Georgia Institute of Technology

CSIP Center for Signal & Image Processing

ECE4270 Fundamentals of DSP

Lecture 26

The Discrete Fourier Transform (DFT) and FFT

School of ECE
Center for Signal and Information Processing
Georgia Institute of Technology

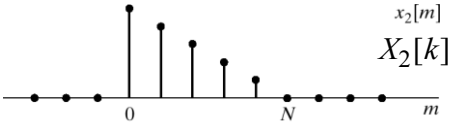
Overview of Lecture

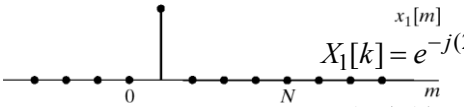
- Circular Convolution via DFT
- Linear Convolution via DFT
- Block Convolution
  -Overlap add (OLA)
  -Overlap save (OLS)
- DCT (Discrete-Cosine Transform)– Moved to last lecture
- The FFT (Chapter 9)
  – Decimation in time

DFT Theorems and Properties

	Finite-Length Sequence (Length N)	N-point DFT (Length N)
1.	$x[n]$	$X[k]$
2.	$x_1[n]$, $x_2[n]$	$X_1[k]$, $X_2[k]$
3.	$ax_1[n] + bx_2[n]$	$aX_1[k] + bX_2[k]$
4.	$X[n]$	$Nx[((-k))_N]$
5.	$x[((n-m))_N]$	$W_N^{km}X[k]$
6.	$W_N^{-\ell n}x[n]$	$X[((k-\ell))_N]$
7.	$\sum_{m=0}^{N-1} x_1(m)x_2[((n-m))_N]$	$X_1[k]X_2[k]$
8.	$x_1[n]x_2[n]$	$\frac{1}{N}\sum_{\ell=0}^{N-1} X_1(\ell)X_2[((k-\ell))_N]$

Circular Convolution - I

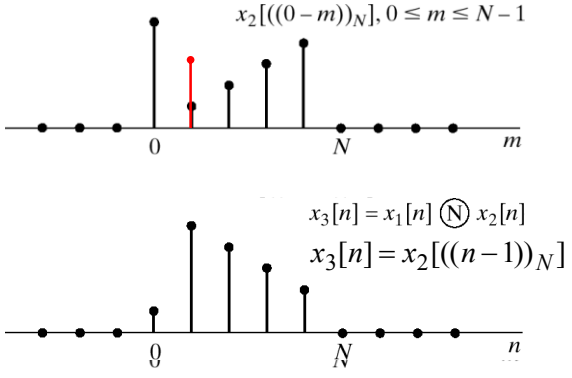




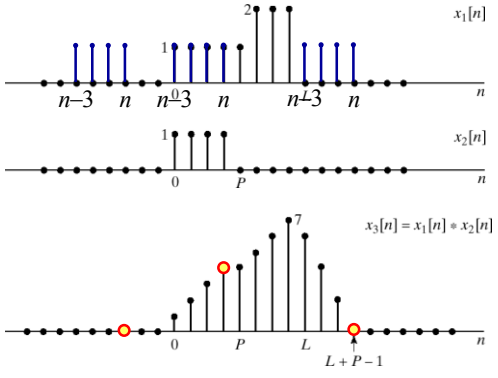
$$X_3[k] = X_1[k]X_2[k] = e^{-j(2\pi/N)k}X_2[k]$$

$$x_3[n] = \sum_{m=0}^{N-1} \delta[m-1]x_2[((n-m))_N] = x_2[((n-1))_N]$$

Circular Flipping and Shifting



"Linear Convolution"



