \sum_{l=-\infty}^{\infty} h(l\Delta) p_\Delta(t - l\Delta). \tag{L4.6}$$

and

$$y_\Delta(t) = \sum_{n=-\infty}^{\infty} y(n\Delta) p_\Delta(t - n\Delta) \tag{L4.7}$$

Substituting Equations (L4.3) and (L4.6) into Equation (L4.5) gives

$$\begin{aligned} y_\Delta(t) &= \int_{-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x(k\Delta) p_\Delta(\tau - k\Delta) \sum_{l=-\infty}^{\infty} h(l\Delta) p_\Delta(t - \tau - l\Delta) d\tau \\ &= \sum_{k=-\infty}^{\infty} x(k\Delta) \sum_{l=-\infty}^{\infty} h(l\Delta) \int_{-\infty}^{\infty} p_\Delta(\tau - k\Delta) p_\Delta(t - \tau - l\Delta) d\tau \end{aligned}. \tag{L4.8}$$

We now define

$$\Lambda(t) \triangleq \int_{-\infty}^{\infty} p_\Delta(\tau) p_\Delta(t-\tau) d\tau = p_\Delta(t) * p_\Delta(t), \tag{L4.9}$$

which looks like this

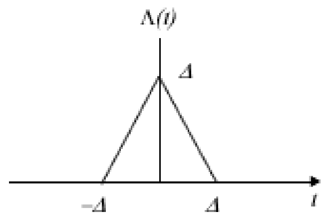

Figure 3: $\Lambda(t)$

By substituting $\mu = \tau - k\Delta$ into the integral in Equation (L4.8) , we see that it can be written

$$\begin{aligned}\int_{-\infty}^{\infty} p_\Delta(\tau - k\Delta) p_\Delta(t - \tau - l\Delta) d\tau &= \int_{-\infty}^{\infty} p_\Delta(\underbrace{\tau - k\Delta}_{\mu}) p_\Delta(t - \underbrace{(\tau - k\Delta)}_{\mu} - k\Delta - l\Delta) d\tau \\ &= \int_{-\infty}^{\infty} p_\Delta(\mu) p_\Delta(t - (l+k)\Delta - \mu) d\mu \\ &= \Lambda(t - (l+k)\Delta)\end{aligned} .$$

So, Equation (L4.8) becomes

$$y_\Delta(t) = \sum_{k=-\infty}^{\infty} x(k\Delta) \sum_{l=-\infty}^{\infty} h(l\Delta)\Lambda(t - (l+k)\Delta) . \tag{L4.10}$$

In general, $y_\Delta(t)$ is a mess to compute for arbitrary values of t . However, if we restrict t to integer values of Δ, that is $t = n\Delta$, then

$$\begin{aligned} y_\Delta(n\Delta) &= \sum_{k=-\infty}^{\infty} x(k\Delta) \sum_{l=-\infty}^{\infty} h(l\Delta)\Lambda((n - (l+k))\Delta) \\ &= \sum_{k=-\infty}^{\infty} x(k\Delta) \sum_{l=-\infty}^{\infty} h(l\Delta)\Lambda(((n-k) - l)\Delta) \end{aligned} . \tag{L4.11}$$

From Figure 3 we see that

$$\Lambda(n\Delta) = \begin{cases} \Delta, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

so $\Lambda(((n-k)-l)\Delta)$ is zero unless $l = n - k$, at which time it has value Δ . That means that only one value of the inner summation in Equation (L4.11) survives:

$$\sum_{l=-\infty}^{\infty} h(l\Delta)\Lambda(((n-k)-l)\Delta) = \Delta h((n-k)\Delta)$$

so Equation (L4.11) becomes

$$y_\Delta(n\Delta) = \Delta \sum_{k=-\infty}^{\infty} x(k\Delta) h((n-k)\Delta) . \tag{L4.12}$$

Evaluating Equation (L4.7) at $t = n\Delta$ gives $y_\Delta(n\Delta) = y(n\Delta)$, so Equation (L4.12) becomes

$$\boxed{y(n\Delta) \;\; = \Delta \sum_{k=-\infty}^{\infty} x(k\Delta) h((n-k)\Delta)} \tag{L4.13}$$

