Linear and Circular Convolution Properties

Recall the (linear) convolution property

\[ x_3[n] = x_1[n] \ast x_2[n] \quad \leftrightarrow \quad X_3(e^{j\omega}) = X_1(e^{j\omega})X_2(e^{j\omega}) \quad \forall \omega \in \mathbb{R} \]

if the necessary DTFTs exist. If \( x_1[n] \) is length \( N_1 \) and \( x_2[n] \) is length \( N_2 \), then \( x_3[n] \) will be length \( N_3 = N_1 + N_2 - 1 \). See MATLAB function conv.

For the DFT, we have the circular convolution property

\[ x_3[n] = x_1[n] \bigcirc N x_2[n] \quad \leftrightarrow \quad X_3[k] = X_1[k]X_2[k] \quad \forall k = 0, \ldots, N - 1 \]

where

\[ x_1[n] \bigcirc N x_2[n] = \sum_{k=0}^{N-1} x_1[k]x_2[\{(n - k)\}_N] = \sum_{k=0}^{N-1} x_1[\{(n - k)\}_N]x_2[k]. \]

Note that \( x_1[n] \) and \( x_2[n] \) must have the same length \( N = N_1 = N_2 \) and the result \( x_3[n] \) will also be length \( N \). See Matlab function cconv.
Linear Convolution with the DFT?

Suppose we want to compute

\[ x_3[n] = x_1[n] \ast x_2[n]. \]

We could compute the DTFTs of \( x_1[n] \) and \( x_2[n] \), take their product, and then compute the inverse DTFT to get \( x_3[n] \), i.e.,

\[ x_3[n] = \text{IDTFT}(\text{DTFT}(x_1[n]) \cdot \text{DTFT}(x_2[n])) \]

What if we want to use the DFT to compute the linear convolution instead? We know

\[ x_3[n] = \text{IDFT}(\text{DFT}(x_1[n]) \cdot \text{DFT}(x_2[n])) \]

will not work because this performs circular convolution.
Avoiding Time-Domain Aliasing

Recall our notation $W_M = e^{-j2\pi/M}$. We have seen previously that the $M$-point DFT of a finite-length sequence $x_i[n]$ with length $N_i$

$$X_i[k] = \sum_{n=0}^{N_i-1} x_i[n] W_M^{kn} \quad k = 0, 1, \ldots, M - 1$$

must satisfy $M \geq N_i$ to avoid time-domain aliasing when computing $\text{IDFT}_M(\text{DFT}_M(x_i[n]))$.

If $x_3[n] = x_1[n] * x_2[n]$ with $x_1[n]$ and $x_2[n]$ both finite length sequences, then the longest sequence is $x_3[n]$ with length $N_3 = N_1 + N_2 - 1$.

This implies that our DFTs $X_1[k]$, $X_2[k]$, and $X_3[k]$ should all be of length $M \geq N_1 + N_2 - 1$ to avoid time-domain aliasing. In other words,

$$x_3[n] = \text{IDFT}_M(\text{DFT}_M(x_1[n]) \cdot \text{DFT}_M(x_2[n]))$$

will result in $x_3[n] = x_1[n] * x_2[n]$ if $M \geq N_1 + N_2 - 1$. 
Example

Suppose $x_1 = [1, 2, 3]$ and $x_2 = [1, 1, 1]$. We can compute the linear convolution as

$$x_3[n] = x_1[n] \ast x_2[n] = [1, 3, 6, 5, 3].$$

If we instead compute

$$x_3[n] = \text{IDFT}_M(\text{DFT}_M(x_1[n]) \cdot \text{DFT}_M(x_2[n]))$$

we get

$$x_3[n] = \begin{cases} [6, 6, 6] & M = 3 \\ [4, 3, 6, 5] & M = 4 \\ [1, 3, 6, 5, 3] & M = 5 \\ [1, 3, 6, 5, 3, 0] & M = 6 \end{cases}$$

Observe that time-domain aliasing of $x_3[n]$ is avoided for $M \geq 5$. 

Example (continued)
Zero-padding avoids time-domain aliasing and make the circular convolution behave like linear convolution.

$M$ should be selected such that $M \geq N_1 + N_2 - 1$.

In practice, the DFTs are computed with the FFT.

The amount of computation with this method can be less than directly performing linear convolution (especially for long sequences).

Since the FFT is most efficient for sequences of length $2^m$ with integer $m$, $M$ is usually chosen so that $M = 2^m \geq N_1 + N_2 - 1$. 