Analysis for Linear Convolution

$$x_3[n] = \sum_{m=-\infty}^{\infty} x_1[m]x_2[n-m] \Leftrightarrow X_3(e^{j\omega}) = X_1(e^{j\omega})X_2(e^{j\omega})$$

$$X_3(e^{j(2\pi/N)k}) = X_1(e^{j(2\pi/N)k})X_2(e^{j(2\pi/N)k})$$

$$\Rightarrow X_3[k] = X_1[k]X_2[k]$$

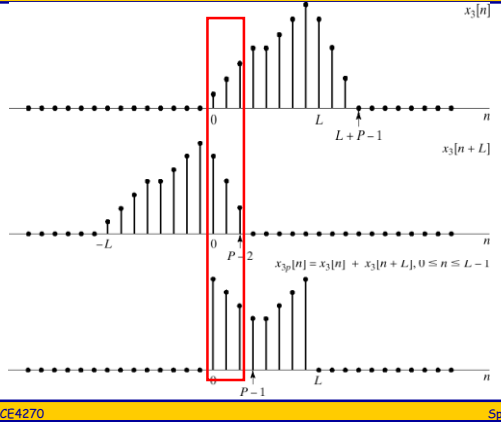
$$x_{3p}[n] = \frac{1}{N}\sum_{k=0}^{N-1} X_1[k]X_2[k]e^{j(2\pi/N)kn}$$

$$x_{3p}[n] = \sum_{m=0}^{N-1} x_1[m]x_2[((n-m))_N]$$

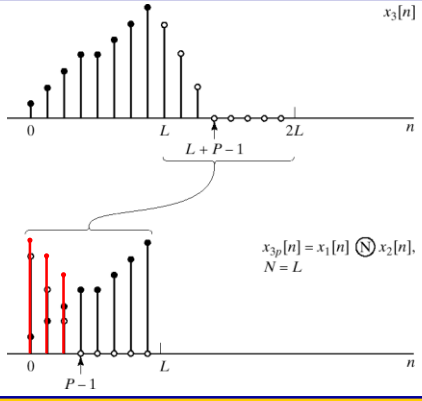
$$x_{3p}[n] = \sum_{r=-\infty}^{\infty} x_3[n-rN] \quad n = 0,1,\ldots,N-1$$

To avoid overlap, pick $N > L+P-2$

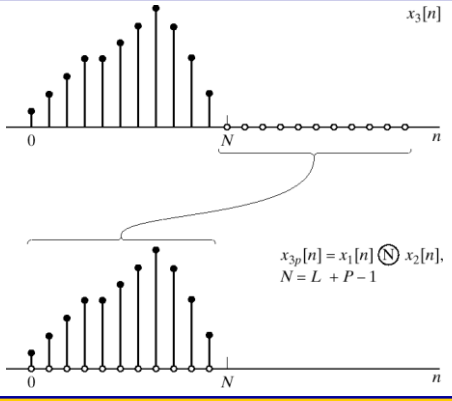
Aliased Convolution - I



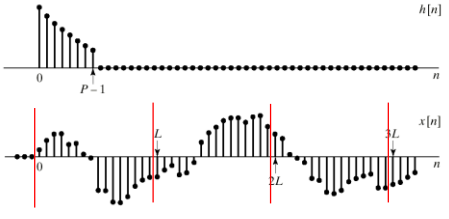
Aliased Convolution - II



Aliased Convolution - III



Block Convolution - Overlap Add Method (OLA)

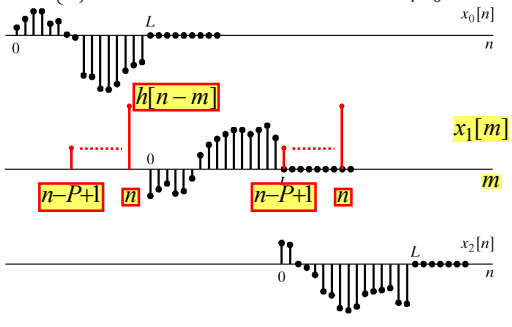


$$x[n] = \sum_{r=0}^{\infty} x_r[n-rL] \rightarrow y[n] = x[n] * h[n] = \sum_{r=0}^{\infty} y_r[n-rL]$$

$$x_r[n] = \begin{cases} x[n+rL], & 0 \le n \le L-1 \\ 0, & \text{otherwise} \end{cases} \rightarrow y_r[n] = x_r[n] * h[n]$$

