The Fast Fourier Transform

Professor Deepa Kundur

University of Toronto

The Fast Fourier Transform

Reference:

Section 8.1 of

John G. Proakis and Dimitris G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 4th edition, 2007.

The N-Point DFT

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi k \frac{n}{N}}, \quad k = 0, 1, \ldots, N-1$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi k \frac{n}{N}}, \quad n = 0, 1, \ldots, N-1$$

New notation: $W_N = e^{-j\frac{2\pi}{N}}$

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0, 1, \ldots, N-1$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn}, \quad n = 0, 1, \ldots, N-1$$

Complexity of the N-Point DFT

$$X(k) = \sum_{n=0}^{N-1} x(n) \times W_N^{kn}, \quad k = 0, 1, \ldots, N-1$$

Straightforward implementation of DFT to compute $X(k)$ for $k = 0, 1, \ldots, N-1$ requires:

- N^2 complex multiplications
  - 1 complex mult = $(a_R + ja_I) \times (b_R + jb_I) = (a_R \times b_R - a_I \times b_I) + j(a_R \times b_I + a_I \times b_R)$ = 4 real mult + 2 real add
  - $4N^2 = O(N^2)$ real multiplications

Complexity of the N-Point DFT

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0, 1, \ldots, N-1$$

Straightforward implementation of DFT to compute $X(k)$ for $k = 0, 1, \ldots, N-1$ requires:

- $N(N-1)$ complex additions
  - 1 complex add = $(a_R + ja_I) + (b_R + jb_I) = (a_R + b_R) + j(a_I + b_I) = 2$ real add
  - $2N(N-1) + 2N^2$ (from complex mult) real additions $= 2N(2N-1) = O(N^2)$ real additions.

Complexity of the N-Point DFT

- Is $O(N^2)$ high?
  - Yes. A linear increase in the length of the DFT increases the complexity by a power of two.
  - Given the multitude of applications where Fourier analysis is employed (linear filtering, correlation analysis, spectrum analysis), a method of efficient computation is needed.

Complexity of the N-Point DFT

- How can we reduce complexity?
  - Exploit symmetry of the complex exponential.

$$\begin{aligned}
W_N^{k+\frac{N}{2}} &= -W_N^k \\
\text{LHS} = W_N^{k+\frac{N}{2}} &= e^{-j2\pi\frac{k+N/2}{N}} = e^{-j2\pi\frac{k}{N}} e^{-j2\pi\frac{N/2}{N}} \\
&= e^{-j2\pi\frac{k}{N}} e^{-j\pi} \\
&= e^{-j2\pi\frac{k}{N}} \cdot (\cos(-\pi) + j\sin(-\pi)) \\
&= e^{-j2\pi\frac{k}{N}} (-1) \\
&= -e^{-j2\pi\frac{k}{N}} = -W_N^k = \text{RHS}
\end{aligned}$$

Complexity of the N-Point DFT

- How can we reduce complexity?
  - Exploit periodicity of the complex exponential.

$$\begin{aligned}
W_N^{k+N} &= W_N^k \\
\text{LHS} = W_N^{k+N} &= e^{-j2\pi\frac{k+N}{N}} = e^{-j2\pi\frac{k}{N}} e^{-j2\pi\frac{N}{N}} \\
&= e^{-j2\pi\frac{k}{N}} e^{-j2\pi} \\
&= e^{-j2\pi\frac{k}{N}} \cdot (\cos(-2\pi) + j\sin(-2\pi)) \\
&= e^{-j2\pi\frac{k}{N}} (1) \\
&= e^{-j2\pi\frac{k}{N}} = W_N^k = \text{RHS}
\end{aligned}$$

Radix-2 FFT

- We will demonstrate how to exploit the symmetry and periodicity of W_N^k:
  - to make an N-Point DFT look like two $N/2$-Point DFTs;
  - to make an $N/2$-Point DFT look like two $N/4$-Point DFTs;
  - to make an $N/4$-Point DFT look like two $N/8$-Point DFTs;
- The halving of the DFT length each time gives the name Radix-2 FFT.

Note: We use the convention N-DFT to specify an N-Point DFT.

Radix-2 FFT

Two strategies:

- Decimation in time (our focus in the lecture)
- Decimation in frequency

- Note: We assume that N is a power of two; i.e., $N = 2^r$.

