Administrative Stuff

- Matlab4 is posted – due **Thursday** Nov 2
- Exam1 handed back after class

Tufts

EE-125:
Digital Signal Processing

FFT and Related Algorithms
(P&M Chapter 8, in 1 lecture!)

Professor Tracey

Tufts

Outline

- Writing the DFT as a matrix
- Big picture / motivation for FFT
- Reviewing the FFT video: decimation in time
  - Van Veen covers the 'decimation in time' algorithm – splitting data x(n) into even and odd samples n
- If time permits: 'decimation in frequency' algorithm – split X(k) into even and odd frequencies n
- Some other transforms
  - You'll use DCT in the next Matlab project

Big picture with FFT -1

- We saw earlier that the DFT can be written in matrix form:

$$X = \mathbf{W}\, x$$

where X and x are N-point vectors (in freq and time, respectively, and **W** is a NxN matrix with exponential terms

- Computing the DFT using the matrix approach takes O(N^2) calculations
- Same is true for inverse DFT (IDFT)

$$x = \mathbf{W}^{-1} X$$

Big picture with FFT - 2

- FFT exploits symmetries in **W** to speed things up
  - Break original problem into even and odd: 2 problems, each half the length
  - Keep breaking down until we get to many 2x2 systems
  - Most **W** factors for that system are very simplified; many terms come down to exp(0) (i.e., 1) or exp(pi) (i.e. -1) so we can add and subtract instead of doing multiplications
- Result: calculation is done in O(N log2 N) calculations - but assumes we start with data vector with length power-of-2 (or, zero-pad to a power of 2)
- The above splitting by 2 is called a 'radix 2' FFT; other versions are available (radix 4, etc) but less common

FFT exploits symmetries of $e^{-j\frac{2\pi}{N}kn}$

Define $\boxed{W_N = e^{-j2\pi/N}}$

1) Complex conjugate symmetry $W_N^{k(N-n)} = W_N^{-kn} = (W_N^{kn})^*$

2) Periodicity in n, k $W_N^{kn} = W_N^{k(N+n)} = W_N^{(k+N)n}$

Decimation in Time FFT (one of many)

- build a big DFT from smaller ones
- Assume $N = 2^m$

separate $x[n]$ into even and odd-indexed subsequences

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

The Fast Fourier Transform Algorithm

3

even indices: $n = 2r$

odd indices: $n = 2r+1$, $\quad r = 0, 1, \dots, \frac{N}{2} - 1$

$$X[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r]\, W_N^{k2r} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1]\, W_N^{(2r+1)k}$$

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

But $W_N^2 = e^{-j\frac{2\pi}{N}2} = e^{-j2\pi/(N/2)} = W_{N/2}$

$$X[k] = \underbrace{\sum_{r=0}^{N/2-1} x[2r]\, W_{N/2}^{kr}}_{\substack{N/2 \text{ DFT of even samples} \\ X_e[k]}} + W_N^k \underbrace{\sum_{r=0}^{N/2-1} x[2r+1]\, W_{N/2}^{kr}}_{\substack{N/2 \text{ DFT of odd samples} \\ X_o[k]}}$$

