ESE 531: Digital Signal Processing

Week 12

Lecture 22: Apr 4, 2021

Discrete Fourier Transform, Pt 2

Penn ESE 531 Spring 2021 – Khanna
Adapted from M. Lustig, EECS Berkeley

Today

- Review:
  - Discrete Fourier Transform (DFT)
  - Circular Convolution
- Fast Convolution Methods
- Discrete Cosine Transform

Discrete Fourier Transform

- The DFT

$$W_N \triangleq e^{-j2\pi/N}$$

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

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

- It is understood that,

$$x[n] = 0 \quad \text{outside } 0 \le n \le N-1$$

$$X[k] = 0 \quad \text{outside } 0 \le k \le N-1$$

DTFT Vs. DFT

DTFT:

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

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

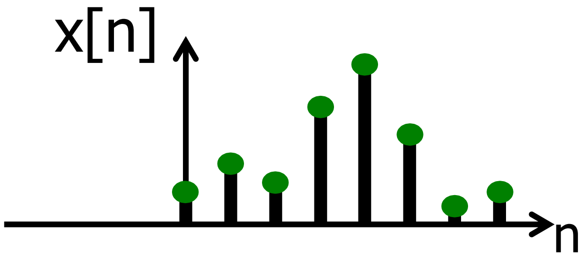
DFT:

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

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

DFT Intuition

Time

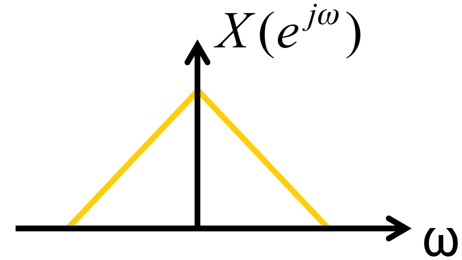


Transform

DTFT

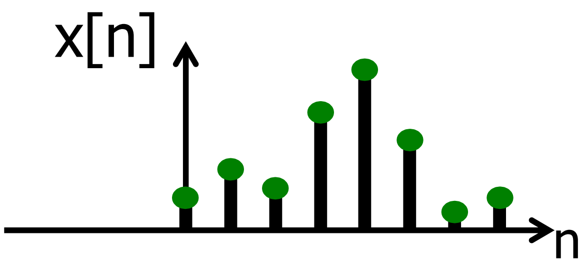
$$x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega$$

Frequency



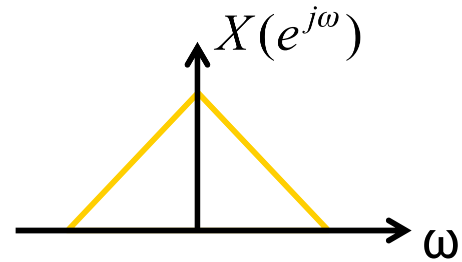
DFT Intuition

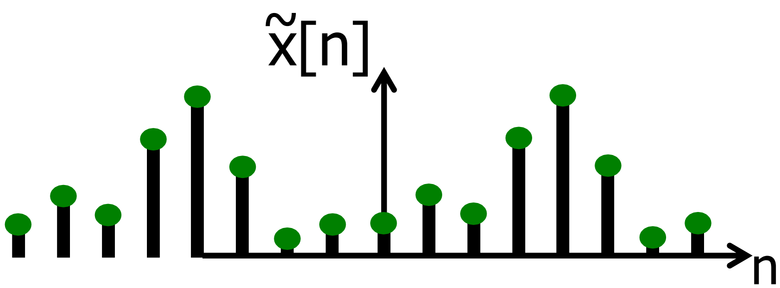
Time	Transform	Frequency



DTFT

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

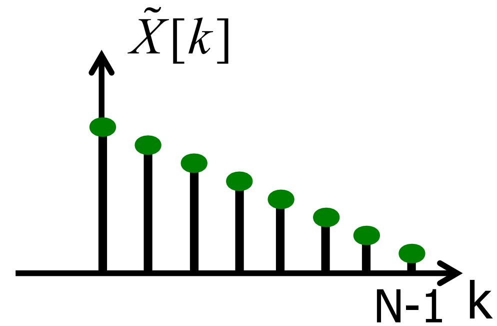




Periodic in N

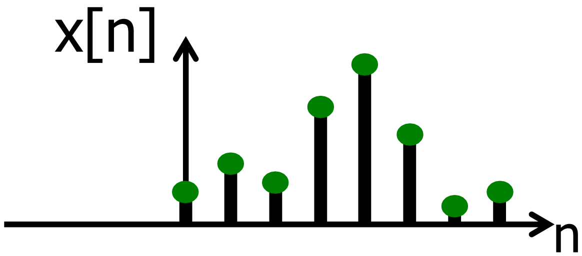
DFS

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



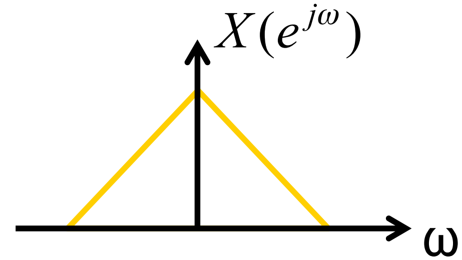
DFT Intuition

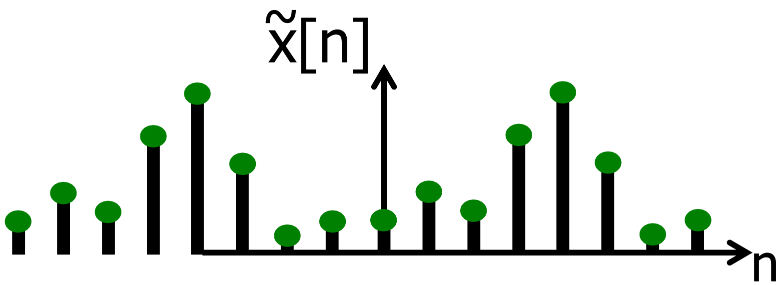
Time | **Transform** | **Frequency**



DTFT →

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

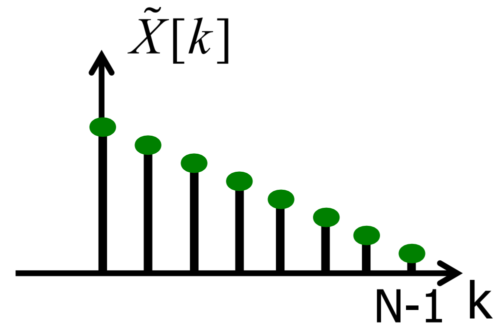




Periodic in N

DFS →

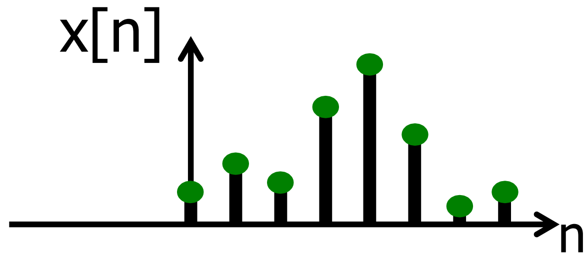
$$\tilde{x}[n] = \frac{1}{N}\sum_{k=0}^{N-1} \tilde{X}[k]W_N^{-kn}$$



$$W_N = e^{-j\frac{2\pi}{N}}$$

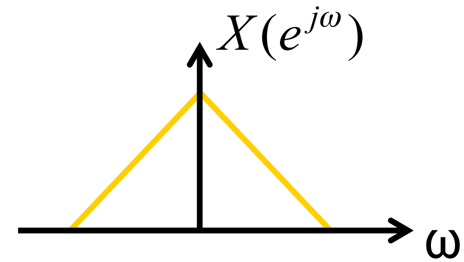
DFT Intuition

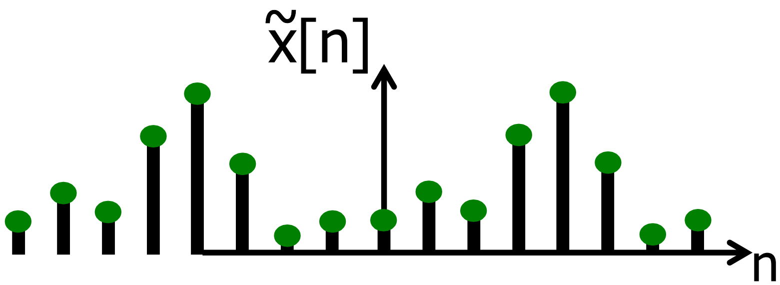
Time | **Transform** | **Frequency**



DTFT

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

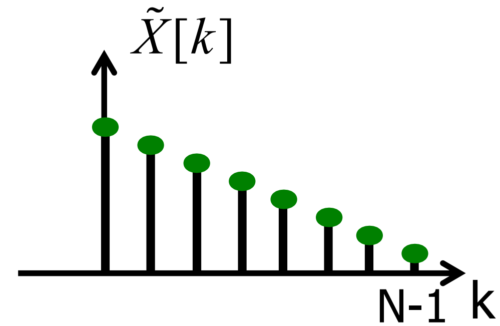


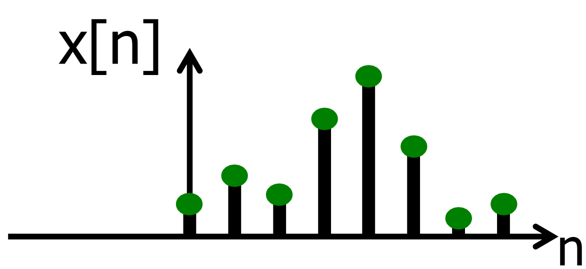


Periodic in N

DFS

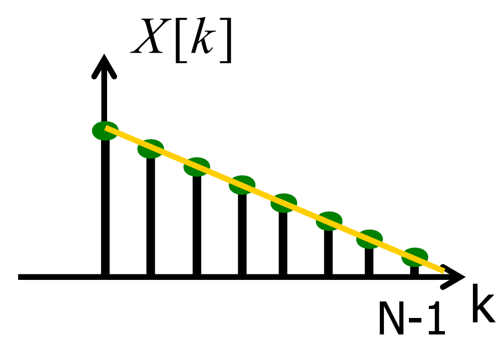
$$\tilde{x}[n] = \frac{1}{N}\sum_{k=0}^{N-1}\tilde{X}[k]W_N^{-kn}$$





