Linear Convolution

The linear convolution of two sequences $x_n$ and $y_n$ is defined by

$$(x * y)_n := \sum_{m=-\infty}^{\infty} x_m y_{n-m}. $$

If one of the sequences, say $x_m$, has finite duration, say $x_m = 0$ for $m < M_1$ and $m > M_2$, then

$$(x * y)_n := \sum_{m=M_1}^{M_2} x_m y_{n-m}. $$

Let us further suppose that we are interested in the convolution only for $N_1 \leq n \leq N_2$, say for plotting purposes. Then in computing the convolution, the values of the subscript of $y$, that is, $n - m$, range from $N_1 - M_2$ to $N_2 - M_1$. The point here is that to compute $(x * y)_n$ for $n = N_1, \ldots, N_2$ requires only the values $x_{M_1}, \ldots, x_{M_2}$ and $y_{N_1 - M_2}, \ldots, y_{N_2 - M_1}$. For example, if $N_1 = M_1 = 0$, then we need $y_{-M_2}, \ldots, y_{N_2}$. [Draw pictures]

Let us now define $\hat{y}_n := y_n$ for $n = N_1 - M_2, \ldots, N_2 - M_1$ and $\hat{y}_n := 0$ otherwise. A graphical argument with $y_{n-m}, \hat{y}_{n-m},$ and $x_m$ shows that $(x * y)_n$ and $(x * \hat{y})_n$ are equal for $n = N_1, \ldots, N_2$. Fortunately, the convolution of two finite-length sequences is easily done with the MATLAB command $z = \text{conv}(x, \hat{y})$. However, since MATLAB computes $(x * \hat{y})_n$ for $n = N_1 - (M_2 - M_1)$ to $n = N_2 + (M_2 - M_1)$, the values $(x * \hat{y})_{N_1}, \ldots, (x * \hat{y})_{N_2}$ must be extracted from $z$ this can be done with the expression $z(M_2 - M_1 + 1 : \text{end} - (M_2 - M_1))$.

A nice application of our discussion arises if we approximate the sampled version of (1.14), i.e.,

$$\int_{-\infty}^{\infty} h(n/f_s - \tau)x(\tau)d\tau = \frac{1}{f_s} \sum_{m=-\infty}^{\infty} h([n-m]/f_s)x(m/f_s).$$

with a finite sum, say

$$\frac{1}{f_s} \sum_{m=M_1}^{M_2} h([n-m]/f_s)x(m/f_s).$$

Since (1.14) assumes $h$ is bandlimited, it cannot be time limited. However, when only finitely many values of $n$ are of interest, as would be the case for plotting the convolution, we can use conv as described.

How Does Circular Convolution with FFTs Compare with conv?

From our theoretical discussion, there is no question that the circular convolution of zero-padded sequences is faster than direct convolution. To see the difference in action, run the following MATLAB script, which computes the linear convolution of two causal, finite-duration sequences by both methods and reports the time taken by each.

```matlab
x=ones(1,2000); y=ones(1,5000); tic N = length(x)+length(y)-1; N = 2^ceil(log2(N)); % round up to a power of 2 v = ifft(fft(x,N).*fft(y,N)); ffttime = toc; tic w = conv(x,y); convtime = toc; plot([0:length(v)-1],real(v)); grid on fprintf('conv takes %g times longer.
',convtime/ffttime)
```

```matlab
x=ones(1,2000); y=ones(1,5000); tic N = length(x)+length(y)-1; N = 2^ceil(log2(N)); % round up to a power of 2 v = ifft(fft(x,N).*fft(y,N)); ffttime = toc; tic w = conv(x,y); convtime = toc; plot([0:length(v)-1],real(v)); grid on fprintf('conv takes %g times longer.
',convtime/ffttime)
```