Introduction to Fast Fourier Transform (FFT) Algorithms

R.C. Maher

EE477 DSP
Spring 2007
Discrete Fourier Transform (DFT)

• The DFT provides uniformly spaced samples of the Discrete-Time Fourier Transform (DTFT)

• DFT definition:

\[ X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi nk}{N}} \]

\[ x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] e^{j\frac{2\pi nk}{N}} \]

• Requires \( N^2 \) complex multiplies and \( N(N-1) \) complex additions
Faster DFT computation?

• Take advantage of the symmetry and periodicity of the complex exponential:
  – symmetry: \( e^{-j2\pi k[N-n]/N} = e^{+j2\pi kn/N} = (e^{-j2\pi kn/N})^* \)
  – periodicity: \( e^{-j2\pi mk/N} = e^{-j2\pi [n+N]k/N} = e^{-j2\pi n[k+N]/N} \)

• Note that two length \( N/2 \) DFTs take less computation than one length \( N \) DFT: \( 2(N/2)^2 < N^2 \)

• Algorithms that exploit computational savings are collectively called *Fast Fourier Transforms*
Decimation-in-Time Algorithm

- Consider expressing DFT with even and odd input samples:

\[
X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N} = \sum_{\text{n even}} x[n] e^{-j2\pi nk/N} + \sum_{\text{n odd}} x[n] e^{-j2\pi nk/N}
\]

\[
= \sum_{r=0}^{N/2-1} x[2r] (e^{-j4\pi r/N})^k + e^{-j2\pi k/N} \sum_{r=0}^{N/2-1} x[2r+1] (e^{-j4\pi r/N})^k
\]

\[
= \sum_{r=0}^{N/2-1} x[2r] e^{-j2\pi r/(N/2)} + e^{-j2\pi k/N} \sum_{r=0}^{N/2-1} x[2r+1] e^{-j2\pi r/(N/2)}
\]
DIT Algorithm (cont.)

- Result is the sum of two N/2 length DFTs

\[ X[k] = \underbrace{G[k]}_{\text{N/2 DFT}} + e^{-j2\pi k / N} \cdot \underbrace{H[k]}_{\text{N/2 DFT}} \]

- Then repeat decomposition of N/2 to N/4 DFTs, etc.

\[ x[0,2,4,6], x[1,3,5,7] \rightarrow \text{N/2 DFT} \rightarrow X[0...7] \]

\[ e^{-j2\pi(0...7)/N} \]
Detail of “Butterfly”

- Cross feed of $G[k]$ and $H[k]$ in flow diagram is called a “butterfly”, due to shape

\[
e^{-j2\pi r/N} e^{-j2\pi(r+N/2)} = -e^{-j2\pi r}
\]

or simplify:
8-point DFT Diagram

Note that input is in *bit reversed* order.

Output is in *regular* order.

\[ x[0,4,2,6,1,5,3,7] \]

\[ X[0\ldots7] \]

\[ W_8^0 = 1; \quad W_8^1 = e^{-j\pi/4}; \quad W_8^2 = e^{-j\pi/2} = -j; \quad W_8^3 = e^{-j3\pi/4}; \quad W_8^4 = e^{-j\pi} = -1 \]
Computation on DSP

• Input and Output data
  – Real data in X memory
  – Imaginary data in Y memory
• Coefficients ("twiddle" factors)
  – $\cos(\text{real})$ values in X memory
  – $\sin(\text{imag})$ values in Y memory
• Inverse computed with exponent sign change and $1/N$ scaling