FFT Implementation of Linear Systems

Julius O. Smith III (jos@ccrma.stanford.edu)  
Center for Computer Research in Music and Acoustics (CCRMA)  
Department of Music, Stanford University  
Stanford, California 94305

April 10, 2018

- Overlap Add
Fourier Implementation of LTI systems (Overlapp add summation)

Linear Convolution of finite length signals

Recall the convolution theorem:

\[(x * y)(n) \leftrightarrow X(\omega_k)Y(\omega_k)\]

It is important to remember that the specific form of convolution that implied in the above equation is circular convolution.

\[y(n) = \sum_{m=0}^{N-1} x(m)h(n - m)_N\]

where \((n - m)_N\) means \((n - m) \text{ modulo } N\)

Another way to look at this is as the inner product of \(x\), and \(\text{SHIFT}_n[\text{FLIP}(h)]\) i.e.:

\[y(n) = \langle x, \text{SHIFT}_n[\text{FLIP}(h)] \rangle\]

The convolution theorem shows us that there are 2 ways to perform circular convolution.
• direct calculation of the summation
• freq domain approach
  – FT both signals
  – perform term by term multiplication of the transformed signals
  – inverse transform the result to get back to the time domain

Remember ... this still gives us cyclic convolution

Note that if our signals are short enough, then we have no problems

Idea: If we add enough zeros to the above signals, we can end up with the same results as linear convolution

How many zeros do we need to add?

• If we perform the convolution of 2 signals, $x$ and $h$, with lengths $N_x$ and $N_h$, the resulting signal is length $N = N_x + N_h - 1$

• We must add enough zeros so that our result is the appropriate length

  – if we don’t add enough zeros, some of our terms are added back upon itself (in a circular fashion)
– this can be thought of as time domain aliasing

A sampling theorem based insight:

Adding zeros in the time domain results in more samples (closer spacing) in the frequency domain. This can be thought of as a higher 'sampling rate' in the frequency domain. If we have a high enough sampling rate, we can avoid time domain aliasing.

The motivation for going through this exercise, is that there is an efficient way to calculate the DFT. This is known as the Fast Fourier Transform FFT. (See Oppenheim and Schafer for details)

Lets compare the number of operations needed to perform the convolution of 2 length $N$ sequences:

- It takes $N^2$ multiply/add operations to calculate the convolution summation directly.
- It takes $N\log(N)$ operations to compute an FFT. Note: $H$ can be calculated ahead of time

**Infinite Length signals**

We saw that we can perform efficient linear convolution of 2 finite length sequences using the Fourier based techniques
There are some situations where it will not be practical to perform the convolution of 2 signals:

- \( N_x \) is extremely large
- real time operation (we can’t wait around until the signal ends)

Theoretically, there is no problem doing this with direct convolution. Since \( h \) is finite in length we only need to store the last \( N_h \) samples of the input signal \( x \) to calculate the next output.

Idea: We might be able to perform convolution on a block at time. Basically, we chop up the input signal, \( x \) by windowing, and performing frequency domain convolution on each block seperately.

Problem: We need to make sure we put it all back together correctly.

Consider breaking the input signal \( x \), into frames(using a finite length window function) where \( x_m \) denotes the mth frame.

\[
x_m \triangleq x(n)w(n - mR)
\]

where:

\( R \triangleq \) frame step (hop size) \hspace{1cm} m \triangleq frame index
In order for this to work, we need to be sure that we can reconstruct $x$ from the individual frames. This constraint can be written as:

$$x(n) = \sum_{m=-\infty}^{\infty} x_m(n)$$

$$= \sum_{m=-\infty}^{\infty} x(n)w(n - mR)$$

$$= x(n) \sum_{m=-\infty}^{\infty} w(n - mR)$$

Hence, $x = \sum_m x_m$ iff $\sum_m w(n - mR) = 1$

In the frequency domain, we have the following:

$$X = F(x)$$

$$= F(\sum_m x_m)$$

$$= \sum_m F(x_m)$$

$$= \sum_m X_m$$
where \( X_m \stackrel{\Delta}{=} F(x_m), \) and
\[
\tilde{x}_m \stackrel{\Delta}{=} x_m(n + mR) = \text{SHIFT}_{mR}(x_m)
\]
This is summarized in the following diagram:

In practice we will use \( \tilde{X}_0, \tilde{X}_1, \ldots \)

Putting this all together we have the following:

( remember \( x, y, h \in \mathbb{C}^\infty \) )

\[
y = x \ast h \\
= \left( \sum_m x_m \right) \ast h \\
= \sum_m (x_m \ast h) \\
= \sum_m (\text{SHIFT}_{mR}(\tilde{x}_m \ast h)) \\
= \sum_m \text{SHIFT}_{mR}(\tilde{x}_m \ast h) \\
= \sum_m \text{SHIFT}_{mR}(\text{DTFT}^{-1}(\tilde{X}_m \hat{H}))
\]

In the last equation, \( \tilde{X}_m = \text{DTFT}(\tilde{x}_m, \) and \( H = \text{DTFT}(h) \)
We aren’t quite done ... In the last equation, we still need to calculate the DTFTs (infinite duration). At this point we need to make use of the following facts:

- $\tilde{x}_m$ is time limited
- We need to assume $h$ is time limited to $N_h$
- Then $\tilde{x}_m * h$ will be time limited to $M + N_h - 1$

Since $\tilde{x}_m * h$ is time limited, you can sample its DTFT at intervals $\Omega \leq \frac{2\pi}{N_h + M - 1}$

( This is the dual of the sampling theorem. )

This last observation implies that we can get by with a length $N \geq N_h + M - 1$ DFT

Important points:

- $x$ is infinite, or of an indefinite length
- $h$ can be any FIR filter
- $h$ can be time varying
  ( We will see the implications of this soon )

- Using the FFT gives us computational efficiency
  ( 100,000 tap filters at audio rates with standard DSP chips! )