DFT

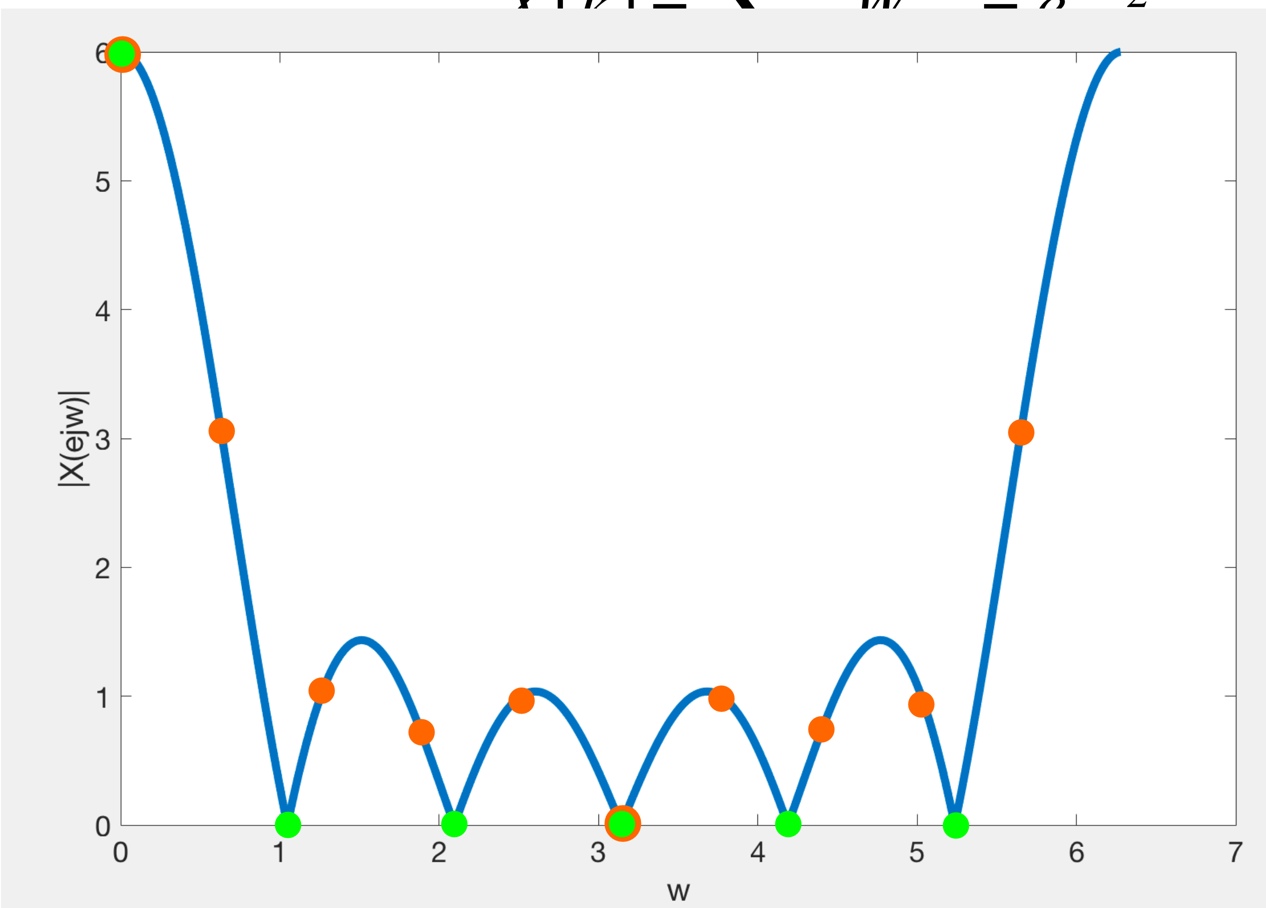
$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1}X[k]W_N^{-kn}$$



DFT vs DTFT

- Back to example

$$X[k] = \sum_{n=0}^{5} W^{nk} = e^{-j\frac{\pi}{2}k} \frac{\sin\left(\frac{3\pi}{5}k\right)}{\sin\left(\frac{\pi}{10}k\right)}$$



"6-point" DFT

"10-point" DFT

Use `fftshift` to center around dc

Circular Convolution

- Circular Convolution:

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

For two signals of length N

Note: Circular convolution is commutative

$$x_2[n] \circledast_N x_1[n] = x_1[n] \circledast_N x_2[n]$$

Linear Convolution

- Next....
  - Using DFT, circular convolution is easy
    - Matrix multiplication
  - But, linear convolution is useful, not circular
  - So, show how to perform linear convolution with circular convolution
  - Use DFT to do linear convolution (via circular convolution)

Linear Convolution

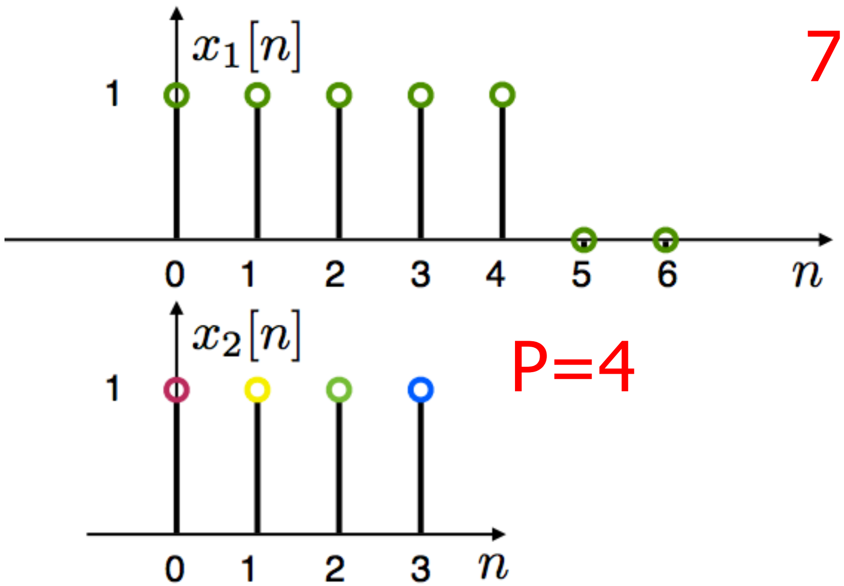
- We start with two non-periodic sequences:

$$x[n] \quad 0 \leq n \leq L-1$$
$$h[n] \quad 0 \leq n \leq P-1$$

  - E.g. x[n] is a signal and h[n] a filter's impulse response

Compute Circular Convolution Sum

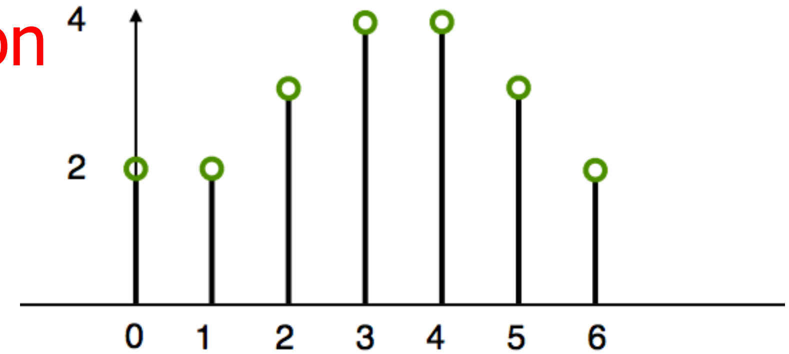
L=7



P=4

7-point circular convolution

Length=7



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

Linear Convolution

- We start with two non-periodic sequences:

$$x[n] \quad 0 \le n \le L-1$$
$$h[n] \quad 0 \le n \le P-1$$

  - E.g. x[n] is a signal and h[n] a filter's impulse response
- We want to compute the linear convolution:

$$y[n] = x[n] * h[n] = \sum_{m=0}^{L-1} x[m]h[n-m]$$

  - y[n] is nonzero for $0 \le n \le L+P-2$ (ie. length $M=L+P-1$)

Requires L*P multiplications

Linear Convolution via Circular Convolution

- Zero-pad x[n] by P-1 zeros

$$x_{\mathrm{zp}}[n] = \begin{cases} x[n] & 0 \leq n \leq L-1 \\ 0 & L \leq n \leq L+P-2 \end{cases}$$

- Zero-pad h[n] by L-1 zeros

$$h_{\mathrm{zp}}[n] = \begin{cases} h[n] & 0 \leq n \leq P-1 \\ 0 & P \leq n \leq L+P-2 \end{cases}$$

