Fast Fourier Transform

Prof. Brian L. Evans
Dept. of Electrical and Computer Engineering
The University of Texas at Austin

Lecture 17

Discrete-Time Fourier Transform

- **Forward transform of discrete-time signal $x[n]$**

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\,\omega\, n}$$

  - Assumes that $x[n]$ is two-sided and infinite in duration
  - Produces $X(\omega)$ that is periodic in ω (in units of rad/sample) with period $2\,\pi$ due to exponential term

- **Inverse discrete-time Fourier transform**

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) e^{j\omega n} d\omega$$

- **Basic transform pairs**

$$x[n] = \delta[n] \Leftrightarrow X(\omega) = 1$$

$$x[n] = 1 \Leftrightarrow X(\omega) = \sum_{k=-\infty}^{\infty} \delta(\omega - 2\pi k)$$

Discrete Fourier Transform (DFT)

- **Discrete Fourier transform (DFT) of a discrete-time signal $x[n]$ with finite extent $n \in [0, N\text{-}1]$**

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}nk} = X(\omega)\Big|_{\omega=\frac{2\pi}{N}k} \qquad \text{for } k = 0, 1, \ldots, N\text{-}1$$

$X[k]$ periodic with period N due to exponential

Also assumes $x[n]$ periodic with period N

Two-Point DFT

$$X[0] = x[0] + x[1]$$

$$X[1] = x[0] - x[1]$$

- **Inverse discrete Fourier transform**

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

- **Twiddle factor** $W_N = e^{j\frac{2\pi}{N}} \Rightarrow x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\, W_N^{nk}$

Discrete Fourier Transform (con't)

- **Forward transform**

  $$X[k] = \sum_{n=0}^{N-1} x[n]\, W_N^{-nk}$$

  for $k = 0, 1, \ldots, N\text{-}1$

  Exponent of W_N has period N

- **Memory usage**

  $x[n]$: N complex words of RAM

  $X[k]$: N complex words of RAM

  W_N: N complex words of ROM

- **Halve memory usage**

  Allow output array $X[k]$ to write over input array $x[n]$

  Exploit twiddle factors symmetry

- **Computation**

  N^2 complex multiplications

  $N\,(N-1)$ complex additions

  N^2 integer multiplications

  N^2 modulo indexes into lookup table of twiddle factors

- **Inverse transform**

  $$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\, W_N^{nk}$$

  for $n = 0, 1, \ldots, N\text{-}1$

  Memory usage?

  Computational complexity?

Fast Fourier Transform Algorithms

- **Communication system application: multicarrier modulation using harmonically related carriers**
  - Discrete multitone modulation in ADSL & VDSL modems
  - OFDM transceivers such as in IEEE 802.11a wireless LANs
- **Efficient divide-and-conquer algorithm**
  - Compute discrete Fourier transform of length $N = 2^{\nu}$
  - $\frac{1}{2} N \log_2 N$ complex multiplications and additions
  - How many real complex multiplications and additions?
- **Derivation: Assume N is even and power of two**

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} = \sum_{n=even}^{N-1} x[n] W_N^{nk} + \sum_{n=odd}^{N-1} x[n] W_N^{nk}$$

Fast Fourier Transform (cont'd)

- **Substitute $n = 2r$ for n even and $n = 2r+1$ for odd**

$$X[k] = \sum_{r=0}^{N/2-1} x[2r]\, W_N^{2rk} + \sum_{r=0}^{N/2-1} x[2r+1]\, W_N^{(2r+1)k}$$

$$= \sum_{r=0}^{N/2-1} x[2r] \left(W_N^2\right)^{rk} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1] \left(W_N^2\right)^{rk}$$

- **Using the property** $W_N^{2l} = e^{-j 2\pi \frac{2l}{N}} = e^{-j \frac{2\pi l}{N/2}} = W_{N/2}^{l}$

$$X[k] = \sum_{r=0}^{N/2-1} x[2r]\, W_{N/2}^{rk} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1]\, W_{N/2}^{rk} = G[k] + W_N^k\, H[k]$$

One FFT length N => two FFTs length $N/2$

Repeat process until two-point FFTs remain

Computational complexity of two-point FFT?

Two-Point FFT

$$X[0] = x[0] + x[1]$$

$$X[1] = x[0] - x[1]$$

Linear Convolution by FFT

- **Linear convolution**

  $x[n]$ has length N_x and $h[n]$ has length N_h

  $y[n]$ has length N_x+N_h-1

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

- **Linear convolution requires $N_x N_h$ *real-valued* multiplications and $2N_x + 2N_h - 1$ words of memory**
- **Linear convolution by FFT of length $N = N_x+N_h - 1$**

  Zero pad $x[n]$ and $h[n]$ to make each N samples long

  Compute forward DFTs of length N to obtain $X[k]$ and $H[k]$

  $Y[k] = H[k]\, X[k]$ for $k = 0 \ldots N-1$: may overwrite $X[k]$ with $Y[k]$

  Take inverse DFT of length N of $Y[k]$ to obtain $y[n]$
- **If $h[n]$ is fixed, then precompute and store $H[k]$**

Linear Convolution by FFT

- **Implementation complexity using *N*-length FFTs**

  $3 N \log_2 N$ complex multiplications and additions

  $2 N$ complex words of memory if $Y[k]$ overwrites $X[k]$

- **FFT approach requires fewer computations if**

  $$12(N_x + N_h - 1)\log_2(N_x + N_h - 1) < N_x N_h$$

- **Disadvantages of FFT approach**
  - Uses twice the memory: $2(N_x + N_h - 1)$ complex words vs. $2N_x + 2N_h - 1$ words
  - Often requires floating-point arithmetic
  - Adds delay of N_x samples to buffer $x[n]$ whereas linear convolution is pointwise
  - Creates discontinuities at boundaries of blocks of input data, which can be overcome by overlapping blocks

FFT under fixed-point arithmetic?