Administrative

- Matlab3 due ***<u>tomorrow</u>*** at midnight
- I'll return Exam 1 next Monday; will email when grades are on Trunk
  - Will also email when Matlab/Python project grades posted
- **Assignment for Monday: watch video on FFT. I'll email the link**
- HW due next Monday: Problems 7.8, 7.9

EE-125:
Digital Signal Processing

DFT and Circular Convolution

Professor Tracey

Tufts

Outline

- DFT brief review; time reversal, time shifting
- DFT properties (7.2)
  - Effects of time reversal, shifting in frequency domain
  - Symmetry properties
  - Circular convolution!!
- When does circular convolution = linear convolution? (P&M 7.3)
- DFT-based filtering of long sequences (P&M 7.3)
  - Overlap-add vs. overlap-sum

Idea: Sampling in freq

- Sampling in frequency has the same structure:
  - Multiply $X(\omega)$ (or $X(f)$) by an impulse train in frequency, spacing $\delta\omega$, giving N points over one period.
  - Multiplication in ω $\leftarrow\rightarrow$ convolution in time.
  - Thus, sampling in ω corresponds to analyzing a periodic signal xp(n)
  - By sampling more finely (bigger N) we avoid time-domain aliasing
  - We can recover x(n) from xp(n) by just taking the first N points

Circular shift of sequence

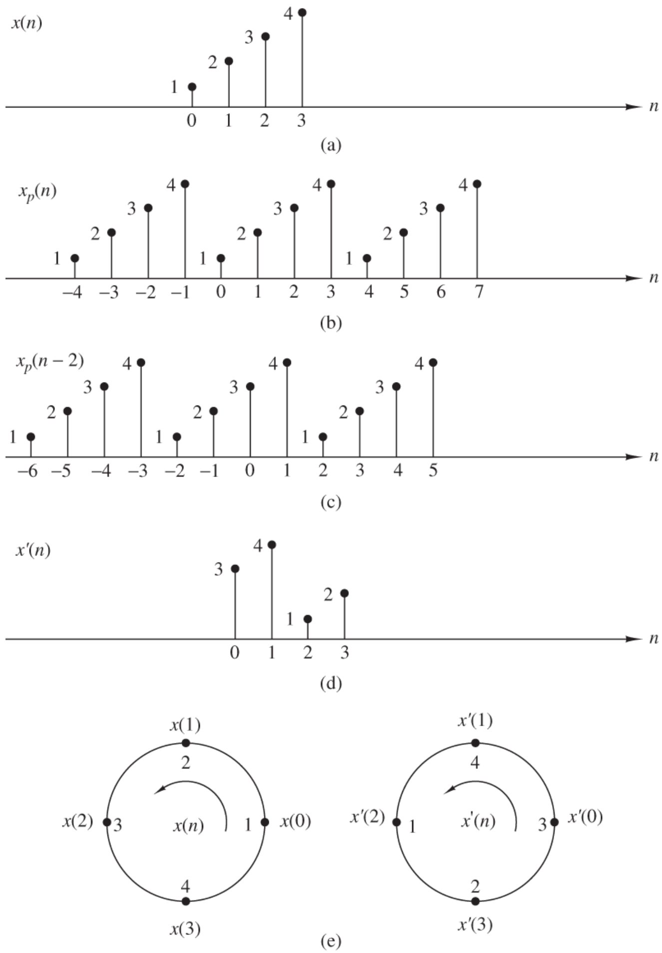

Figure 7.2.1 Circular shift of a sequence.