- Now, both sequences are length M=L+P-1

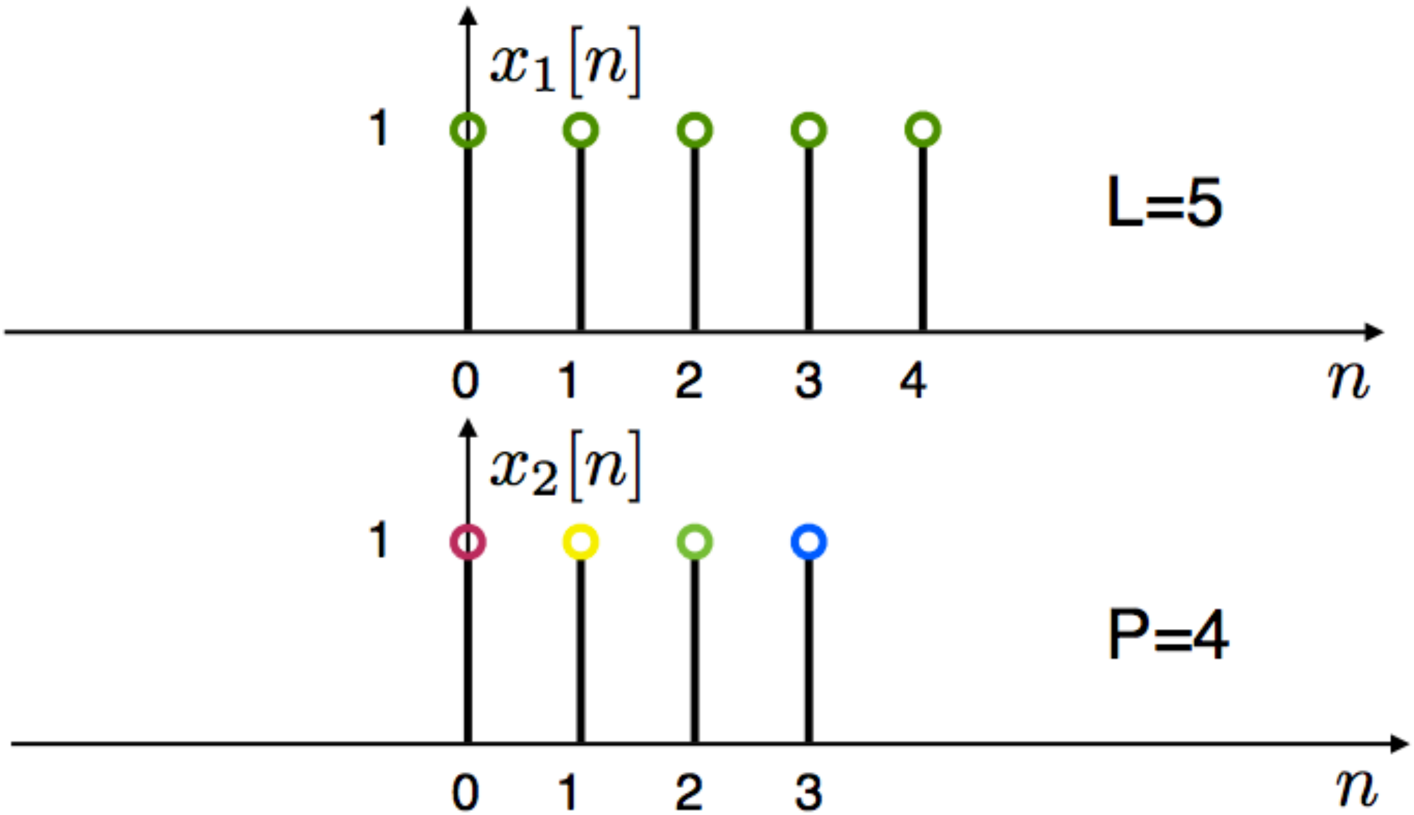
Linear Convolution via Circular Convolution

- Now, both sequences are length M=L+P-1
- We can now compute the linear convolution using a circular one with length M=L+P-1

Linear convolution via circular

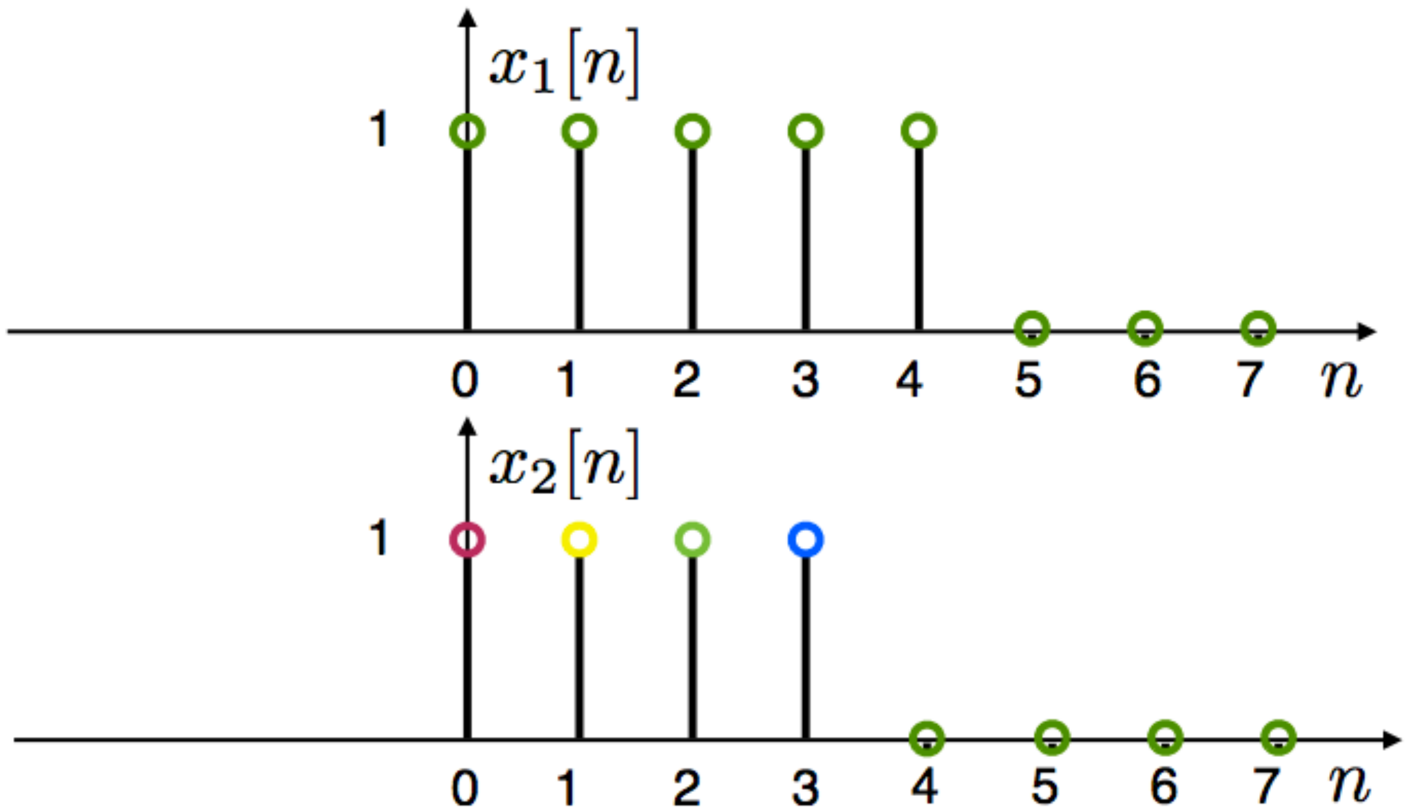
$$y[n] = x[n] * y[n] = \begin{cases} x_{\text{zp}}[n] \circledM h_{\text{zp}}[n] & 0 \le n \le M-1 \\ 0 & \text{otherwise} \end{cases}$$

Example



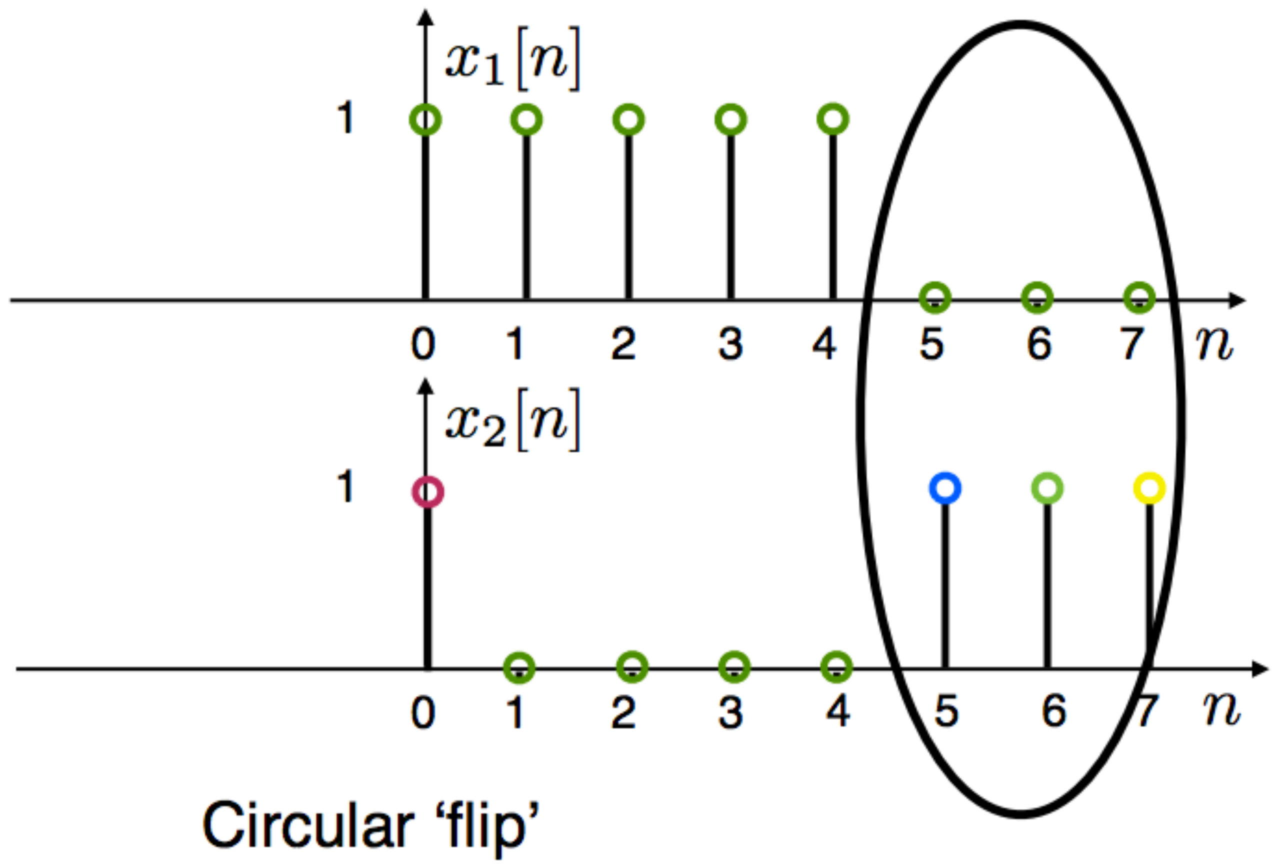
M = L + P - 1 = 8

Example



M = L + P - 1 = 8

Example



M = L + P - 1 = 8

$$y[n] = x_1[n] \ ⑧ \ x_2[n] = x_1[n] * x_2[n]$$

Linear Convolution with DFT

- In practice we can implement a circular convolution using the DFT property:

$$
\begin{aligned}
x[n] * h[n] &= x_{\mathrm{zp}}[n] \circledM h_{\mathrm{zp}}[n] \\
&= \mathcal{DFT}^{-1}\{\mathcal{DFT}\{x_{\mathrm{zp}}[n]\} \cdot \mathcal{DFT}\{h_{\mathrm{zp}}[n]\}\}
\end{aligned}
$$

for $0 \leq n \leq M-1$, $M=L+P-1$

Linear Convolution with DFT

- In practice we can implement a circular convolution using the DFT property:

$$\begin{aligned} x[n] * h[n] &= x_{\mathrm{zp}}[n] \circledM h_{\mathrm{zp}}[n] \\ &= \mathcal{DFT}^{-1}\{\mathcal{DFT}\{x_{\mathrm{zp}}[n]\} \cdot \mathcal{DFT}\{h_{\mathrm{zp}}[n]\}\} \end{aligned}$$

for $0 \leq n \leq M-1$, $M=L+P-1$

- Advantage: DFT can be computed with $N\log_2 N$ complexity (FFT algorithm later!)

Linear Convolution with DFT

- In practice we can implement a circular convolution using the DFT property:

$$\begin{aligned} x[n] * h[n] &= x_{\text{zp}}[n] \circledM h_{\text{zp}}[n] \\ &= \mathcal{DFT}^{-1}\{\mathcal{DFT}\{x_{\text{zp}}[n]\} \cdot \mathcal{DFT}\{h_{\text{zp}}[n]\}\} \end{aligned}$$

for $0 \leq n \leq M-1$, $M=L+P-1$

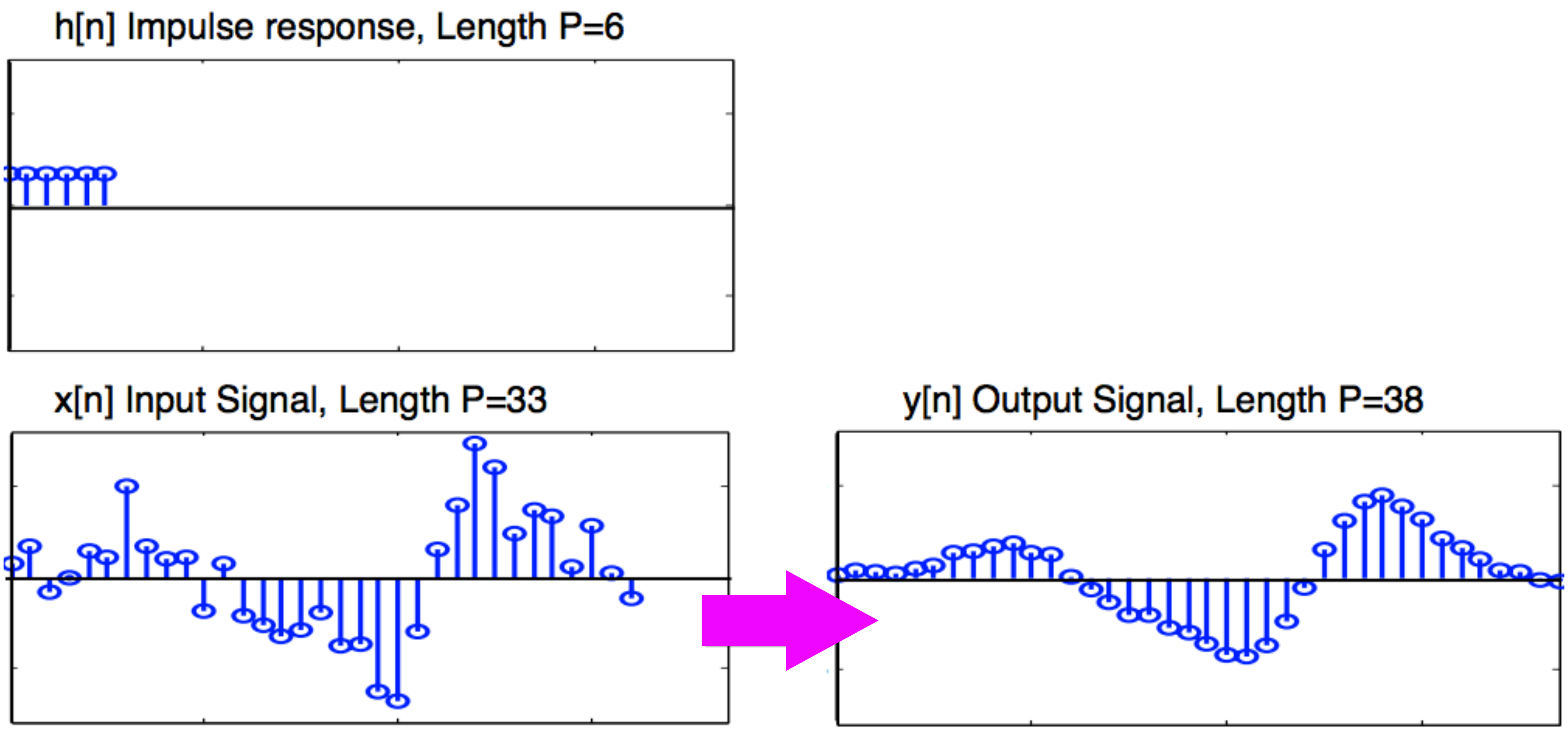
- Advantage: DFT can be computed with $N\log_2 N$ complexity (FFT algorithm later!)
- Drawback: Must wait for all the samples -- huge delay -- incompatible with real-time filtering

Block Convolution

- Problem:
  - An input signal x[n], has very long length (could be considered infinite)
  - An impulse response h[n] has length P
  - We want to take advantage of DFT/FFT and compute convolutions in blocks that are shorter than the signal
- Approach:
  - Break the signal into small blocks
  - Compute convolutions (via DFT)
  - Combine the results
    - Overlap-add
    - Overlap-save

Block Convolution

Example:



Overlap-Add Method

- Decompose into non-overlapping segments

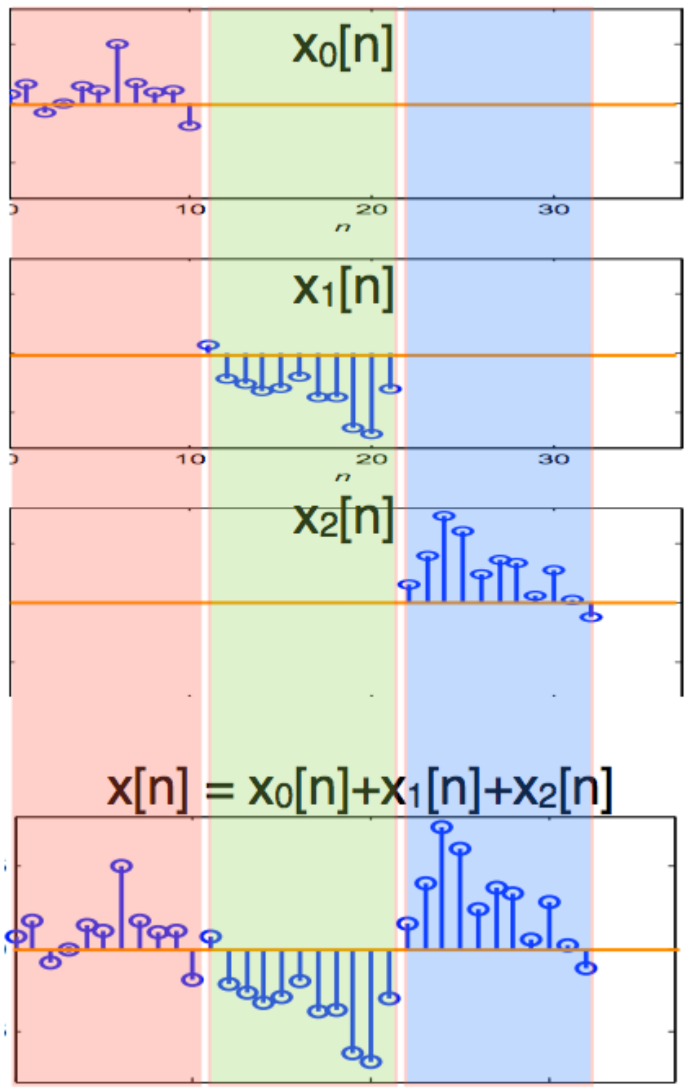
$$x_r[n] = \begin{cases} x[n] & rL \le n < (r+1)L \\ 0 & \text{otherwise} \end{cases}$$

- The input signal is the sum of segments

$$x[n] = \sum_{r=0}^{\infty} x_r[n]$$

Example

L=11



Overlap-Add Method

$$x[n] = \sum_{r=0}^{\infty} x_r[n]$$

- The output is:

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

  - Each output segment $x_r[n]*h[n]$ is length M=L+P-1
    - h[n] has length P
    - $x_r[n]$ has length L

Overlap-Add Method

- We can compute $x_r[n] * h[n]$ using circular convolution with the DFT
- Using the DFT:
  - Zero-pad $x_r[n]$ to length M
  - Zero-pad h[n] to length M and compute $DFT_M\{h_{zp}[n]\}$
    - Only need to do once!