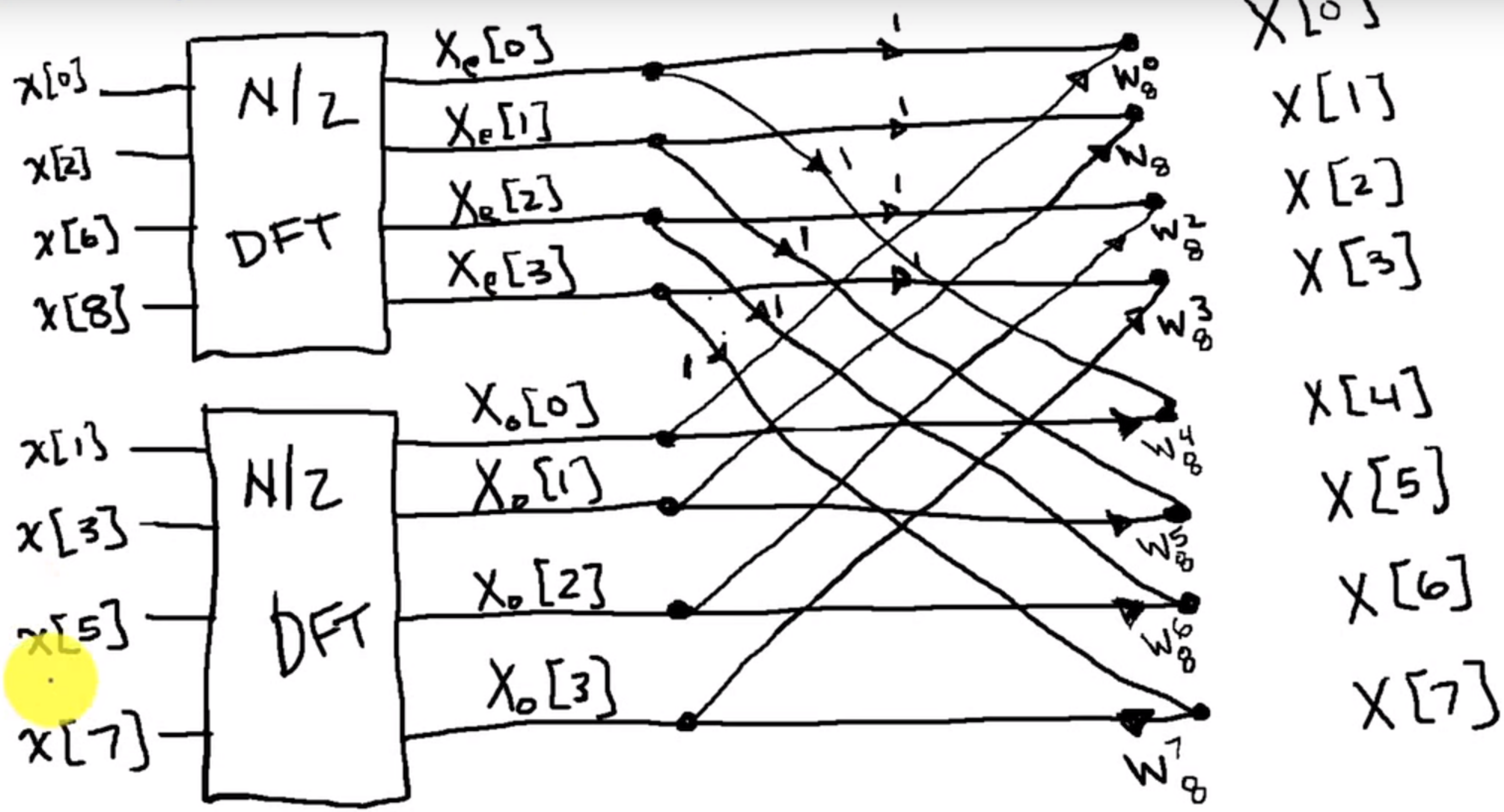
$X[k] = X_e[k] + W_N^k X_o[k]$ — sum of 2 $N/2$ point DFTs