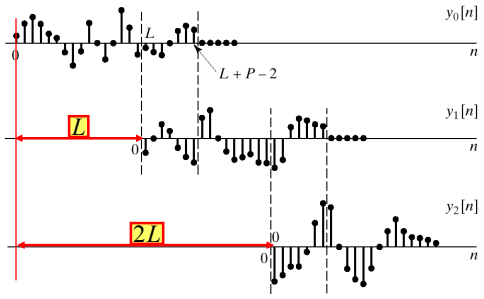
OLA: Segmenting the Input

$$x_r[n] = \begin{cases} x[n+rL], & 0 \le n \le L-1 \\ 0, & \text{otherwise} \end{cases} \qquad x[n] = \sum_{r=0}^{\infty} x_r[n-rL]$$

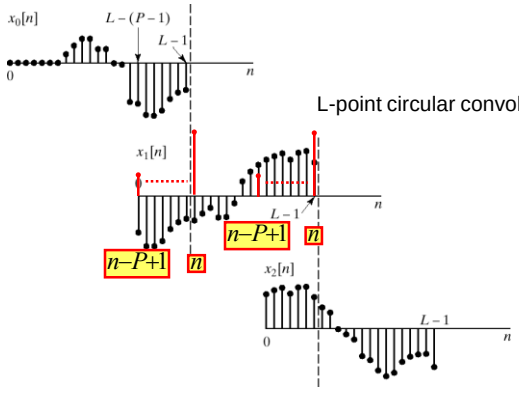


OLA: Putting the Output Pieces Together

$$y[n] = \sum_{r=0}^{\infty} y_r[n - rL] \qquad y_r[n] = x_r[n] * h[n]$$

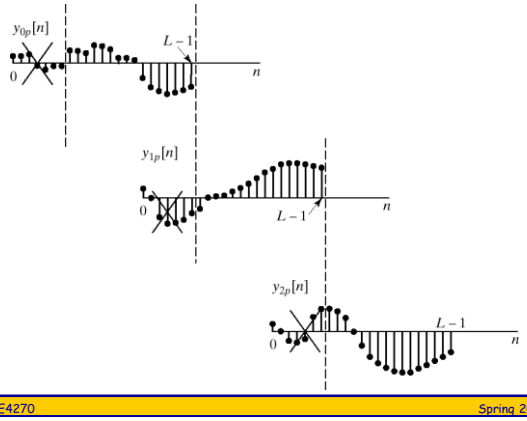


OLS Method - Segmenting the Input



L-point circular convolution

OLS Method - Extracting the Output



Computation of the DFT

- In order for the DFT to be useful for linear filtering, spectrum analysis, etc., we need efficient computation algorithms for

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

- Using the above directly requires *N* complex multiplications and *N-1* complex additions for each of the *N* DFT values $\Rightarrow \mu(N) = N^2$ complex multiplications.

  For example,

$$N = 1024 \Rightarrow \mu(N) \approx 10^6$$

Cooley and Tukey, 1965

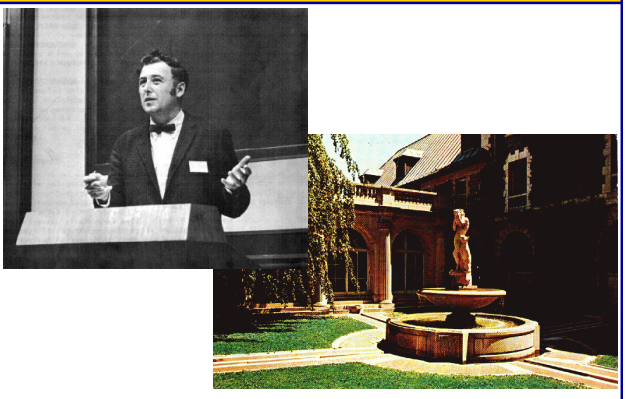
An Algorithm for the Machine Calculation of Complex Fourier Series