Overlap-Add Method

- We can compute $x_r[n]*h[n]$ using circular convolution with the DFT
- Using the DFT:
  - Zero-pad $x_r[n]$ to length M
  - Zero-pad h[n] to length M and compute $DFT_N\{h_{zp}[n]\}$
    - Only need to do once!
  - Compute:

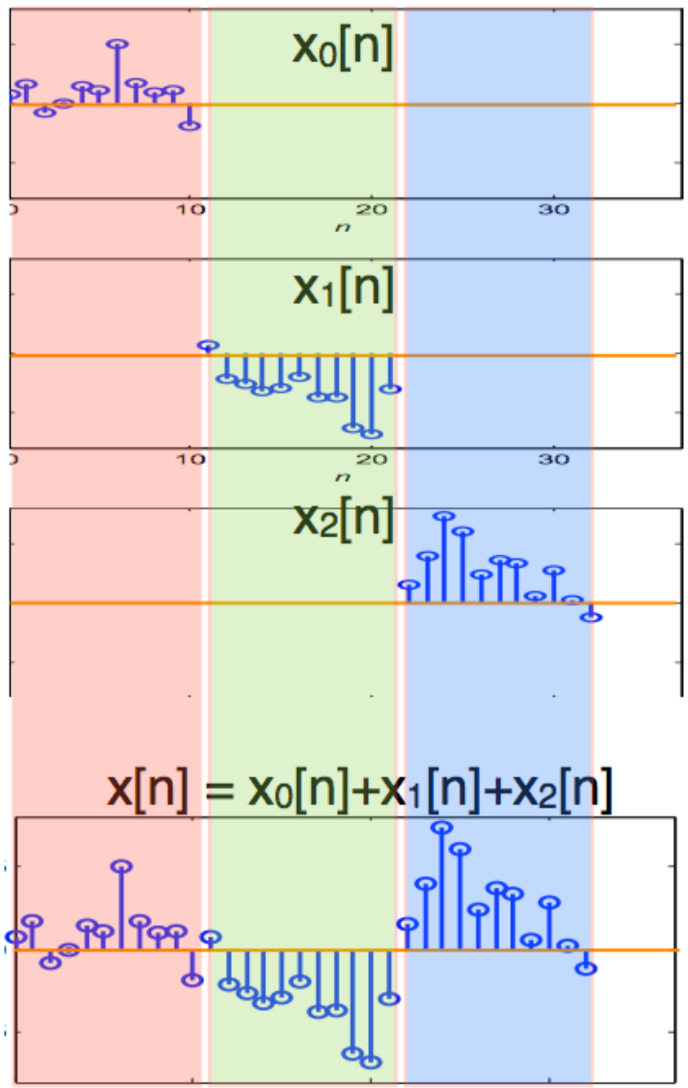
$$x_r[n] * h[n] = \mathrm{DFT}^{-1}\left\{\mathrm{DFT}\{x_{r,zp}[n]\} \cdot \mathrm{DFT}\{h_{zp}[n]\}\right\}$$

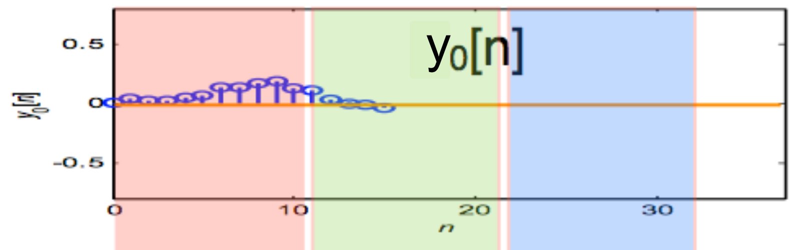
- Results are of length M=L+P-1
  - Neighboring results overlap by P-1
  - Add overlaps to get final sequence

Example of Overlap-Add

L+P-1=16

L=11





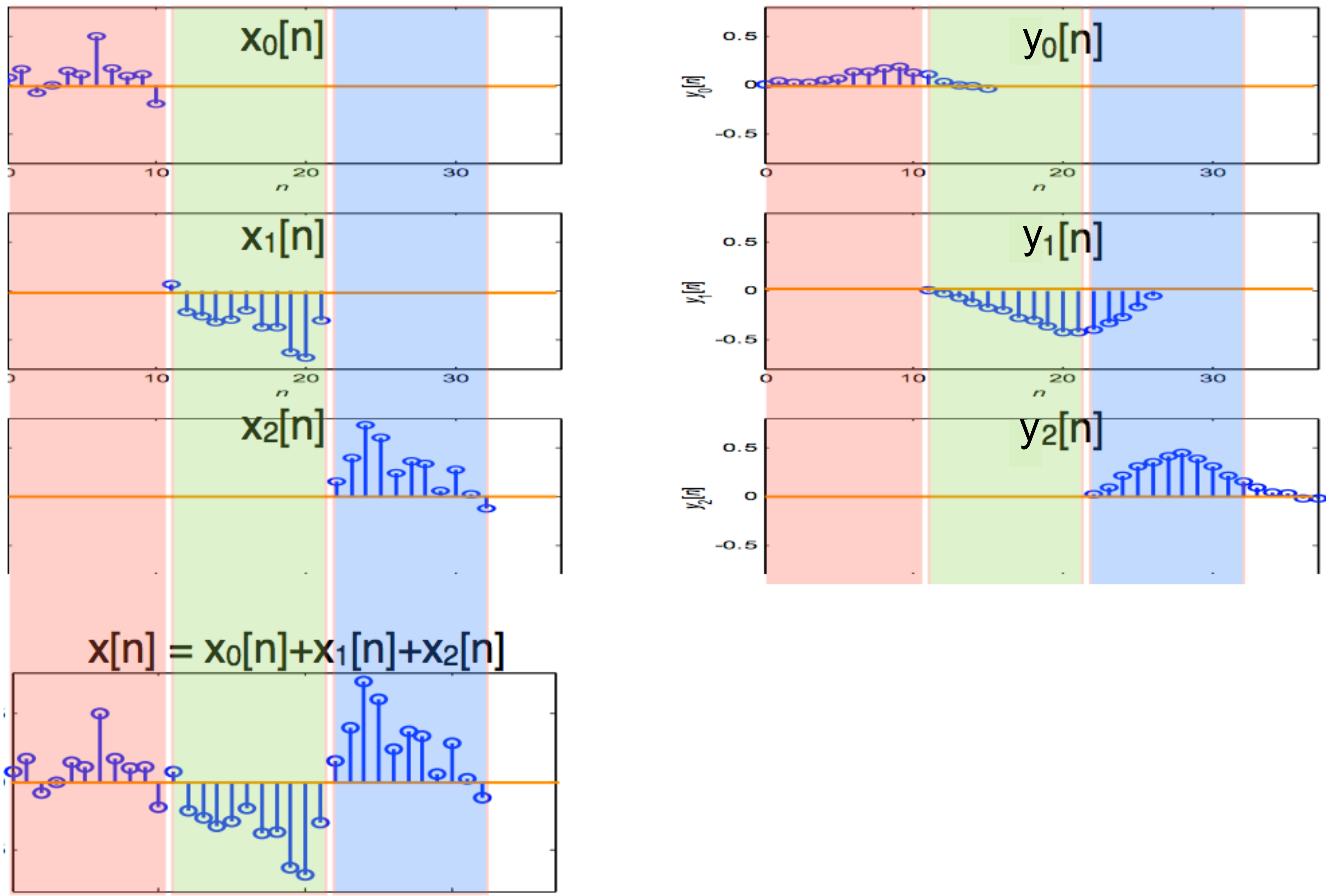
Example:

h[n] Impulse response, Length P=6

Example of Overlap-Add

L+P-1=16

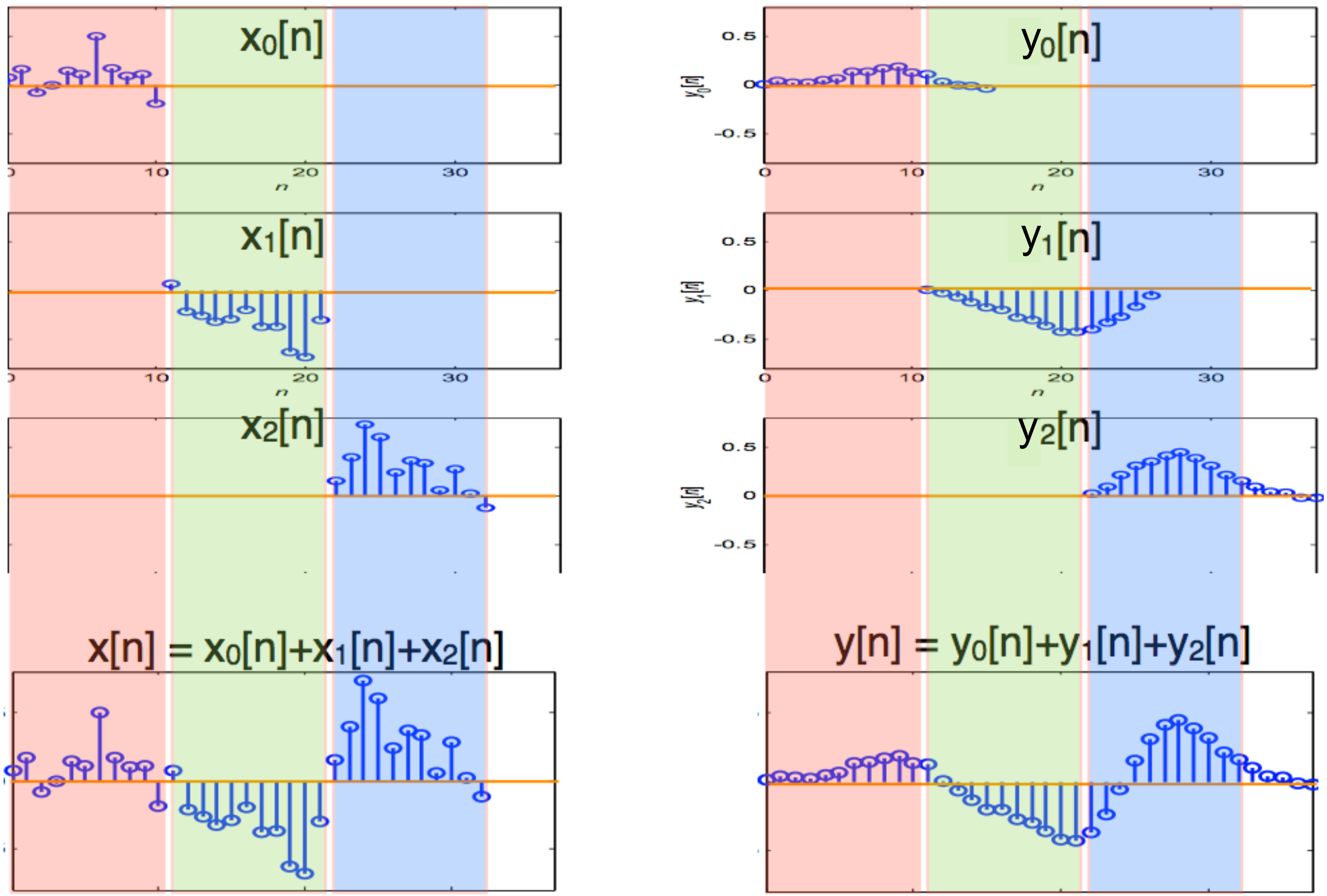
L=11



Example of Overlap-Add

L+P-1=16

L=11



Overlap-Save Method

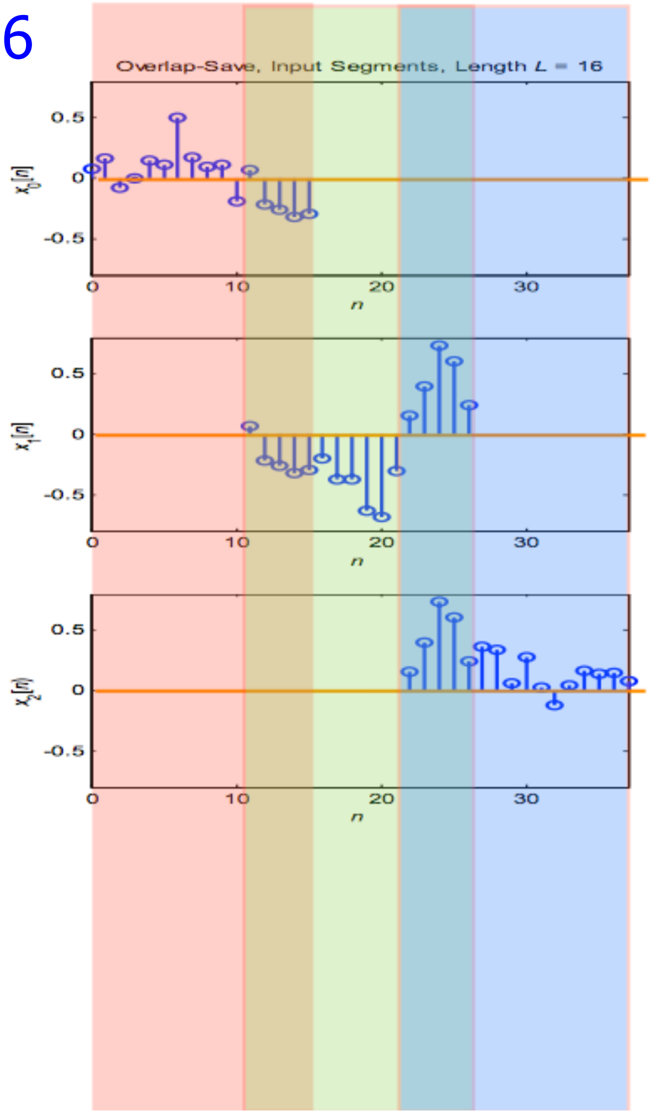
- Basic idea:
- Split input into overlapping segments with length L+P-1
  - P-1 sample overlap

$$x_r[n] = \begin{cases} x[n] & rL \leq n < (r+1)L + P \\ 0 & \text{otherwise} \end{cases}$$

- Perform circular convolution in each segment, and keep the L sample portion which is a valid linear convolution

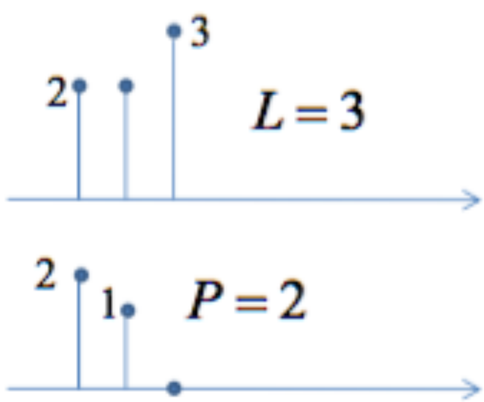
Example of Overlap-Save

L+P-1=16



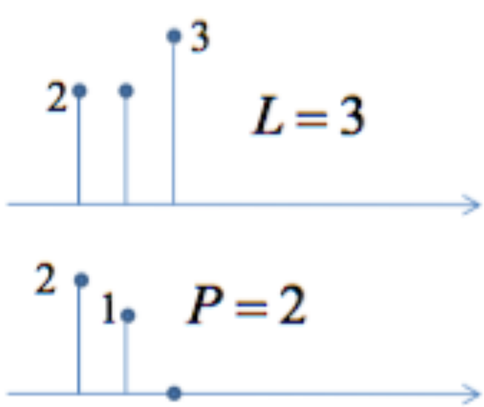
Circular to Linear Convolution

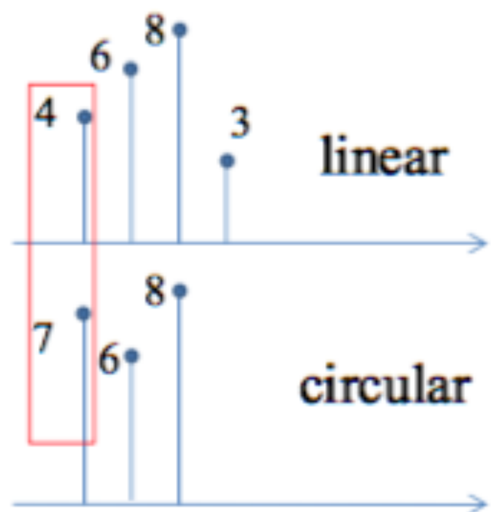
- An L-point sequence circularly convolved with a P-point sequence
  - with L - P zeros padded, $P < L$
- gives an L-point result with
  - the first P - 1 values *incorrect* and
  - the next L - P + 1 the *correct* linear convolution result



Circular to Linear Convolution

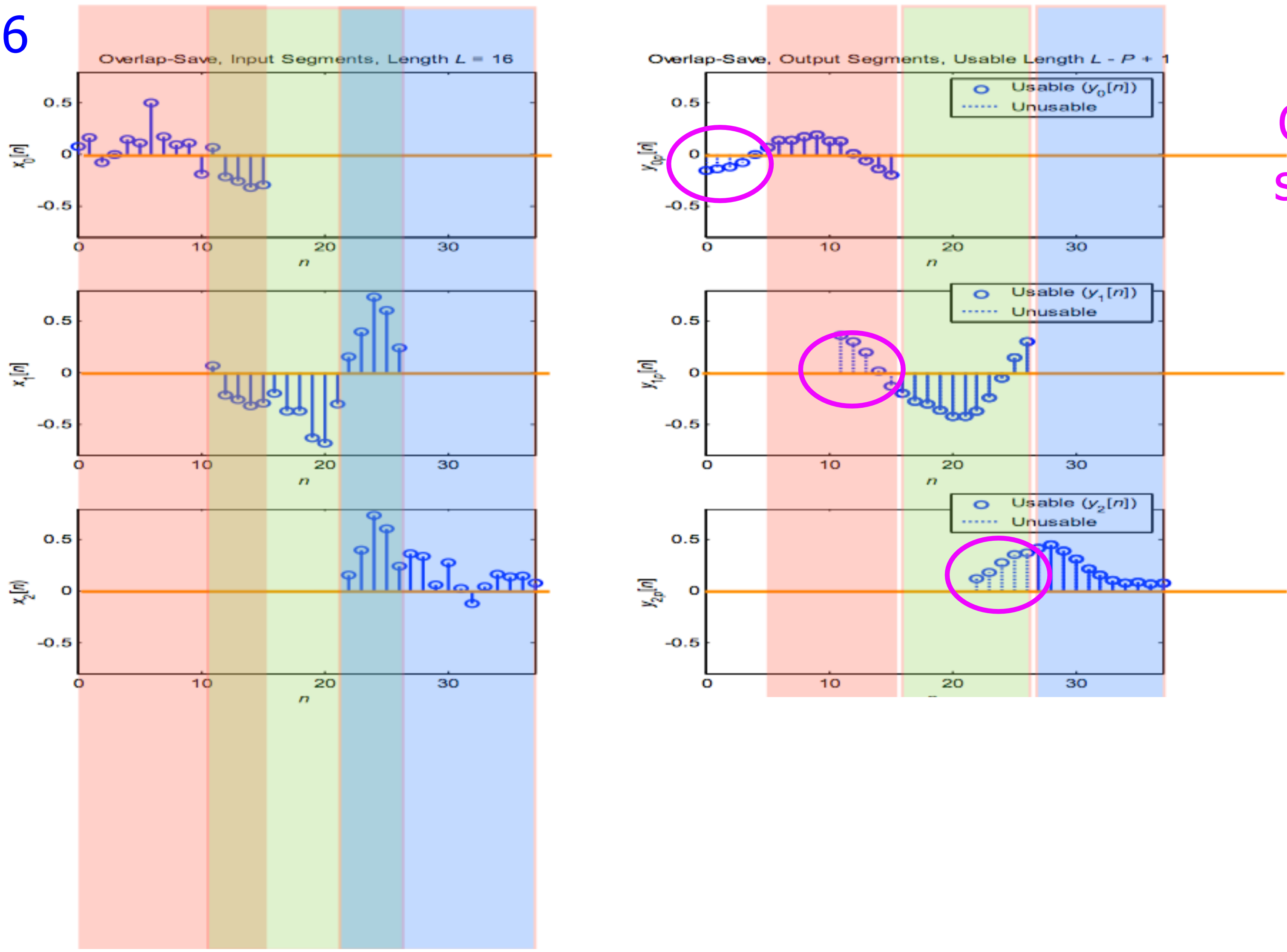
- An L-point sequence circularly convolved with a P-point sequence
  - with L - P zeros padded, $P < L$
- gives an L-point result with
  - the first P - 1 values *incorrect* and
  - the next L - P + 1 the *correct* linear convolution result