This equation says that for values of $y(t)$ that are sampled at integer multiples of Δ, that is $t = n\Delta$, the convolution integral of Equation (L4.1) can be approximated by a discrete summation at $x(t)$ and $h(t)$ sampled at multiples of Δ. This summation is called the *discrete-time convolution sum*. Put another way, we can approximate $y(t)$ at specific values of $t = n\Delta$ using the convolution sum instead of computing the continuous-time integral of Equation (L4.1). The good news here is that we can use Matlab's **conv** function to compute the convolution sum. For example, given $x(t) = u(t) - u(t-1)$ and $h(t) = u(t) - u(t-2)$, here's a little Matlab script to compute and plot the convolution of two simple functions, for $\Delta = 0.01$, and compare it to the theoretically expected result, sampled at 0.1 sec:

```
% Approximation(computed convolution)
delta = 0.01;
t = 0:delta:5;
x = u(t) - u(t-1);
h = u(t) - u(t-2);
y = delta * conv(x, h);
tt = 0:delta:10; % note: it's twice as long as 't'
plot(tt, y, 'LineWidth', 2);
xlim([0 5])
hold on;

% Theoretical convolution
delta2 = 0.1;
t = 0:delta2:5;
y2 = t.*u(t)+(1-t).*u(t-1)+(2-t).*u(t-2)+(t-3).*u(t-3);
plot(t, y2, 'ro', 'MarkerFaceColor', 'r');
axis([0 5 0 1.1]);
legend('Approximation', 'Theoretical');
```

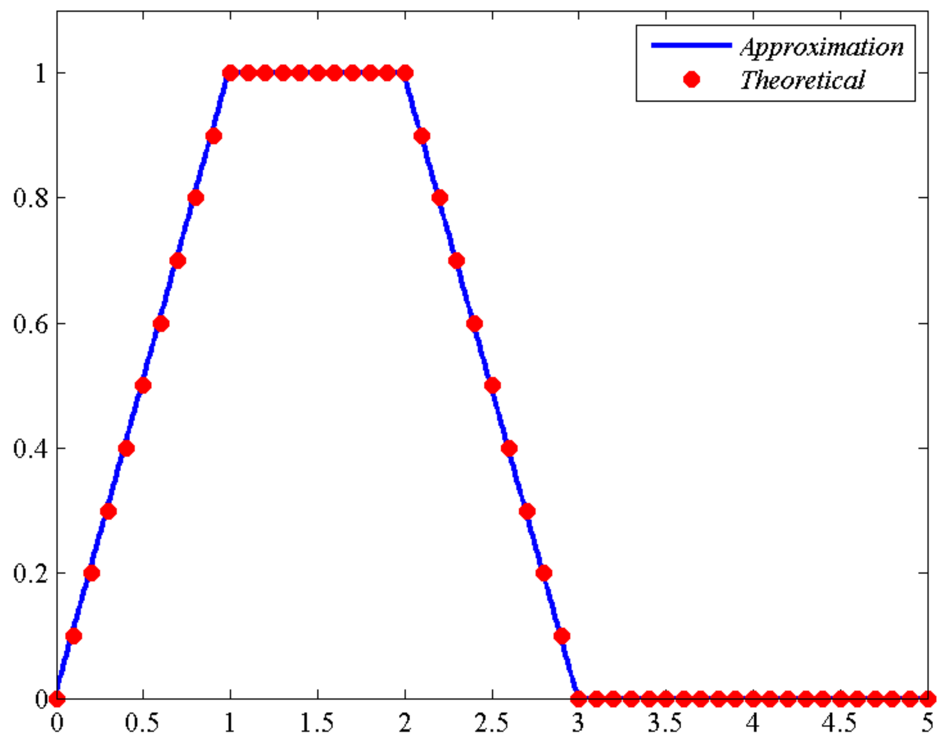

Figure 4: Example of convolution

Properties

Convolution is an operator, in the same sense that multiplication and addition are operators. It satisfies commutative, associative and distributive properties.

The commutative property
Commutivity says the order of successive convolutions doesn't matter. That is

$$x(t) * h(t) = h(t) * x(t) \tag{L4.14}$$

This is illustrated in Figure 5 in terms of two experiments. In the first experiment, we first pass input, *x(t)*, through a system characterized by impulse response $h(t)$ yielding $y(t) = x(t) * h(t)$. In the second experiment, we convolve *h(t)* with $x(t)$, so $y(t) = h(t) * x(t)$. The commutative properties says that results of the two experiments is the same.

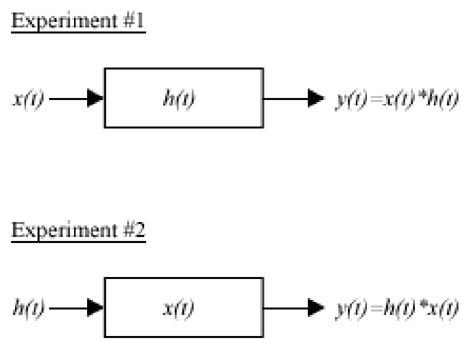

Figure 5: Commutativity of convolution

The associative property
Associativity says the grouping of successive convolutions doesn't matter. That is

$$\left(x(t) * h_1(t)\right) * h_2(t) = x(t) * \left(h_1(t) * h_2(t)\right) \tag{L4.15}$$

This is illustrated in Figure 6 in terms of two experiments. In the first experiment, we convolve $x(t)$ with $h_1(t)$ and then convolve the result with $h_2(t)$. Hence, $y(t) = \left(x(t) * h_1(t)\right) * h_2(t)$. In the second experiment, we first convolve $h_1(t)$ with $h_2(t)$ and then convolve the input, $x(t)$, with the result, yielding $y(t) = x(t) * \left(h_1(t) * h_2(t)\right)$. The associative property says that results of the two experiments is the same.

