ECE 161A: Linear Convolution Using DFT/FFT

Florian Meyer
University of California, San Diego
Email: flmeyer@ucsd.edu

Fast Fourier Transform (FFT)

$$\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x[n] W_N^{nk}, k = 0, 1, \ldots, N-1 \text{ (DFT)} \\
x[n] &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}kn} = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-nk}, n = 0, 1, \ldots, N-1 \text{ (IDFT)} \\
&= \frac{1}{N} \left(\sum_{k=0}^{N-1} X^*[k] W_N^{nk} \right)^* = \frac{1}{N} \left(\mathcal{DFT}(X^*[k]) \right)^*
\end{aligned}$$

DFT Direct Computation: N^2 complex multiplies assuming $x[n]$ is complex.

DFT Computation via FFT for $N = 2^\nu$: $\frac{N}{2} \log_2 N$ complex multiplies.

Example: For $N = 1024$, we have $1024^2 \approx 10^6$ for the direct computation versus $512 \log_2 1024 = 5120$ via the FFT

Similar comments can be made about number of additions.

Complexity of Linear Convolution

Filtering example: $h[n]$ of duration $P = 100$ and input sequence $x[n]$ of duration $L = 157$. Assuming everything is complex.

Direct Convolution $y[n] = \sum_{m=0}^{P-1=99} h[m]x[n-m]$: Every value of $y[n]$ requires 100 complex multiplies.

For this direct approach one can save a little in the first 99 samples and the last 99 samples, but the comparative conclusion we will reach holds.

For simplicity we will assume 100 complex multiplies per output sample.

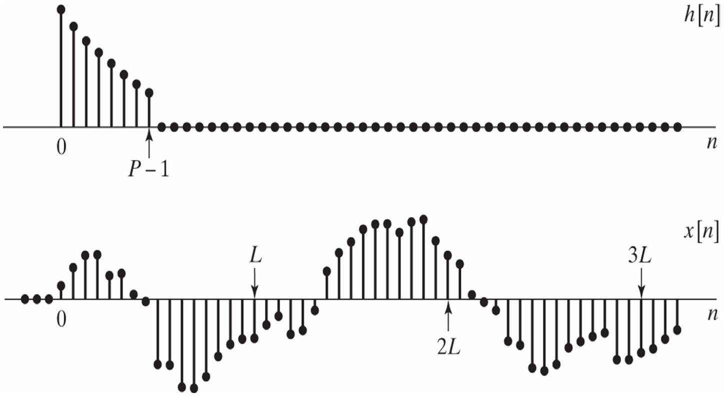
Complexity of FFT Based Linear Convolution

FFT based: Choose $N = L + P - 1 = 256$, and note that a DFT and IDFT are of complexity $\frac{N}{2}\log_2 N$.

1. Computation of $H[k]$ and $X[k]$ each require $128 \log_2 256$ complex multiplies for a total complexity of $256 \log_2 256 = 2048$ complex multiplies.
2. Computation of $Y[k] = H[k]X[k]$ requires 256 complex multiplies.
3. The IDFT of $Y[k]$ to obtain $y[n]$ requires $128 \log_2 256 = 1024$ complex multiplies.

Total number of complex multiplies is $2048 + 256 + 1024 = 3328$ complex multiplies. The total number of linear convolution samples is 256, and so the complexity per output sample is $\frac{3328}{256} = 13$ complex multiplies. This is in contrast to 100 complex multiplies per output sample for the direct approach.

Filtering Long Input Sequences



Linear Convolution Involving Long Input Sequences

Usually the input sequence $x[n]$ is very long.

The standard approach is no longer practical for the following reasons:

1. For real time applications, one can not wait till all the samples are available. This introduces too much delay.
2. In some applications, one may has access to FFT hardware that can do a fixed size DFT efficiently. So we are not free to choose an arbitrary size DFT.

What are the options?

1. Divide the data into smaller segments
    - Overlapping or Non-overlapping segments are two options
2. Operate on each segment
3. Stich the results from each segment to get overall convolution
    - Need to be careful at the boundaries of the segments

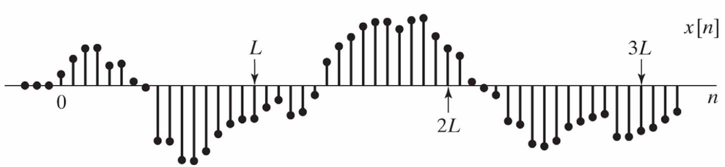
Methods for Dealing with Long Input Sequences

Two popular Options:

1. Overlap and Add Method
   - Non-Overlapping input segments
   - Output of the Convolution with each segment overlap and need to be added.
2. Overlap and Save Method
   - Overlapping input Segments
   - Output of each convolution need to be windowed and assembled. No additions.