Example of Overlap-Save

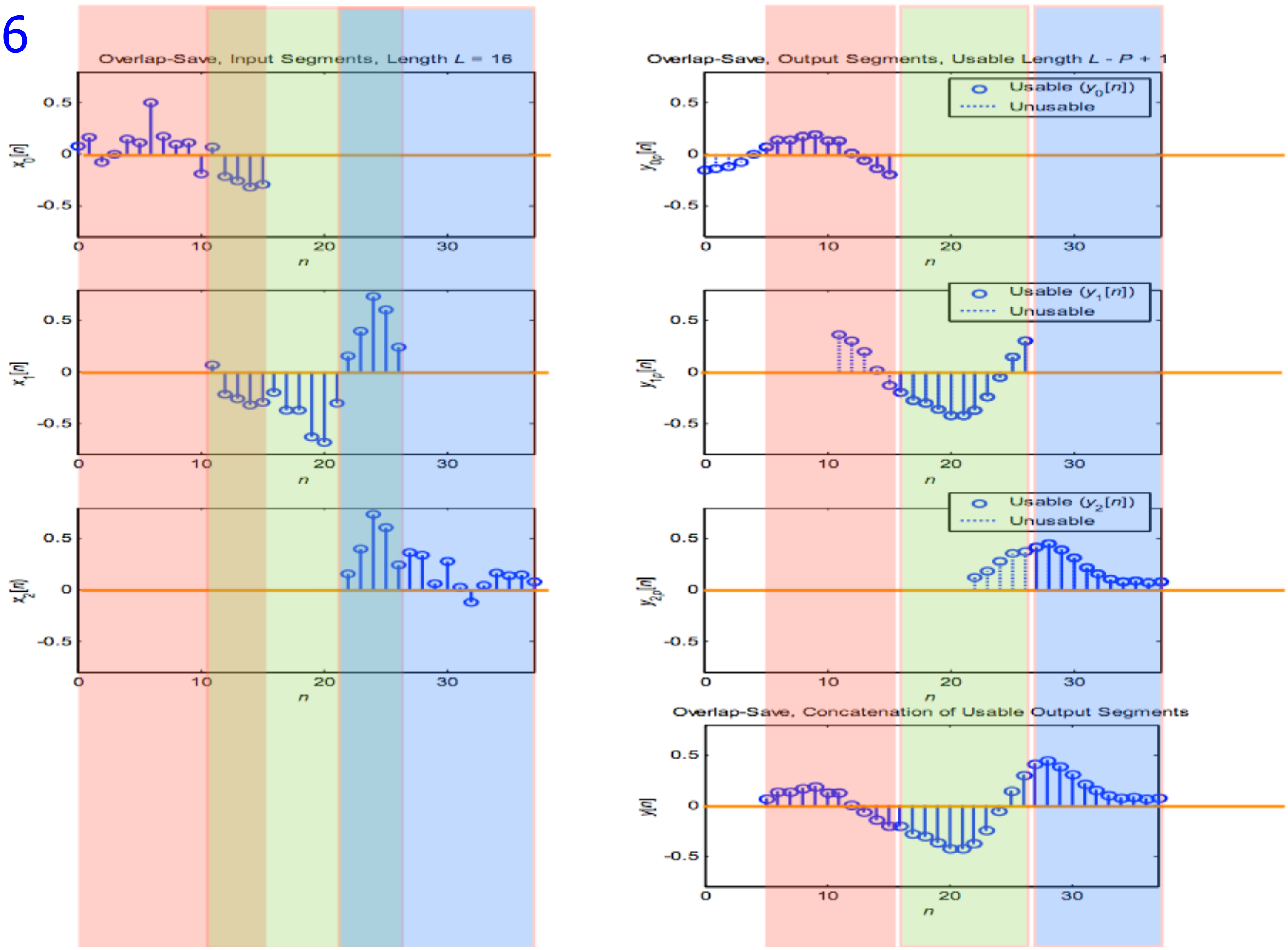
L+P-1=16



P-1=5
Overlap
samples

Example of Overlap-Save

L+P-1=16

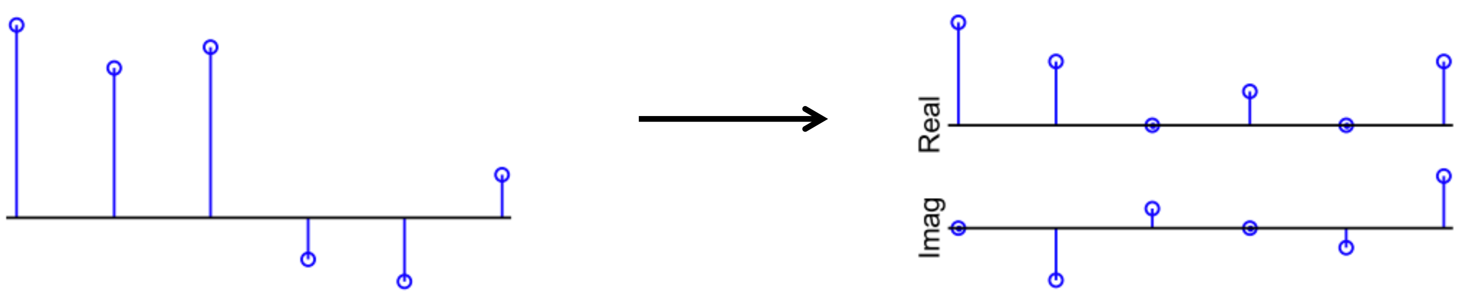


Discrete Cosine Transform

- Similar to the discrete Fourier transform (DFT), but using only real numbers
- Widely used in lossy compression applications (eg. Mp3, JPEG)
- Why use it?

DFT Problems

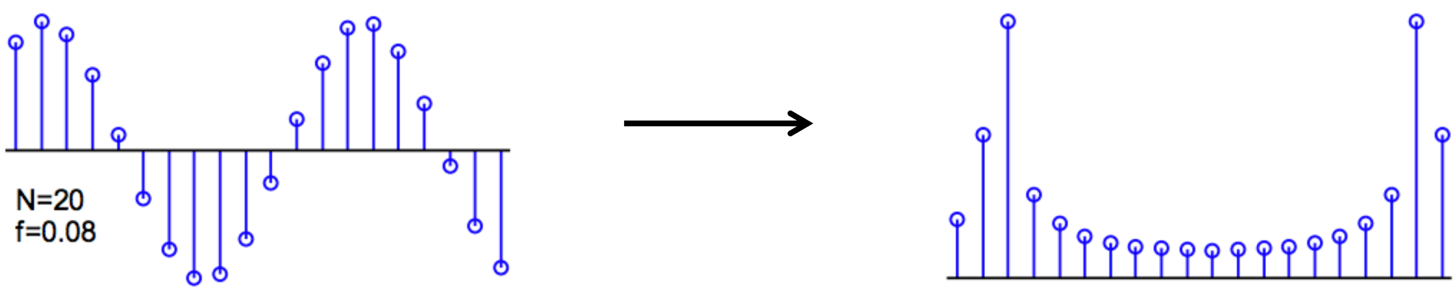
- For processing 1-D or 2-D signals (especially coding), a common method is to divide the signal into “frames” and then apply an invertible transform to each frame that compresses the information into few coefficients.
- The DFT has some problems when used for this purpose:
  - N real x[n] ↔ N complex X[k] : 2 real, $N/2 - 1$ conjugate pairs
  - DFT is of the periodic signal formed by replicating x[n]



DFT Problems

- For processing 1-D or 2-D signals (especially coding), a common method is to divide the signal into "frames" and then apply an invertible transform to each frame that compresses the information into few coefficients.
- The DFT has some problems when used for this purpose:
  - N real x[n] ↔ N complex X[k] : 2 real, $N/2 - 1$ conjugate pairs
  - DFT is of the periodic signal formed by replicating x[n]

    ⇒ Spurious frequency components from boundary discontinuity



The Discrete Cosine Transform (DCT) overcomes these problems.