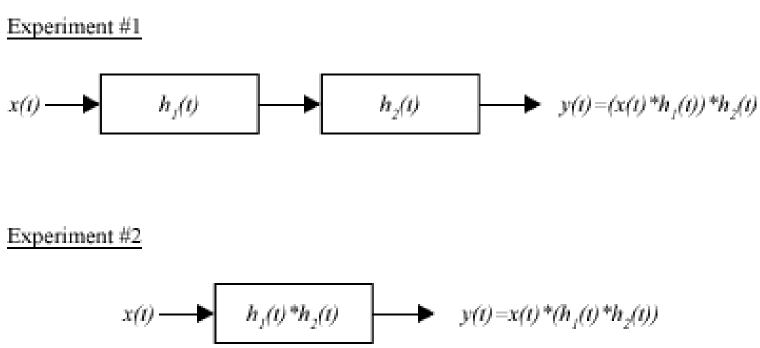

Figure 6: Associativity of convolution

The distributative property

Distributivity says that

$$x(t)*h_1(t)+x(t)*h_2(t)=x(t)*\left(h_1(t)+h_2(t)\right) \tag{L4.16}$$

This is illustrated in Figure 7 in terms of two experiments. In the first experiment, we convolve $x(t)$ with $h_1(t)$ and convolve *x(t)* with $h_2(t)$ and then add the results of these two convoltions; hence, $y(t)=x(t)*h_1(t)+x(t)*h_2(t)$. In the second experiment, we first add $h_1(t)$ and $h_2(t)$ and then convolve the input, $x(t)$, with the result, yielding $y(t)=x(t)*\left(h_1(t)+h_2(t)\right)$. The distributative property says that results of the two experiments is the same.

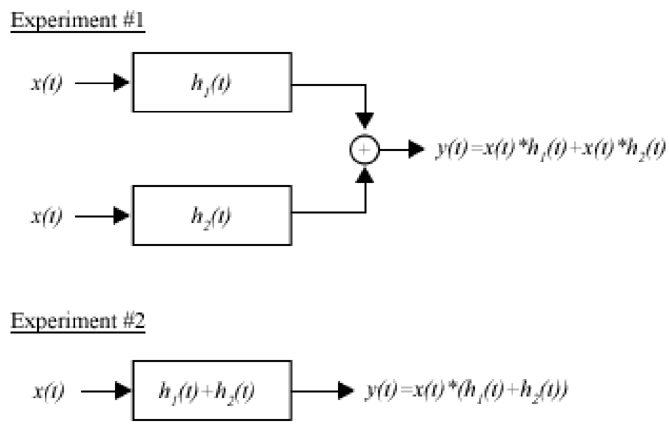

Figure 7: Distributivity of convolution

3. Assignment

I. In the first part of this assignment we compute the theoretical convolution of two functions and then use Matlab to check the results. For each of the following parts produce a plot of the convolution of the given pairs of functions such as that shown in Figure 4. Plot the Matlab approximation with a solid blue line. Plot the theoretical curve with red dots at regular intervals of time with a step size for the no larger than 0.2 sec. Your theoretical plots should not just be an array of numbers, but a formula, as shown in the example accompanying Figure 4. The time limits of the plot should be $-2\le t<6$ for all plots.

a. $\boxed{\begin{aligned}x(t)&=u(t+1)+u(t)-2u(t-1)\\h(t)&=u(t)-2u(t-1)+u(t-2)\end{aligned}}$

b. $\boxed{\begin{aligned}x(t)&=\sin(2\pi t)\left(u(t)-u(t-3)\right)\\h(t)&=u(t)-u(t-3)\end{aligned}}$

c. $\boxed{\begin{aligned}x(t)&=e^{-t}u(t)\\h(t)&=u(t-1)-u(t-3)\end{aligned}}$

II. In the second part of the assignment, we investigate the properties of convolution. Plot each part on a separate panel and use two different colors for the two cases (e.g. solid blue for $x(t)*h(t)$ and dotted red for $h(t)*x(t)$ in part a)). Again, the time limits of the plots should be $-2\le t<6$.

a. Commutative. Use Matlab to demonstrate that convolution is commutative given

$$\boxed{\begin{aligned}&x(t)=2\left(u(t)-u(t-2)\right)\\&h(t)=u(t)-u(t-1)\end{aligned}}$$

b. Associative. Show that convolution is associative given

$$\boxed{\begin{aligned}&x(t)=u(t)-u(t-1)\\&h_1(t)=u(t)-2u(t-2)+u(t-3)\\&h_2(t)=u(t+1)+u(t-1)-2(t-2)\end{aligned}}$$

c. Distributive. Show that convolution is distributative using the same $x(t)$, $h_1(t)$ and $h_2(t)$ given in part b.