Radix-2 FFT: Decimation-in-time

$$
\begin{aligned}
X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{kn} \quad k = 0, 1, \ldots, N-1 \\
&= \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn} \\
&= \sum_{m=0}^{(N/2)-1} x(2m) W_N^{k(2m)} + \sum_{m=0}^{(N/2)-1} x(2m+1) W_N^{k(2m+1)} \\
&= \sum_{m=0}^{(N/2)-1} \underbrace{x(2m)}_{\equiv f_1(m)} W_N^{2km} + \sum_{m=0}^{(N/2)-1} \underbrace{x(2m+1)}_{\equiv f_2(m)} W_N^{2km} W_N^k
\end{aligned}
$$

Radix-2 FFT: Decimation-in-time

Note: $W_N^2 = e^{-j\frac{2\pi}{N}\cdot 2} = e^{-j\frac{2\pi}{N/2}} = W_{N/2}$

$$
\begin{aligned}
X(k) &= \sum_{m=0}^{(N/2)-1} \underbrace{x(2m)}_{\equiv f_1(m)} W_N^{2km} + \sum_{m=0}^{(N/2)-1} \underbrace{x(2m+1)}_{\equiv f_2(m)} W_N^{2km} W_N^k \\
&= \underbrace{\sum_{m=0}^{(N/2)-1} f_1(m) W_{N/2}^{km}}_{\frac{N}{2}-\text{DFT of } f_1(m)} + W_N^k \underbrace{\sum_{m=0}^{(N/2)-1} f_2(m) W_{N/2}^{km}}_{\frac{N}{2}-\text{DFT of } f_2(m)} \\
&= F_1(k) + W_N^k F_2(k), \quad k = 0, 1, \ldots, N-1
\end{aligned}
$$

Radix-2 FFT: Decimation-in-time

Note: since $F_1(k)$ and $F_2(k)$ are $\frac{N}{2}$-DFTs:

$$F_1(k) = F_1(k + \frac{N}{2})$$
$$F_2(k) = F_2(k + \frac{N}{2})$$

we have,

$$X(k) = F_1(k) + W_N^k F_2(k)$$
$$X(k + \frac{N}{2}) = F_1(k + \frac{N}{2}) + W_N^{k+\frac{N}{2}} F_2(k + \frac{N}{2})$$
$$= F_1(k) - W_N^k F_2(k)$$

since $W_N^{k+\frac{N}{2}} = e^{-j\frac{2\pi}{N}(k+\frac{N}{2})} = e^{-j\frac{2\pi}{N}k} \cdot e^{-j\frac{2\pi}{N}\frac{N}{2}} = e^{-j\frac{2\pi}{N}k}(-1) = -W_N^k$

Radix-2 FFT: Decimation-in-time

Therefore,

$$X(k) = F_1(k) + W_N^k F_2(k) \quad k = 0, 1, \ldots, \frac{N}{2} - 1$$
$$X(k + \frac{N}{2}) = F_1(k) - W_N^k F_2(k) \quad k = 0, 1, \ldots, \frac{N}{2} - 1$$

Radix-2 FFT: Decimation-in-time

Repeating the decimation-in-time for $f_1(n)$ and $f_2(n)$, we obtain:

$$v_{11}(n) = f_1(2n) \quad n = 0, 1, \ldots, N/4 - 1$$
$$v_{12}(n) = f_1(2n+1) \quad n = 0, 1, \ldots, N/4 - 1$$
$$v_{21}(n) = f_2(2n) \quad n = 0, 1, \ldots, N/4 - 1$$
$$v_{22}(n) = f_2(2n+1) \quad n = 0, 1, \ldots, N/4 - 1$$

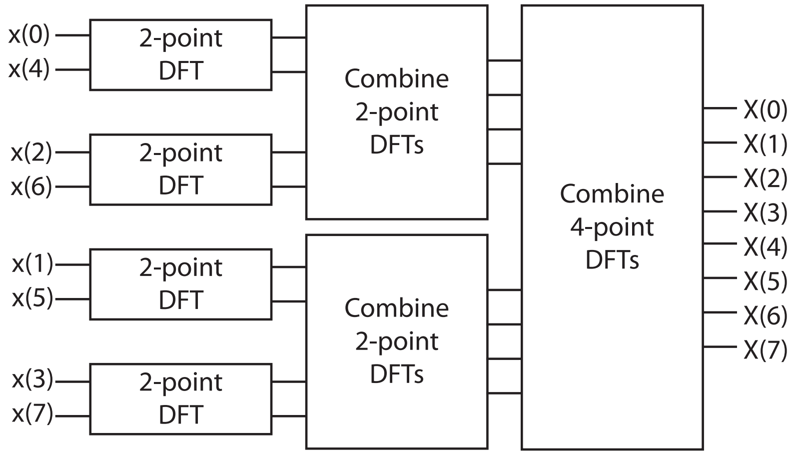
and

$$F_1(k) = V_{11}(k) + W_{N/2}^k V_{12}(k) \quad k = 0, 1, \ldots, N/4 - 1$$
$$F_1(k + N/4) = V_{11}(k) - W_{N/2}^k V_{12}(k) \quad k = 0, 1, \ldots, N/4 - 1$$
$$F_2(k) = V_{21}(k) + W_{N/2}^k V_{22}(k) \quad k = 0, 1, \ldots, N/4 - 1$$
$$F_2(k + N/4) = V_{21}(k) - W_{N/2}^k V_{22}(k) \quad k = 0, 1, \ldots, N/4 - 1$$

consisting of $N/4$-DFTs.

