Lecture 3

Convolution

Introduction

For a continuous-time, causal, LTI system with finite-duration impulse response $h(t)$, the output $y(t)$ is related to the input $x(t)$ by the convolution

$$y(t)=h(t)*x(t)=\int_0^{t_M} h(\tau)x(t-\tau)\,d\tau \tag{1}$$

Although this accurately describes the system, it does not provide a practical method for implementing it. It may not even be possible to analytically calculate the convolution integral.

In the discrete-time case the convolution is a finite sum

$$y(n)=h(n)*x(n)=\sum_{k=0}^{M} h(k)x(n-k) \tag{2}$$

which can actually be calculated on a computer (or DSP chip) and gives us a practical formula for implementing the digital system. For discrete-time systems convolution is of both theoretical *and* practical interest. Therefore, it is worth our while to study discrete convolution in some detail.

Interpretation

The convolution

$$y(n)=h(n)*x(n) \tag{3}$$

can be expressed in two ways,

$$y(n)=\sum_{k=-\infty}^{\infty} h(k)x(n-k) \tag{4}$$

and

$$y(n)=\sum_{k=-\infty}^{\infty} x(k)h(n-k) \tag{5}$$

Consider the $n=2$ terms of (4)

$$\begin{aligned} y(2)=h(0)x(2)+h(1)x(1)+h(2)x(0)+\cdots \\ +h(-1)x(3)+h(-2)x(4)+\cdots \end{aligned} \tag{6}$$

This shows how $y(2)$ is a linear combination of all input samples (some of the coefficients can, of course, be zero) and is illustrated in Fig. 1 at left.

Consider the $k=2$ term of (5). This tells us that for any value of n, $x(2)$ contributes the term

$$y(n)\leftarrow x(2)h(n-2) \tag{7}$$

to the corresponding output. Specifically

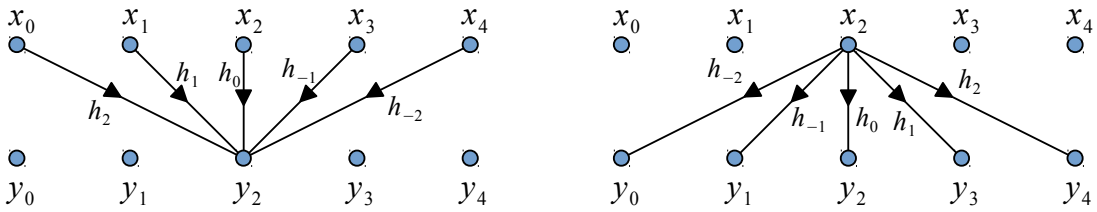

Fig. 1: (Left) each output value is a linear combination of input values. The weight of the contribution is h_k where k is the number of time steps in the future of the output relative to the input. If k is negative then the present is affected by the future (a non-causal system). (Right) each input value can contribute to every output value; x_n contributes to y_{n+k} with weight h_k.

$$\begin{aligned} y(0) &\leftarrow x(2)h(-2)\ ,\ y(1) \leftarrow x(2)h(-1) \\ y(2) &\leftarrow x(2)h(0)\ ,\ y(3) \leftarrow x(2)h(1)\ ,\ y(4) \leftarrow x(2)h(2) \end{aligned} \tag{8}$$

This is illustrated in Fig. 1 at right. Each input sample contributes (in principle) to every output sample (again, some of the coefficients can be zero). The coefficients that "connect" the input samples to the output samples are the impulse response values. We can think of the representations in Fig. 1 as the "output point of view" and the "input point of view."

Calculation

The general forms (4) and (5) have infinitely many terms. In some cases the summations may be performed analytically, but an infinite summation cannot be performed numerically. Suppose both $h(n)$ and $x(n)$ are causal ($h(n)=x(n)=0$ for all $n<0$). Then for $n \geq 0$

$$y(n)=\sum_{k=0}^{n} h(k)x(n-k) \tag{9}$$

is a finite sum which we can easily code in Scilab as

```
for n=0:nMax
  y(n+1) = 0;
  for k=0:n
    y(n+1) = y(n+1)+h(k+1)*x(n-k+1);
  end
end
```

The $+1$ terms in the indices compensate for Scilab indexing starting at 1 instead of 0. The number of terms in the k summation grow with n. If the system has a *finite impulse response* (FIR) such that $h(n)=0$ for $n<0, n>M$ then

$$y(n)=\sum_{k=0}^{M} h(k)x(n-k) \tag{10}$$

and we only have $M+1$ terms to calculate for each value of n. "FIR filters" are typically implemented using this summation directly, either in software or DSP hardware.