If circular shift by k, and sequence length is N, then

$x'(n) = x((n-k))_N$

Time reversal of a sequence

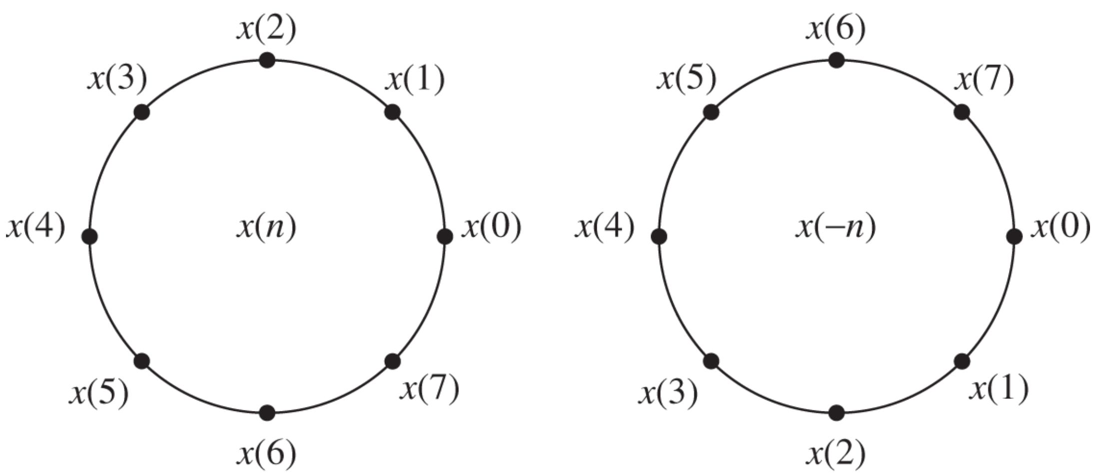

Figure 7.2.3 Time reversal of a sequence.

If time-reverse, and sequence length is N, then

$x'(n) = x((-n))_N$

$= x(N-n)$ for

$0 <= n <= (N-1)$

Why bother with all this?

- To do FIR filtering, we want to implement *linear* convolution ($y = h*x$)
- For the DFT, convolution becomes ‘circular’
- In this lecture, we’ll see how to set up circular convolution so the results = linear convolution
- Next lecture, we’ll see that the FFT gives a very fast way to calculate the DFT
- Then, high-speed filtering can be done:
  - Take the FFT’s of $h(n)$ and $x(n)$
  - Multiply their FFT’s to give $Y(k) = H(k)\,X(k)$
  - Take the inverse IFFT to find $y(n)$
- Question: why is this just for FIR filtering?

Video resources

Some notation differences: the videos follow the notation used in Oppenheim & Schafer, the other widely used DSP textbook. Two differences you will see:

a) Instead of x(n) and h(n), square brackets are used: h[n] and x[n]

b) Instead of writing DTFT transforms as X(ω) or H(ω), they are written as X(ejω) or H(ejω).

This is actually more accurate (though extra writing), since H(ω) really means "H(z) evaluated at z = ejω "

VIDEOS

1) DFT properties: https://youtu.be/oD1po01Ev0g

2) DFT circular convolution property: https://youtu.be/eOvMUtMoQG8, also https://youtu.be/O2txxBHeXsY

3) This video covers how we can use the DFT and circular convolution to filter very long sequences of data – the overlap / add method.

https://youtu.be/W31IGel1hxg

Circular convolution derivation (P&M, 7.2.2)

- What is the inverse DFT of X3(k) = X1(k) X2(k)? By definition

$$x_3(m) = \frac{1}{N}\sum_{k=0}^{N-1} X_3(k)e^{j2\pi km/N}$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} X_1(k)X_2(k)e^{j2\pi km/N}$$

- Plugging in expressions for X1(k) and X2(k) and rearranging,

$$x_3(m) = \frac{1}{N}\sum_{k=0}^{N-1}\left[\sum_{n=0}^{N-1} x_1(n)e^{-j2\pi kn/N}\right]\left[\sum_{l=0}^{N-1} x_2(l)e^{-j2\pi kl/N}\right]e^{j2\pi km/N}$$

$$= \frac{1}{N}\sum_{n=0}^{N-1} x_1(n)\sum_{l=0}^{N-1} x_2(l)\left[\sum_{k=0}^{N-1} e^{j2\pi k(m-n-l)/N}\right]$$

Circular convolution derivation (P&M, 7.2.2)