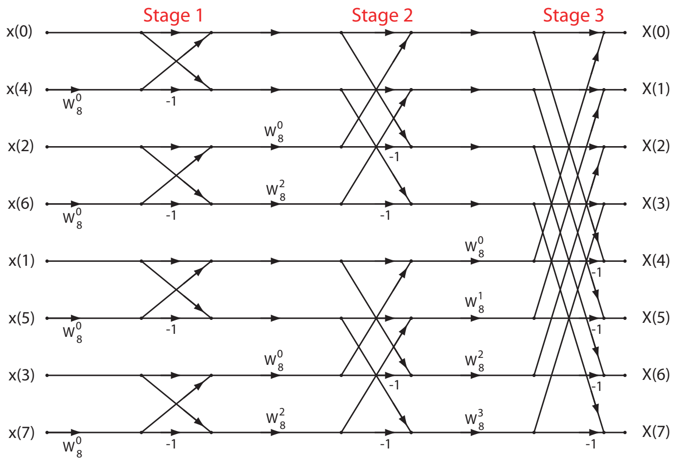
Radix-2 FFT: Decimation-in-time

For $N = 8$.

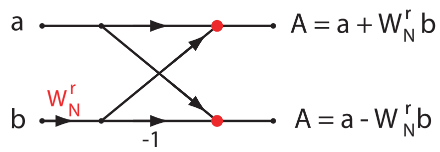


Radix-2 FFT: Decimation-in-time

For $N = 8$.



FFT Complexity



- Each butterfly requires:
  - one complex multiplication
  - two complex additions
- In total, there are:
  - $\frac{N}{2}$ butterflies per stage
  - $\log N$ stages
- Order of the overall DFT computation is: $O(N \log N)$

Convolution using FFT

To compute the convolution of $x(n)$ (support: $n = 0, 1, \ldots, L-1$) and $h(n)$ (support: $n = 0, 1, \ldots, M-1$):

- We know that the output $y(n) = x(n) * h(n)$ will be of length $M + L - 1$.
- We want to select the smallest $N = 2^r$ such that

$$N = 2^r \geq M + L - 1.$$

Convolution using FFT

To compute the convolution of $x(n)$ (support: $n = 0, 1, \ldots, L-1$) and $h(n)$ (support: $n = 0, 1, \ldots, M-1$):

1. Assign N to be the smallest power of 2 such that $N = 2^r \geq M + L - 1$.
2. Zero pad both $x(n)$ and $h(n)$ to have support $n = 0, 1, \ldots, N-1$.
3. Take the N-FFT of $x(n)$ to give $X(k)$, $k = 0, 1, \ldots, N-1$.
4. Take the N-FFT of $h(n)$ to give $H(k)$, $k = 0, 1, \ldots, N-1$.
5. Produce $Y(k) = X(k) \cdot H(k)$, $k = 0, 1, \ldots, N-1$.
6. Take the N-IFFT of $Y(k)$ to give $y(n)$, $n = 0, 1, \ldots, N-1$.

Convolution using FFT

To compute the convolution of $x(n)$ (support: $n = 0, 1, \ldots, L-1$) and $h(n)$ (support: $n = 0, 1, \ldots, M-1$):

1. Assign N to be the smallest power of 2 such that $N = 2^r \geq M + L - 1$.
2. Zero pad both $x(n)$ and $h(n)$ to have support $n = 0, 1, \ldots, N-1$.
   $O(1)$
3. Take the N-FFT of $x(n)$ to give $X(k)$, $k = 0, 1, \ldots, N-1$.
4. Take the N-FFT of $h(n)$ to give $H(k)$, $k = 0, 1, \ldots, N-1$.
   $O(N \log N)$
5. Produce $Y(k) = X(k) \cdot H(k)$, $k = 0, 1, \ldots, N-1$.
   $O(N)$
6. Take the N-IFFT of $Y(k)$ to give $y(n)$, $n = 0, 1, \ldots, N-1$.
   $O(N \log N)$

Complexity of Convolution using FFT

Therefore, the overall complexity of conducting convolution via the FFT is:

$$O(N \log N)$$

which is lower than $O(N^2)$ through naive direct computation of the DFT. ■