Overlap and Add Method

Divide $x[n]$ into non-overlapping segments of length L.



$$x[n] = \sum_{r=0}^{\infty} x_r[n]$$

where $x_r[n] = x[n]w[n - rL]$, and $w[n]$ is a rectangular window of duration L.
or $x_r[n]$ has L non-zero starting at rL and ending at $rL + L - 1$, i.e. the non-zero samples are $\{x[rL], x[rL+1], \ldots, x[rL+L-1]\}$.

$$y[n] = x[n] * h[n] = \sum_{r=0}^{\infty} x_r[n] * h[n] = \sum_{r=0}^{\infty} (x_r[n] * h[n]) = \sum_{r=0}^{\infty} y_r[n]$$

where $y_r[n] = x_r[n] * h[n]$.

1. $y_r[n]$ is non-zero over the interval rL to $rL + L + P - 2$, i.e. $L + P - 1$ non-zero values.
2. $y_r[n]$ overlaps with $y_{r+1}[n]$ over the range $(r+1)L$ to $rL + L + P - 2$, i.e. $(P - 1)$ samples overlap.

Conclusion: a) Carry out the linear convolution for each segment $x_r[n]$ using a $N = L + P - 1$ point DFT to get $y_r[n]$. b) Add the convolution results to obtain $y[n]$. Note that first $(P - 1)$ samples of $y_r[n]$ will overlap with $y_{r-1}[n]$ and the last $(P - 1)$ samples with $y_{r+1}[n]$.

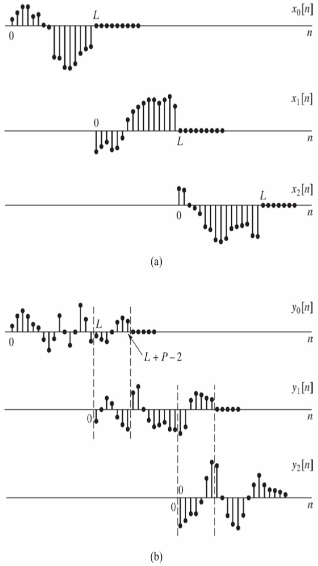
To carry out the convolution of each segment, only use the non-zero samples padded with $P - 1$ zeros.

$$g_r[n] = x_r[n + rL] * h[n], \;\; 0 \le n \le N - 1$$

which is computed by using a $N = L + P - 1$ point DFT procedure and $y_r[n] = g_r[n - rL]$.

Overlap and Add Method Cont'd

Figure 8.23 (a) Decomposition of $x[n]$ in Figure 8.22 into nonoverlapping sections of length L. (b) Result of convolving each section with $h[n]$.



Overlap and Save Method

Goal: a) Is to replace the adding that is required at the output of the overlap and add method with a simpler approach. b) Make each segment self sufficient.

Divide $x[n]$ into overlapping segments of length L.

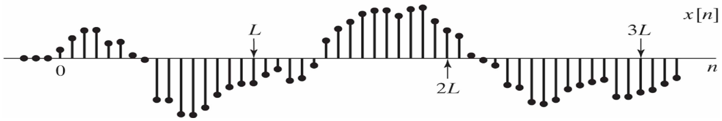
Why overlapping and by how much?

$y[n] = \sum_{m=0}^{P-1} h[m]x[n-m]$. Observe that each output sample requires the current input sample $x[n]$ and $(P-1)$ past samples $x[n-1], .., x[n-P+1]$.

This will cause problems at the boundaries of non-overlapping segments.

So segments must overlap by $(P-1)$ samples

To take care of sample $y[0]$, the first segment must be padded with $(P-1)$ zeros in the beginning of the sequence.



Again note that at the boundaries of the non-overlapping segments we have problem which introduces dependencies between segments.