- What is the inverse DFT of X3(k) = X1(k) X2(k)? By definition

$$x_3(m) = \frac{1}{N} \sum_{k=0}^{N-1} X_3(k) e^{j2\pi km/N}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} X_1(k) X_2(k) e^{j2\pi km/N}$$

- Plugging in expressions for X1(k) and X2(k) and rearranging,

$$x_3(m) = \frac{1}{N} \sum_{k=0}^{N-1} \left[\sum_{n=0}^{N-1} x_1(n) e^{-j2\pi kn/N} \right] \left[\sum_{l=0}^{N-1} x_2(l) e^{-j2\pi kl/N} \right] e^{j2\pi km/N}$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} x_1(n) \sum_{l=0}^{N-1} x_2(l) \left[\sum_{k=0}^{N-1} e^{j2\pi k(m-n-l)/N} \right]$$

Exponential acts as a delta function:

Circular convolution derivation continued

- This last term is the interesting one. If we define

  a = exp(j2π (m-n-l)/N),

  it turns out that

$$\sum_{k=0}^{N-1} a^k = \begin{cases} N, & l = m - n + pN = ((m-n))_N, \quad p \text{ an integer} \\ 0, & \text{otherwise} \end{cases}$$

i.e., a=1 and we get non-zero output when m-n-l either equals 0, or equals an integer multiple of N

- This gives something that looks like a convolution, but has a circular periodicity, so we call it <u>circular convolution</u>

$$x_3(m) = \sum_{n=0}^{N-1} x_1(n) x_2((m-n))_N$$

$$m = 0, 1, \ldots .N-1$$

Calculating the N-point circular convolution of x1 and x2

<u>Time domain</u>

a) Draw x1(m) on an N-point circle (time goes CCW)

b) Draw $x2((-m))_N$ on an N-point circle

c) For n=0,1,…, N-1

   a) Rotate x2 by one point counter clockwise (CCW)
   b) Multiply
   c) Add

<u>Frequency domain</u>

a) Find X1(k) and X2(k) for k=0,1,….N-1

b) Find X3(k) = X1(k) X2(k) for all k

c) x3(n) = IDFT(X3(k)), evaluated for n=0,1,…, N-1

Circular convolution on-line demos

- A helpful demo
  - http://doctord.dyndns.org/Courses/BEI/EE301/EE235/demos/cir_conv.htm

- Convolution in architectural acoustics:
  - https://www.acentech.com/services/3dlistening-acoustics-simulation

Outline

- DFT brief review; time reversal, time shifting
- DFT properties (7.2)
  - Effects of time reversal, shifting in frequency domain
  - Symmetry properties
  - Circular convolution!!
- When does circular convolution = linear convolution? (P&M 7.3)
- DFT-based filtering of long sequences (P&M 7.3)
  - Overlap-add vs. overlap-sum
  - You'll implement these in MATLAB3