9:27 / 18:54

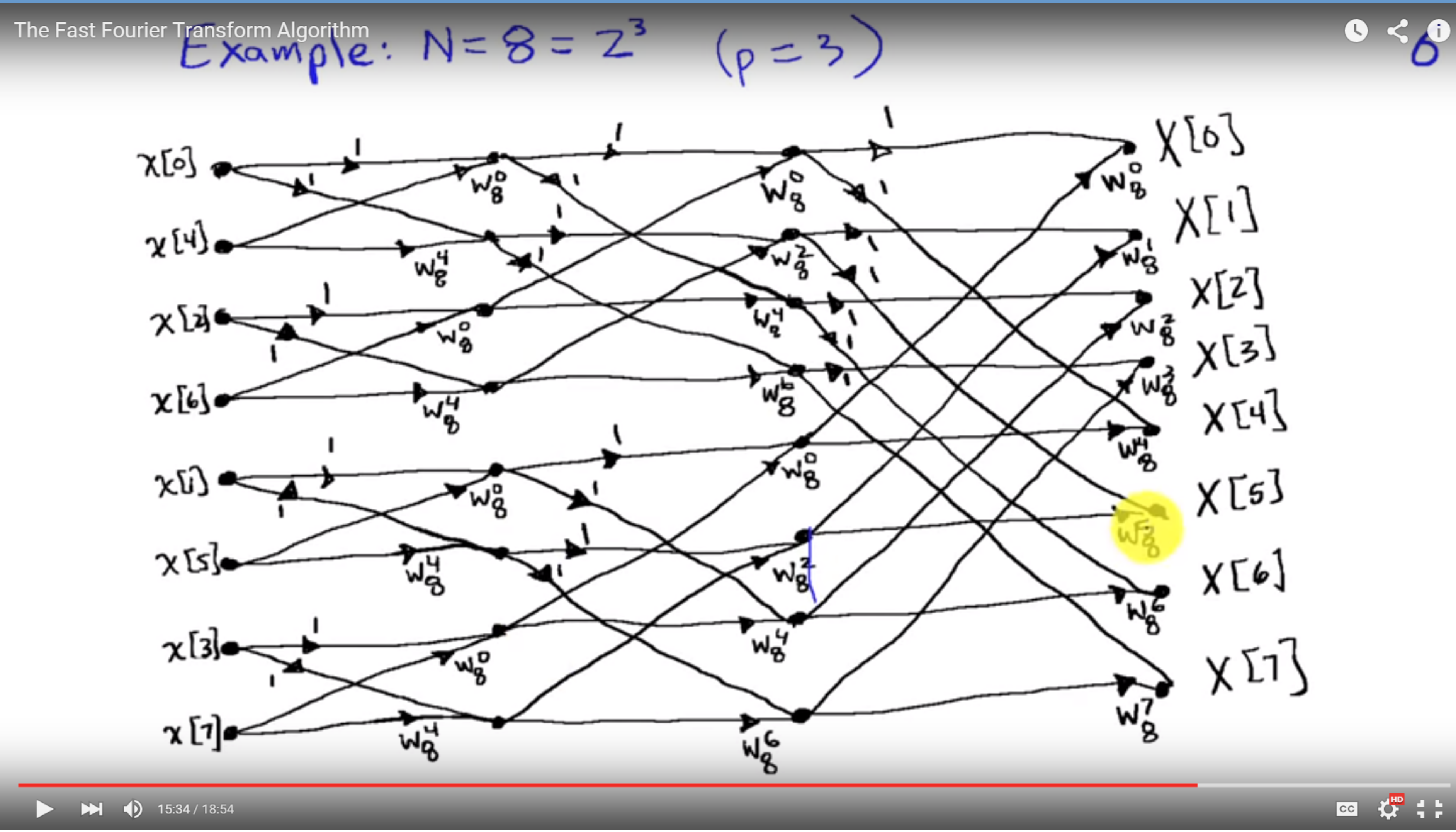
Tufts

The Fast Fourier Transform Algorithm

Example: $N = 8$



10:39 / 18:54



The Fast Fourier Transform Algorithm
Example: N=8=2^3 (p=3)
6
x[0]
x[4]
x[2]
x[6]
x[1]
x[5]
x[3]
x[7]
X[0]
X[1]
X[2]
X[3]
X[4]
X[5]
X[6]
X[7]
15:34 / 18:54

Tufts

Related transforms - Discrete Cosine Transform

- Uses Fourier transform property: if x(n) is real and even, X(k) is real and even.
  - For real and even x(n), exponential in DFT can be replaced by a cosine, since sine only contributes to imag(X(k)).
- To use the DCT, we copy (*extend*) non-symmetric signals to make them symmetric: see below.

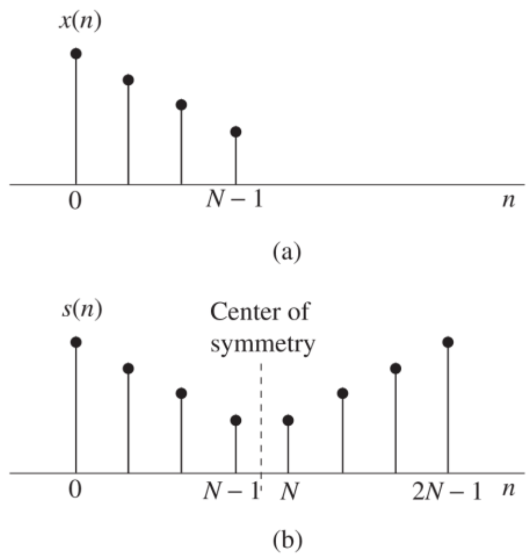


Figure 7.5.1 Original sequence $x(n)$, $0 \le n \le N-1$ and its $2N$-point even extension $s(n)$, $0 \le n \le 2N-1$.