Overlapping Segments:

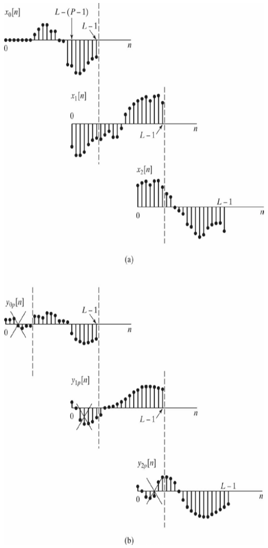
1. Each segment is of length L
2. $x_0[n]$, segment zero,uses samples $-(P-1) \leq n \leq L-P$. Now $x_1[n]$, segment 1, is formed from the nonzero samples in the range $L-2P+2 \leq n \leq 2(L-P)+1$. $x_1[n]$ overlaps $x_0[n]$ by $(P-1)$ samples
3. $x_3[n]$, segment 3, uses samples $2L-3P+3 \leq n \leq 3(L-P)+2$. $x_3[n]$ overlaps $x_2[n]$ by $(P-1)$ samples.
4. We follow the same process of segmenting using overlapping segments, overlapping by $(P-1)$ samples. $x_r[n]$, the rth segment, uses samples $rL-(r+1)P+(r+1) \leq n \leq (r+1)(L-P)+r$

Procedure:

1. Compute circular convolution for each overlapping segment ($P - 1$ samples overlap) using L point DFT (No zero padding).
2. For each segment convolution output, throw away the first $(P - 1)$ samples which do not correspond to linear convolution.
3. Simply concatenate the remaining samples.

Overlap and Save Method Cont'd

Figure 8.24 (a) Decomposition of $x[n]$ in Figure 8.22 into overlapping sections of length L. (b) Result of convolving each section with $h[n]$. The portions of each filtered section to be discarded in forming the linear convolution are indicated.



Example: Linear Convolution

$h[n]=$	1	1	1	$(P=3)$							
$x[n]=$	10	9	8	7	6	5	4	3	2	1	$(L=10)$

Linear Convolution length: $L+P-1=12$.

$y[0]=10$
$y[1]=10+9=19$
$y[2]=10+9+8=27$
$y[3]=9+8+7=24$
$y[4]=8+7+6=21$
$y[5]=7+6+5=18$
$y[6]=6+5+4=15$
$y[7]=5+4+3=12$
$y[8]=4+3+2=9$
$y[9]=3+2+1=6$
$y[10]=2+1=3$
$y[11]=1$

Overlap-Add Using 8 Point DFT

$P = 3, L = 8 - P + 1 = 6.$

$x_0[n] =$ 10 9 8 7 6 5
$x_1[n] =$ 4 3 2 1 0 0

$y_0[n] = x_0[n] \textcircled{8} h[n] =$ 10 19 27 24 21 18 11 5
$y_1[n] = x_1[n] \textcircled{8} h[n] =$ 4 7 9 6 3 1 0 0

Note: $y_l[n] = x_l[n] \textcircled{8} h[n] = x_l[n] * h[n]$

Then combine $y_0[n]$ and $y_1[n]$ to get $y[n]$.

index	0	1	2	3	4	5	6	7	8	9	10	11
$y_0[n]$	10	19	27	24	21	18	11	5				
$+$												
$y_1[n]$							4	7	9	6	3	1
$=$												
$y[n]$	10	19	27	24	21	18	15	12	9	6	3	1

So $y[n] =$ 10 19 27 24 21 18 15 12 9 6 3 1

Overlap-Save Using 8 Point DFT

$P = 3, L = 8.$

$x_0[n] =$ 0 0 10 9 8 7 6 5
$x_1[n] =$ 6 5 4 3 2 1 0 0

$y_0[n] = x_0[n] \textcircled{8} h[n] =$ 11 5 10 19 27 24 21 18
$y_1[n] = x_1[n] \textcircled{8} h[n] =$ 6 11 15 12 9 6 3 1

Then combine $y_0[n]$ and $y_1[n]$ to get $y[n]$.

index	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11
$y_0[n]$	(11	5)	10	19	27	24	21	18						
$+$														
$y_1[n]$							(6	11)	15	12	9	6	3	1
$=$														
$y[n]$			10	19	27	24	21	18	15	12	9	6	3	1

The numbers in () are discarded because of the aliasing of circular convolution.
So $y[n] =$ 10 19 27 24 21 18 15 12 9 6 3 1

Comparison of the two Methods

Filter length P, and Data length T (which is large). DFT size is fixed at N.

Overlap and Add Method:

Since we implement linear convolution using DFT, $N = L + P - 1$. So segment size $L = N - P + 1$.
Number of non-overlapping segments is $\frac{T}{L} = \frac{T}{N-P+1}$.

Overlap and Save Method:

Segment size $L = N$, but because of overlap of $(P - 1)$ samples, the number of new samples per segment is $L - (P - 1) = N - P + 1$.
Number of overlapping segments is $\frac{T}{N-P+1}$.

Conclusion: Complexity is the same. Choice depends on which code is simpler and executes faster!