Substituting $k=n-i$ we have

$$y(n)=\sum_{i=n-M}^{n} x(i)h(n-i) \tag{11}$$

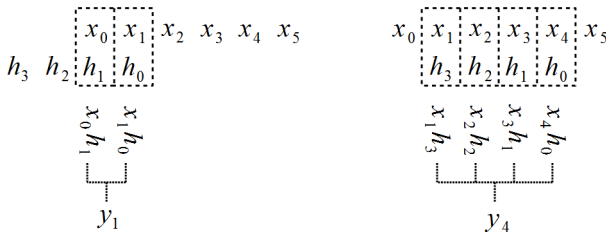

Fig. 2: "Flip and shift" method of convolution.

which we can interpret graphically as telling us to "flip and shift" the impulse response $h(n)$ along the input signal $x(n)$. This is illustrated in Fig. 2.

Example 1: What is the convolution of $x=[\mathring{0},1,3,2,1,0]$ and $h=[\mathring{1},-1,1,-1]$?

Let's list $x(k)$ and $h(n-k)$ for $n=0,1,2,\ldots$, multiply corresponding terms, and sum (the "flip and shift" method). Here we underline the $n=0$ samples.

```
x(k)              0  1  3  2  1  0
h(0-k) -1  1 -1   1
                  0 sum 0

x(k)              0  1  3  2  1  0
h(1-k)    -1  1 -1  1
                  0  1 sum 1

x(k)              0  1  3  2  1  0
h(2-k)       -1   1 -1  1
                  0 -1  3 sum 2

x(k)              0  1  3  2  1  0
h(3-k)           -1  1 -1  1
                  0  1 -3  2 sum 0
```

Continuing the process we find $x*h=[\underline{0},1,2,0,1,-2,-1,-1,0]$.

We can use Scilab's `conv` function to do the calculation.

Example 2: Use Scilab to work the last example.

```
-->x = [0,1,3,2,1,0];

-->h = [1,-1,1,-1];

-->y = conv(h,x)
 y  =

    0.    1.    2.    0.    1.  - 2.  - 1.  - 1.    0.

-->y = conv(x,h)
 y  =

    0.    1.    2.    0.    1.  - 2.  - 1.  - 1.    0.
```

Exercise 1: Compute the convolution of

$$x=[\mathring{1},2,3,2,1] \text{ and } h=[\mathring{1},0,-1]$$

by hand using the "flip and shift" method. Check your answer using Scilab's `conv` function.

Answer: $[0,1,2,0,1,-2,-1,-1,0]$.

The "tabular" method is somewhat more compact. Draw a rectangular array of cells, one row for each value h_i and one column for each value x_j. Label columns at top with the h_i value and label rows at right with h_j values (Fig. 3).

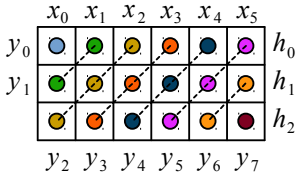

Fig. 3: Tabular method for computing convolution. Each cell contains a product $h_i x_j$. These are summed along diagonal lines to obtain y_n.

In each cell enter the product $h_i x_j$. Finally, sum along diagonal lines to obtain

$$\begin{aligned} y(0)&=h(0)x(0) \\ y(1)&=h(0)x(1)+h(1)x(0) \\ y(2)&=h(0)x(2)+h(1)x(1)+h(2)x(0) \end{aligned} \tag{12}$$

and so on.

Example 3: Use the tabular method to compute the convolution $y(n)=h(n)*x(n)$ for $h(n)=[\mathring{1},0.5,0.25]$, $x(n)=[\mathring{1},0,-1,-0.5,0.5,0.25]$

	1	0	-1	-0.5	0.5	0.25	
1	1	0	-1	-0.5	0.5	0.25	1
0.5	0.5	0	-0.5	-0.25	0.25	0.125	0.5
	0.25	0	-0.25	-0.125	0.125	0.0625	0.25
	-0.75	-1	0	0.375	0.25	0.0625	

We find $y(n)=[\mathring{1},0.5,-0.75,-1,0,0.375,0.25,0.0625]$.

Give this a try.

Exercise 2: Use the tabular method to compute the convolution $y(n)=h(n)*x(n)$ for $h(n)=[\mathring{1},-1]$, $x(n)=[\mathring{1},0.5,0.25]$. Check your answer using the Scilab `conv` command.

Answer: $y(n)=[\mathring{1},-0.5,-0.25,-0.25]$.