Why use the DCT?

- Compared to the DFT, the DCT gives a less compact representation of sinusoids (Figure 7.5.2, book), but more compact representation of other signals . Here, ‘compact’ means ‘fewer coefficients with significant amplitude’
- DCT turns out to be good for many images
- DCT is the basis of the JPEG image compression algorithm

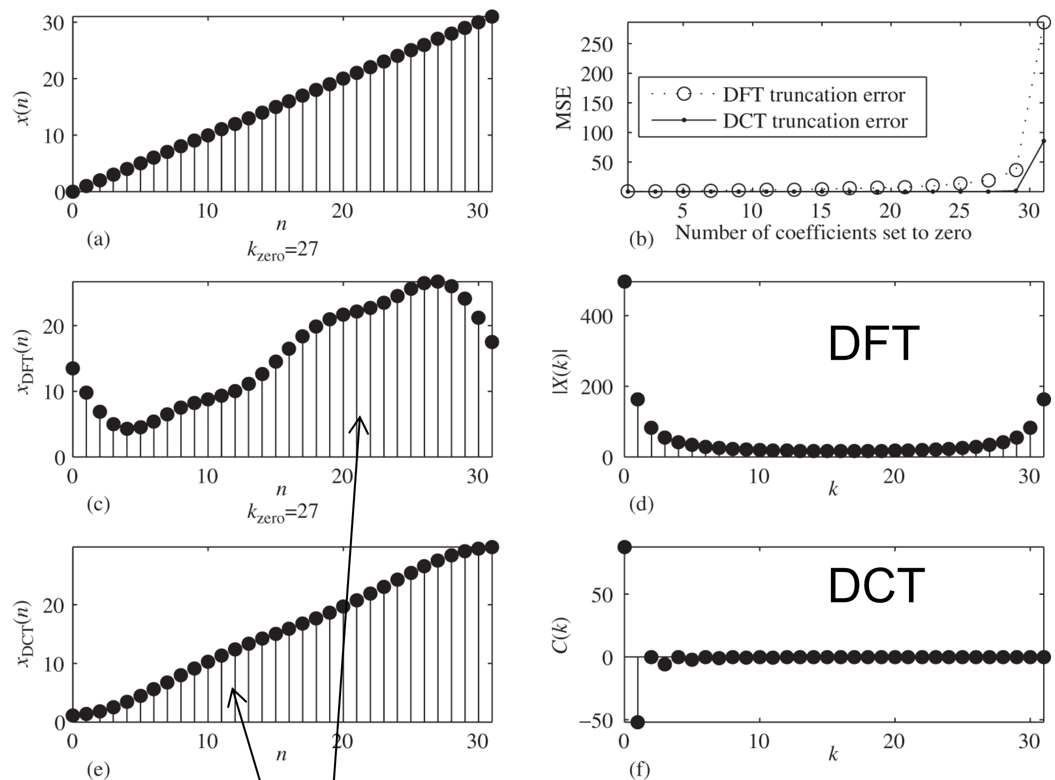


Figure 7.5.3 A discrete time sinusoidal signal and its DFT and DCT representations.

Reconstructions using 3 coefficients

Other algorithms to know about

- Goertzel (P&M 8.3)
  - Rewrite the DFT so we have a parallel bank of filters, each one of which gives the output for a single frequency in the DFT
  - Advantage is that we don't need to implement every frequency; so can be faster than FFT if we just need answers at a few frequencies
  - Classic use: processing of dial tones
- Chirp z-transform (P&M 8.3)
  - Lets us evaluate the transform at points other than the unit circle
  - Used in speech analysis (on-line, see "The Chirp z-Transform Algorithm—A Lesson in Serendipity")