Digital Signal Processing
Fast FIR Filtering with the Fast Fourier Transform

D. Richard Brown III

FIR Filtering Complexity Analysis

Suppose you have a causal FIR filter with length-N impulse response $h[n]$. At each time n, we can compute the output of the filter (direct form) as

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k]$$

which requires N multiplications and $N-1$ accumulates at each time n.

Suppose we have a length-N input signal so that $y[n]$ is length $2N-1$. The total number of MACs to compute $y[n]$ is $\mathcal{O}(N^2)$.

We also know that we could compute

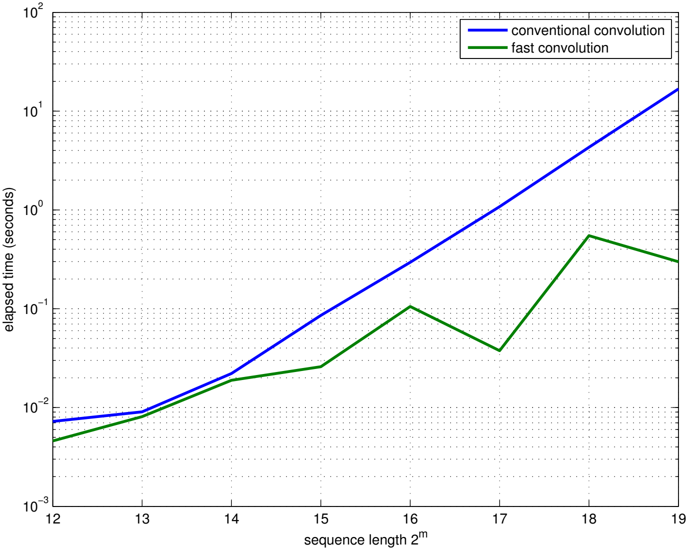
$$y[n] = \mathsf{IFFT}_{2N-1}(\mathsf{FFT}_{2N-1}(h[n]) \cdot \mathsf{FFT}_{2N-1}(x[n]))$$

where the FFT and IFFT require $\mathcal{O}(N \log_2 N)$ MACs. For sufficiently large N, this approach should be faster.

Matlab Example

```
% Example showing fast convolution with FFT
m_test = 12:19;
results = zeros(2,length(m_test));
i1 = 0;
for m=m_test,
    i1 = i1+1;
    N = 2^m;                % length of sequences
    x1 = randn(1,N);        % make sequence 1
    x2 = randn(1,N);        % make sequence 2
    tic
    x3c = conv(x1,x2);      % conventional convolution
    results(1,i1) = toc;
    tic
    x3f = ifft(fft(x1,2*N-1).*fft(x2,2*N-1)); % fast convolution
    results(2,i1)= toc;
end
semilogy(m_test,results,'Linewidth',2); grid on
xlabel('sequence length 2^m'); ylabel('elapsed time (seconds)');
legend('conventional convolution','fast convolution');
```

Example Results



What if $x[n]$ is an Infinite Length Sequence?

In a real-time DSP scenario, the input sequence is usually not finite length. Can we still use fast convolution?

The answer is yes but we have to break $x[n]$ into blocks, process each block separately, and carefully re-assemble the results.

There are two common methods (both based on block processing):

1. "Overlap-and-add"
2. "Overlap-and-save"

Notation:

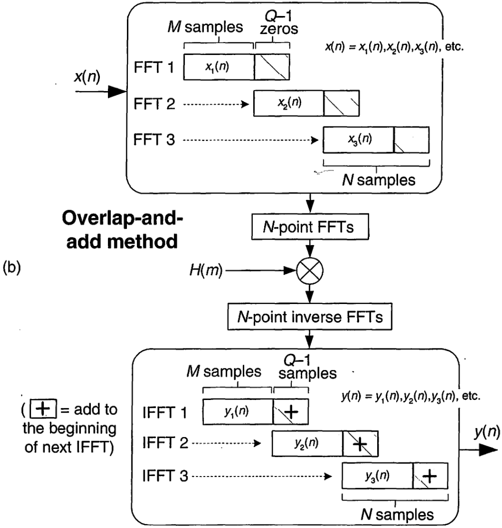
- FIR filter length: Q
- FFT length: N
- Input block length: $M = N - (Q - 1)$.

Overlap-and-Add

Main steps:

1. Select FFT size N such that $N = 2^m$ and $N \approx 2Q$.
2. Compute N-point FFT $h[n] \rightarrow H[m]$ for $m = 0, \dots, N-1$.
3. Let $M = N - (Q-1)$.
4. Fill length-M block $x_i[n]$ and append $Q-1$ zeros.
5. Take N-point FFT $x_i[n] \rightarrow X_i[m]$ for $m = 0, \dots, N-1$.
6. Compute $Y_i[m] = X_i[m]H[m]$ for $m = 0, \dots, N-1$.
7. Compute N-point IFFT $Y_i[m] \rightarrow y_i[n]$ for $n = 0, \dots, N-1$
8. Add the first $Q-1$ samples of $y_i[n]$ to the last $Q-1$ samples of $y_{i-1}[n]$.
9. Go to step 4.

Remark: Different values of N may give better results (less computation).



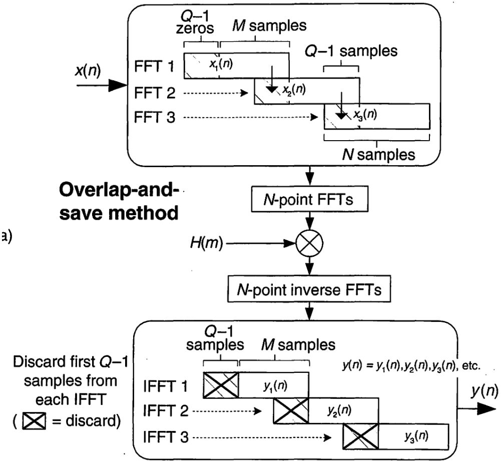
(Figure from Richard G. Lyons "Understanding Digital Signal Processing" (Prentice Hall)).

Overlap-and-Save

Main steps:

1. Select FFT size N such that $N = 2^m$ and $N \approx 4Q$.
2. Compute N-point FFT $h[n] \rightarrow H[m]$ for $m = 0, \dots, N-1$.
3. Let $M = N - (Q-1)$.
4. Fill length-M block $x_i[n]$ and prepend the last $Q-1$ samples from $x_{i-1}[n]$.
5. Take N-point FFT $x_i[n] \rightarrow X_i[m]$ for $m = 0, \dots, N-1$.
6. Compute $Y_i[m] = X_i[m]H[m]$ for $m = 0, \dots, N-1$.
7. Compute N-point IFFT $Y_i[m] \rightarrow y_i[n]$ for $n = 0, \dots, N-1$
8. Discard the first $Q-1$ samples of $y_i[n]$.
9. Go to step 4.

Remark: Different values of N may give better results (less computation).



(Figure from Richard G. Lyons "Understanding Digital Signal Processing" (Prentice Hall)).

Remarks

1. Computational requirements do not change with FIR filter length Q unless N is changed.
2. FFT of $h[n]$ only needs to be computed once.
3. Choosing between overlap-and-add vs. overlap-and-save depends on several factors, including:
    3.1 Fixed/floating point arithmetic
    3.2 Memory constraints
    3.3 Latency constraints
    3.4 Hardware architecture, e.g., pipelining
    3.5 Specialized DSP instruction sets