Long sequence filtering – overlap /add

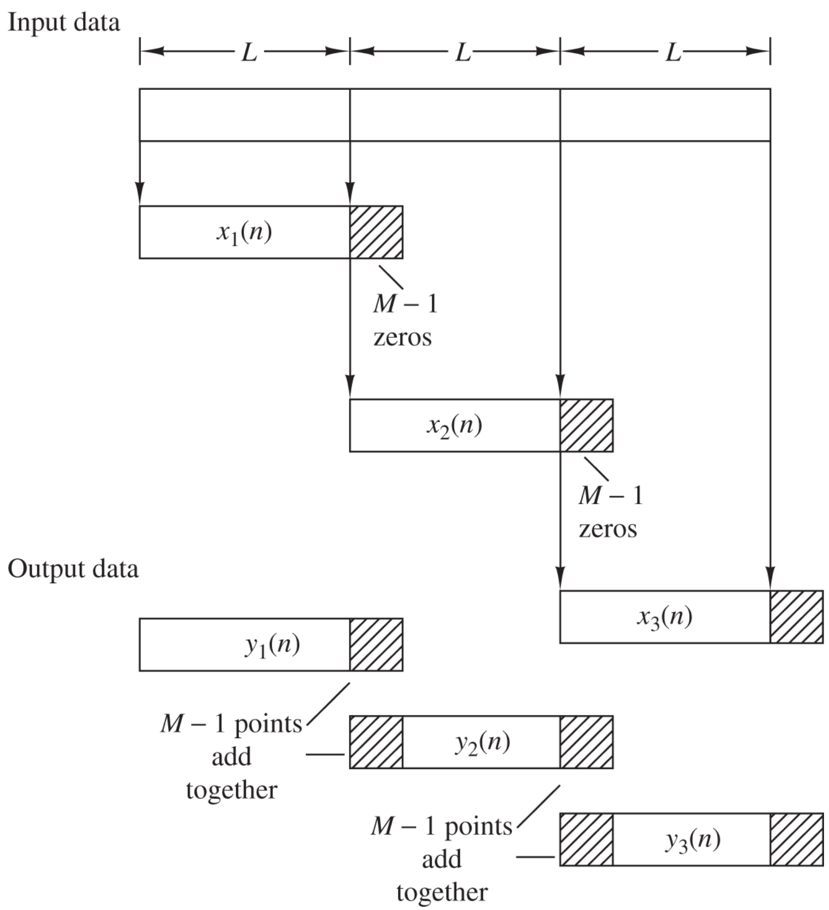

Figure 7.3.2 Linear FIR filtering by the overlap-add method.

We use N=L+M-1 point FFT, with L points of data

All points are unaliased, but they only response to first L points (as L+1 to N are all zero-padded).

We resolve this by starting the next block after the first L points, and adding outputs.

Long sequence filtering – overlap/save

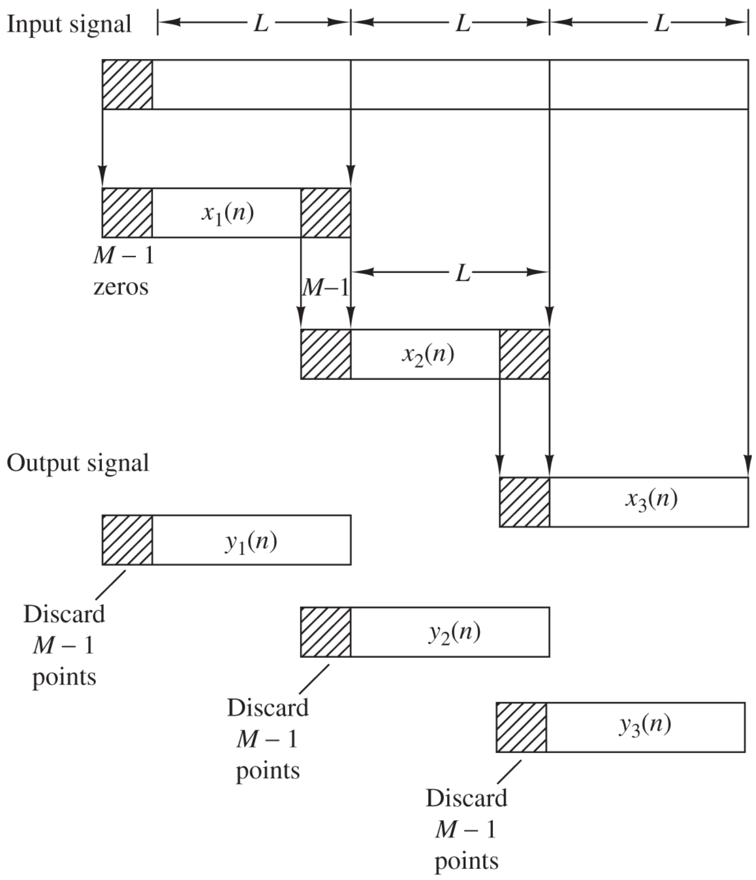

Figure 7.3.1 Linear FIR filtering by the overlap-save method.

We use N=L+M-1 point FFT, with M-1 repeated data points and L new data points

First M-1 points have time-domain aliasing, so they are discarded; the rest are OK.

Slightly faster than overlap/add as no additions needed