Lecture 24: Overlap-Add

Mark Hasegawa-Johnson

ECE 401: Signal and Image Analysis

Outline

Review: Circular convolution

Multiplying the DFT means **circular convolution** of the time-domain signals:

$$y[n] = h[n] \circledast x[n] \leftrightarrow Y[k] = H[k]X[k],$$

Circular convolution ($h[n] \circledast x[n]$) is defined like this:

$$h[n] \circledast x[n] = \sum_{m=0}^{N-1} x[m]h\left[((n-m))_N\right] = \sum_{m=0}^{N-1} h[m]x\left[((n-m))_N\right]$$

Circular convolution is the same as linear convolution if and only if $N \geq L + M - 1$.

Outline

Computational Complexity: Convolution and DFT

Convolution is an $O\{N^2\}$ operation: each of the N samples of $y[n]$ is created by adding up N samples of $x[m]h[n-m]$:

$$y[n] = \sum_m x[m]h[n-m]$$

The way we've learned it so far, the DFT is **also** an $O\{N^2\}$ operation: each of the N samples of $X[k]$ is created by adding up N samples of $x[n]e^{j\omega_k n}$:

$$X[k] = \sum_n x[n]e^{-j\frac{2\pi kn}{N}}$$

However...

The Fast Fourier Transform

- The **fast Fourier transform** (FFT) is a clever divide-and-conquer algorithm that computes all of the N samples of $X[k]$, from $x[n]$, in only $N\log_2 N$ multiplications.
- It does this by computing all N of the $X[k]$, all at once.
- Multiplications ($x[n] \times w_{k,n}$, for some coefficient $w_{k,n}$) are grouped together, across different groups of k and n.
- On average, each of the N samples of $X[k]$ can be computed using only $\log_2 N$ multiplications, for a total complexity of $N\log_2 N$.

What's the difference between N^2 and $N \log_2 N$?

Consider filtering $N = 1024$ samples of audio (about 1/40 second) with a filter, $h[n]$, that is 1024 samples long.

- Time-domain convolution requires $1024 \times 1024 \approx 1,000,000$ multiplications. If a GPU does 40 billion multiplications/second, then it will take an hour of GPU time to apply this operation to a 1000-hour audio database.
- FFT requires $1024 \times \log_2 1024 \approx 10,000$ multiplications. If a GPU does 40 billion multiplications/second, then it will take 36 seconds of GPU time to apply this operation to a 1000-hour audio database.

How is it used?

Suppose we have a 1025-sample $h[n]$, and we want to filter a one-hour audio (144,000,000 samples). Divide the audio into frames, $x[n]$, of length $M = 1024$, zero-pad to $N = L + M - 1 = 2048$, and take their FFTs.

- $H[k] = \mathrm{FFT}\{h[n]\}$: total cost is trivial, because we only need to do this once.
- $X[k] = \mathrm{FFT}\{x[n]\}$: total cost is $N \log N$ per M samples.
- $Y[k] = X[k]H[k]$: total cost is N multiplications per M samples.
- $y[n] = \mathrm{FFT}^{-1}\{Y[k]\}$: total cost is $N \log N$ per M samples.

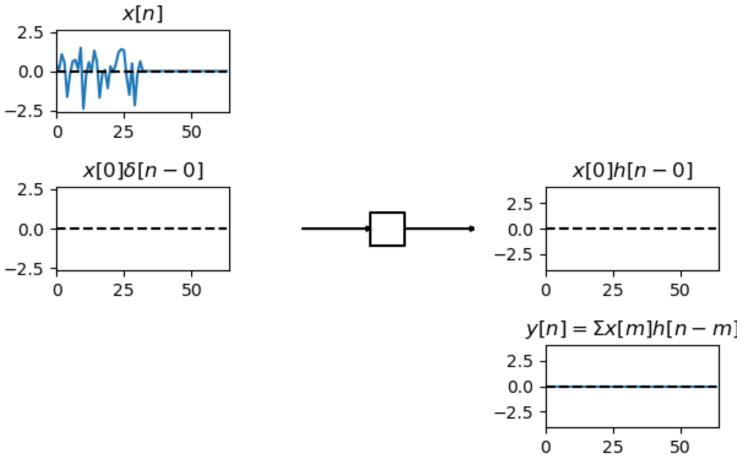
Grand total: $N \times (2 \log_2 N + 1) = 2048 \times 23 = 47104$ multiplications per 1024 audio samples, or 46 multiplications per sample.

How do we recombine the $y[n]$?

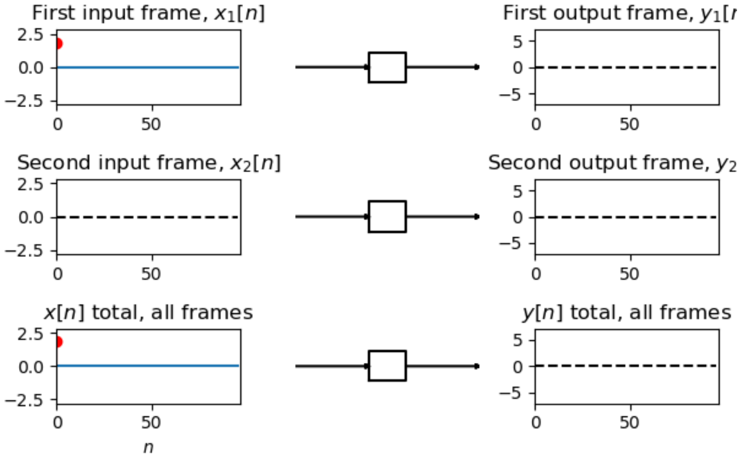
- The main topic of today's lecture: how do we recombine the $y[n]$?
- Remember: each frame of $x[n]$ was 1024 samples, but after zero-padding and convolution, each frame of $y[n]$ is 2048 samples.
- How do we recombine them?

Outline

Let's look more closely at what convolution is. Each sample of x[n] generates an impulse response. Those impulse responses are added together to make the output.



First two lines show the first two frames (input on left, output on right). Last line shows the total input (left) and output (right).



The Overlap-Add Algorithm

1. Divide $x[n]$ into frames
2. Generate the output from each frame
3. Overlap the outputs, and add them together

The Overlap-Add Algorithm

1. Divide $x[n]$ into frames ($w[n]$ is a length-M rectangle).

$$x_t[n] = x[n + tM]w[n]$$
$$X_t[k] = \text{FFT}\{x_t[n]\}$$

2. Generate the output from each frame

$$Y_t[k] = X_t[k]H[k]$$
$$y_t[n] = \text{FFT}^{-1}\{y_t[n]\}$$

3. Overlap the outputs, and add them together

$$y[n] = \sum_t y_t[n - tM]$$

Outline

Written Example

- Suppose you have a billion samples of audio (about 6 hours' worth), and you want to convolve it with a 1025-sample lowpass filter. How many multiplications are required to do this using time-domain convolution? How many using overlap-add?
- Suppose that the audio is periodic, with a period of 1024 samples. Each period is 600 ones, followed by 424 zeros. Suppose that the filter is

  $$h[n] = a^n, \quad 0 \le n \le 1024$$

  Use overlap-add (but with convolutions, not FFT) to find the first 2048 samples of $y[n]$