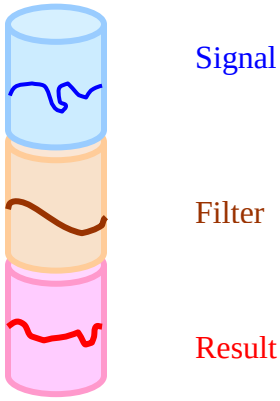
ADSP : Lecture 6

8th Feb 2004
Sunday [@ Codito, Pune]
Scribed by: Tanmay Dharmadhikari

The lecture started with circular convolution calculation viewable with cylinders. This technique is also useful when the filter kernel is larger (in sample size) than one period of the signal itself.

Circular Convolution: Using Cylinders

Consider the signal and filter to be marked on two different cylinders as shown in the figure below. For circular convolution:



a.) The filter cylinder and result cylinder both, are rotated 1 unit about their height axes.

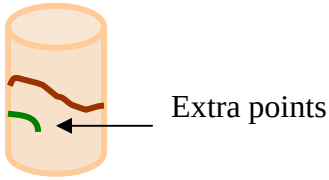
b.) The corresponding points on the signal cylinder and filter cylinders are multiplied with each other.

c.) These products are added to give the output at that point of the result cylinder.

d.) On performing these steps for all the points we unfurl the result cylinder to get the convolution result.

What if the filter size is larger than one period of the signal? The most intuitive thing is to change the size of the filter cylinder as well. Another solution that we can see is to mark the filter kernel on the cylinder spirally i.e. in rudimentary terms let the extra kernel points get marked below the kernel.

The extra points are also used during the convolution along with the corresponding point from top.



Thus during one cycle of our convolution process, for some points on the signal, we perform two multiplications.

This idea can be simplified by adding the extra points into the corresponding kernel points that align with them. Thus, we get a new kernel of the same size as the signal period.

With this clear we moved on to show that convolution of two signals is easier to calculate by

(i) Decomposing the signals into their spiral components

(ii) Convolving only those corresponding spirals which have the same frequency

(iii) And finally synthesizing all these individual results to get the final result i.e. the convolution of the signals.

Representing this algebraically:

Let x and f be the signal and filter we want to convolve. Lets decompose x and f into the spirals given by:

$$x = x_1 + x_2 + \ldots + x_n$$

$$f = f_1 + f_2 + \ldots + f_n$$

x_i, f_i are the corresponding spiral components of x and f with same frequencies

$$\begin{aligned}\therefore (x * f) &= (x_1 + x_2 + \ldots + x_n) * (f_1 + f_2 + \ldots + f_n) \\ &= (x_1 * f_1) + (x_1 * f_2) + \ldots (x_1 * f_n) \\ &\quad + (x_2 * f_1) + (x_2 * f_2) + \ldots (x_2 * f_n) \\ &\quad \ldots \\ &\quad + (x_n * f_1) + (x_n * f_2) + \ldots (x_n * f_n)\end{aligned}$$

Now, we can prove that if the two convolving spirals do not have the same frequency then their resultant is zero and that only those with same frequency contribute to the value of the final convolution result. Thus the above equation reduces to:

$$(x * f) = (x_1 * f_1) + (x_2 * f_2) + \ldots (x_n * f_n)$$

The convolution of x and f is thus reduced to the ‘n’ convolutions of the ‘n’ corresponding spiral components and adding their individual resultants.

This procedure is feasible because we know or rather have the data about each spiral component viz. :

(i) Starting phase

(ii) Individual frequency

(iii) Individual amplitude

The convolution result of each corresponding component pair is given by the spiral with:

(i) Starting phase = $x_{i\theta} + f_{i\theta}$

(ii) Frequency = $x_{i\omega} - f_{i\omega}$

(iii) Amplitude = $N \left| x_{i\omega} \right| \left| f_{i\omega} \right|$

[N is no. of sample points.]

We noted when to use which form of the Fourier Transform depending on the signal domain:

Signal Domain	**Transform**
Discrete & Finite	DFT
Continuous & Finite	Fourier Series
Discrete & Infinite	DTFT
Continuous & Infinite	FT

Shift Property:

Shifting a sequence in time results in the multiplication of the DTFT by a complex exponential (linear phase term):

$$x(n - n_0) \overset{\text{DTFT}}{\longleftrightarrow} e^{-jn_0 w} X(e^{jw})$$

We also saw the application of using rules governing linear systems to simplify convolution of continuous signals:

Modify one of the signals in some linear way, perform the convolution, and then undo the original modification. As an example we used the *derivative* as the modification and it is undone by taking the *integral*.

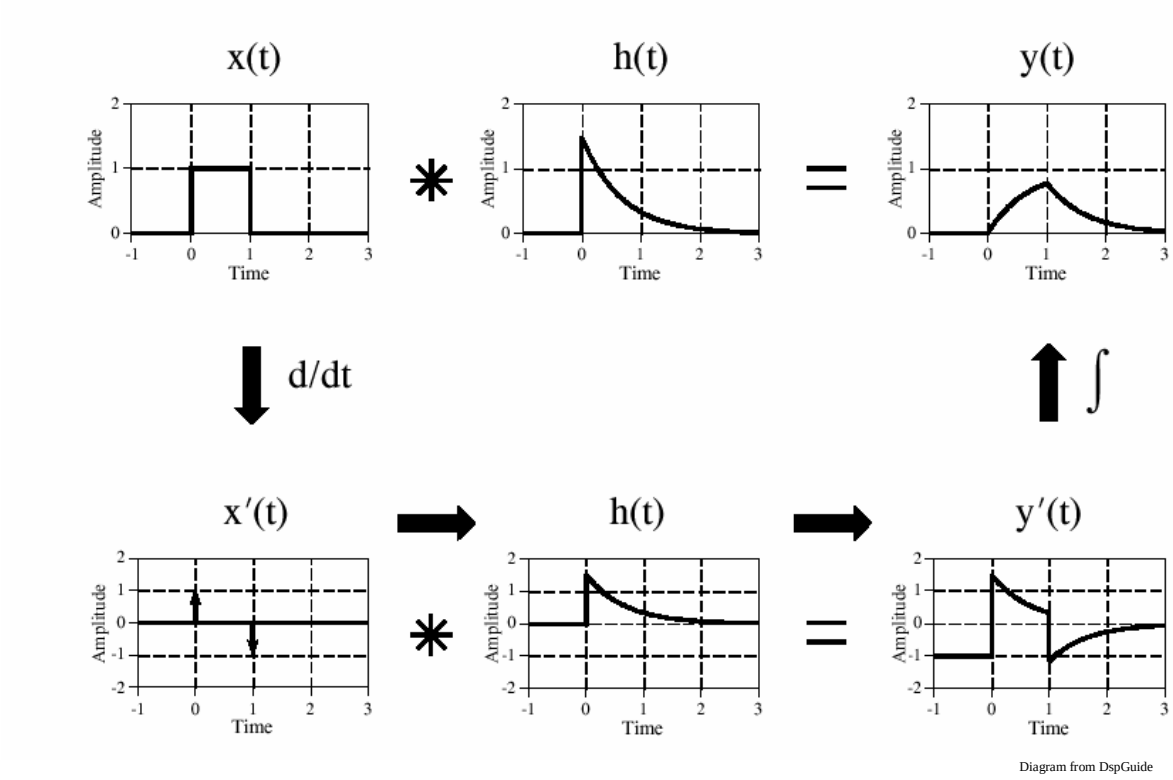


Diagram from DspGuide