By James W. Cooley and John W. Tukey

An efficient method for the calculation of the interactions of a 2^m factorial experiment was introduced by Yates and is widely known by his name. The generalization to 3^m was given by Box et al. [1]. Good [2] generalized these methods and gave elegant algorithms for which one class of applications is the calculation of Fourier series. In their full generality, Good's methods are applicable to certain problems in which one must multiply an N-vector by an $N \times N$ matrix which can be factored into m sparse matrices, where m is proportional to $\log N$. This results in a procedure requiring a number of operations proportional to $N \log N$ rather than N^2. These methods are applied here to the calculation of complex Fourier series. They are useful in situations where the number of data points is, or can be chosen to be, a highly composite number. The algorithm is here derived and presented in a rather different form. Attention is given to the choice of N. It is also shown how special advantage can be obtained in the use of a binary computer with $N = 2^m$ and how the entire calculation can be performed within the array of N data storage locations used for the given Fourier coefficients.

J. W. Cooley and J. W. Tukey, Mathematics of Computation, Vol. 19, pp. 297-301, 1965.

Jim Cooley at Arden House, 1968



Decimation-in-Time Derivation - I

- We want to compute *X[k]* efficiently. Divide *x[n]* into even- and odd-indexed

$$X[k] = \sum_{n \text{ even}} x[n] W_N^{nk} + \sum_{n \text{ odd}} x[n] W_N^{nk}$$

- Substituting *n*=2*r* and *n*=2*r*+1 into above gives (assume N is even)

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

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

Decimation-in-Time Derivation - II

- Using the fact

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

it follows that

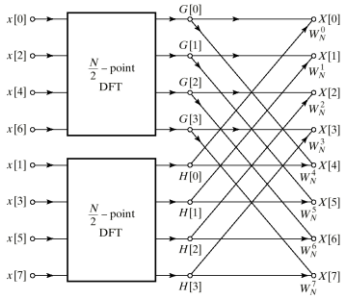
$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1] W_{N/2}^{rk}$$

$$= G[k] + W_N^k H[k], \qquad k = 0, 1, \ldots, N-1.$$

In other words, $G[k]$ and $H[k]$ are $N/2$-point DFTs.

$\Rightarrow \mu(N) = N + 2\mu(N/2)$

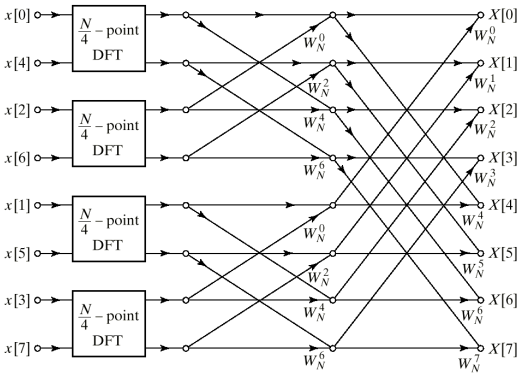
Decimation in Time (First Stage)



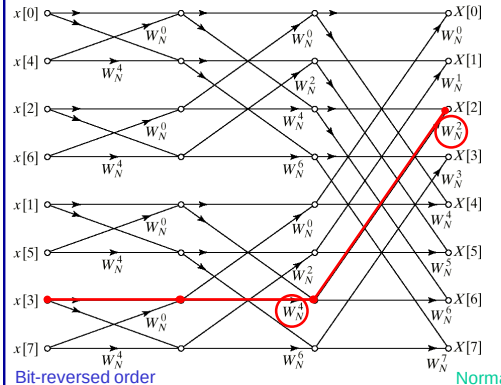
$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r]W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1]W_{N/2}^{rk}$$

$$= G[k] + W_N^k H[k], \qquad k = 0, 1, \ldots, N-1.$$

Decimation in Time (Second Stage)



Decimation in Time (Third Stage)



$= \ldots$

$+x[3]W_N^{3\cdot 2}$

$+\ldots$

Bit-reversed order

Normal order