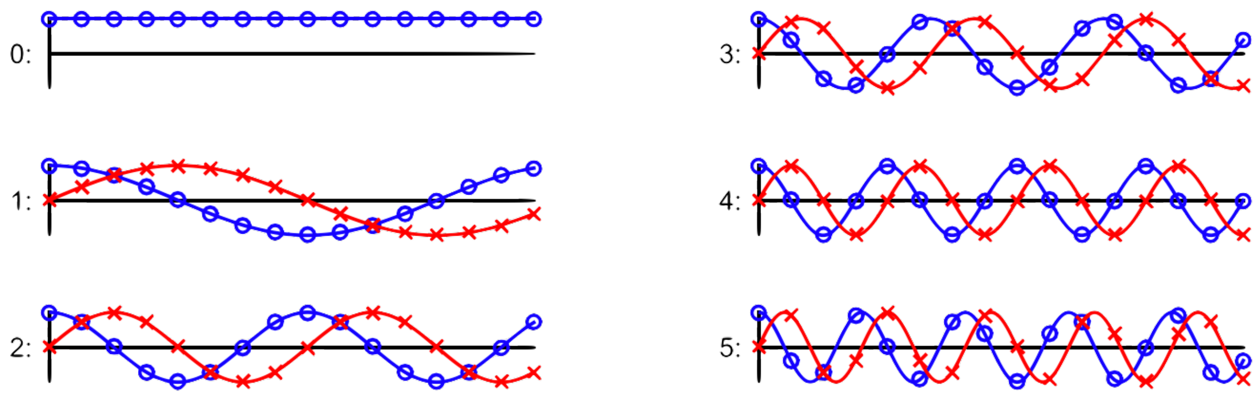
Discrete Cosine Transform

Forward DCT: $X_C[k] = \sum_{n=0}^{N-1} x[n] \cos \frac{2\pi(2n+1)k}{4N}$ for $k = 0 : N-1$

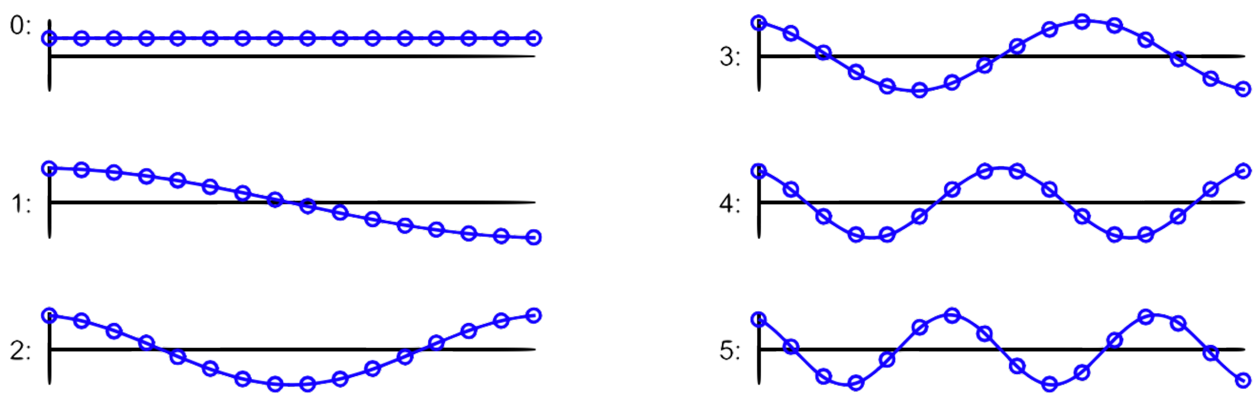
Inverse DCT: $x[n] = \frac{1}{N} X[0] + \frac{2}{N} \sum_{k=1}^{N-1} X[k] \cos \frac{2\pi(2n+1)k}{4N}$

Basis Functions

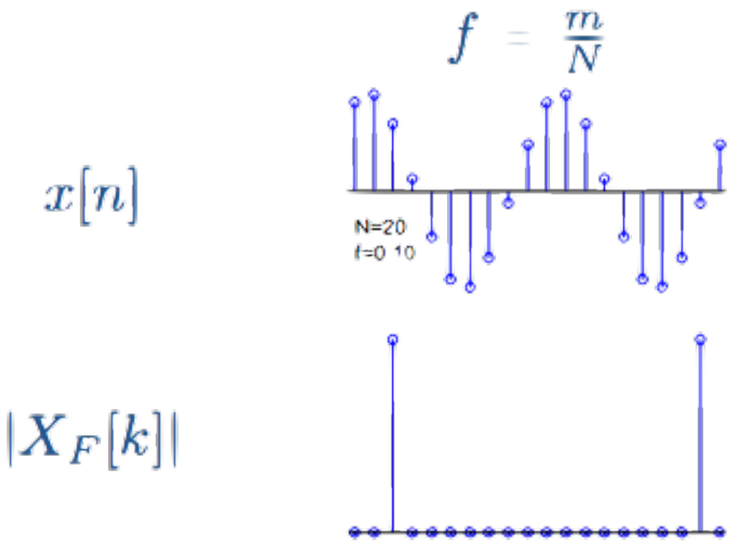
DFT basis functions: $x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j2\pi\frac{kn}{N}}$



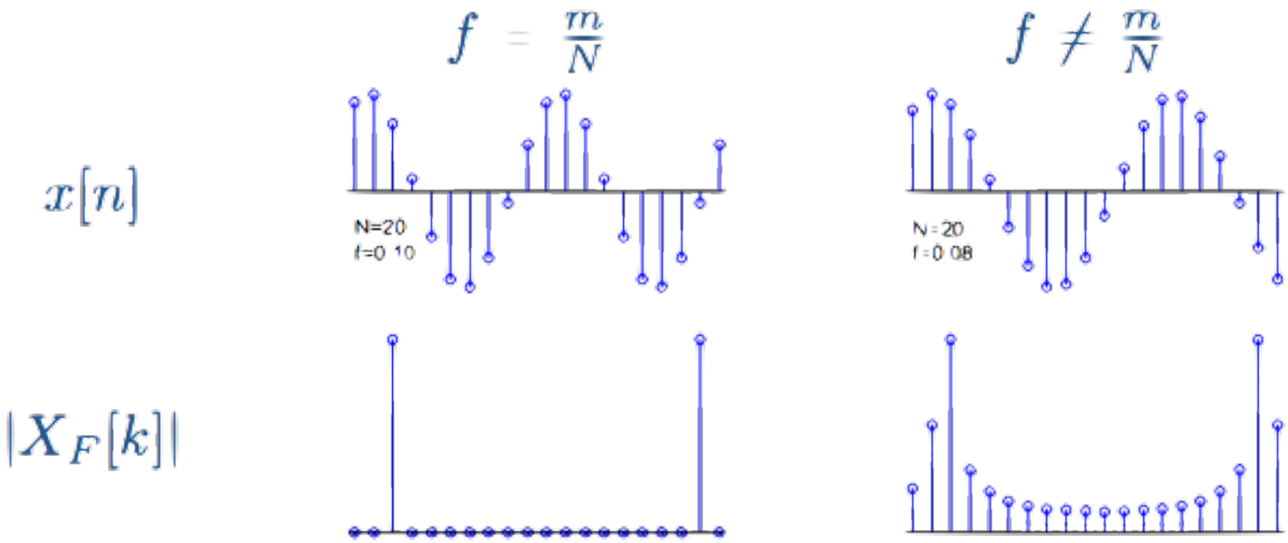
DCT basis functions: $x[n] = \frac{1}{N}X[0] + \frac{2}{N}\sum_{k=1}^{N-1} X[k]\cos\frac{2\pi(2n+1)k}{4N}$



DFT of Sine Wave

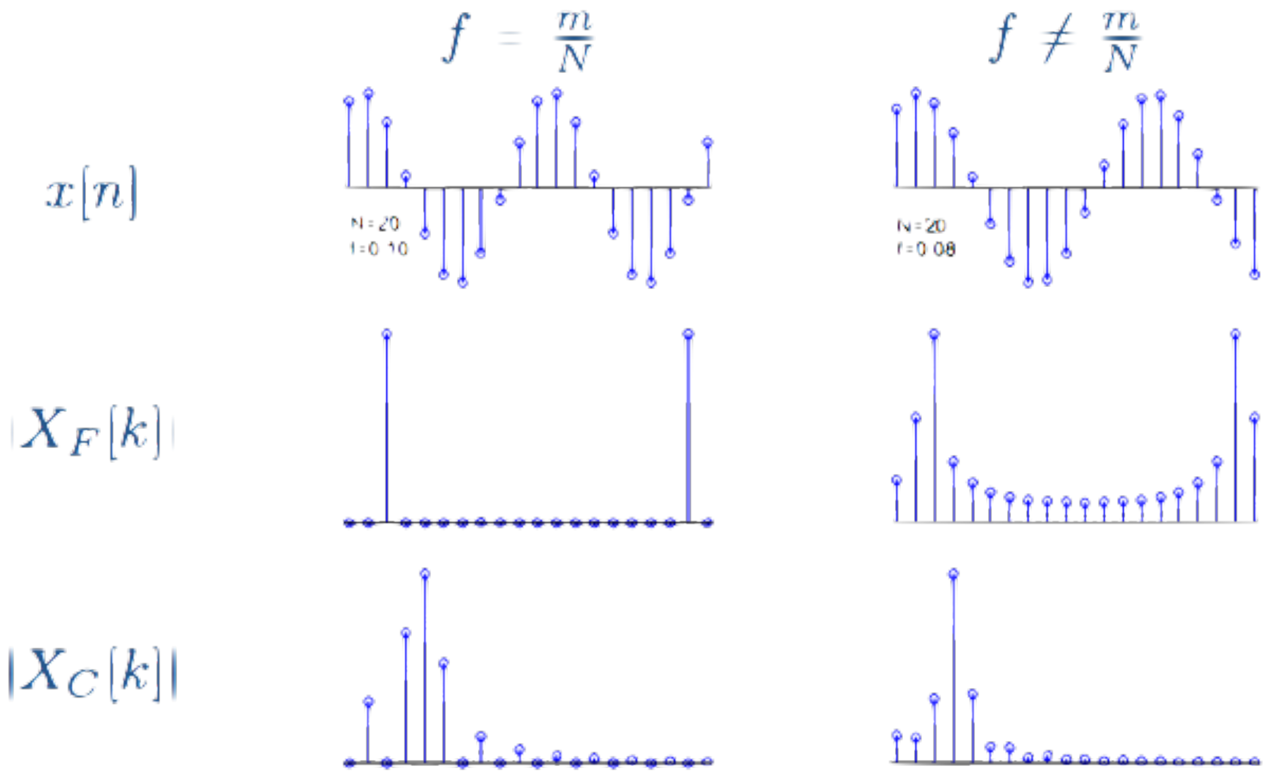


DFT of Sine Wave



DFT: Real→Complex; Freq range $[0, 1]$; Poorly localized unless $f = \frac{m}{N}$; $|X_F[k]| \propto k^{-1}$ for $Nf < k \ll \frac{N}{2}$

DCT of Sine Wave



DFT: Real→Complex; Freq range $[0, 1]$; Poorly localized unless $f = \frac{m}{N}$; $|X_F[k]| \propto k^{-1}$ for $Nf < k \ll \frac{N}{2}$

DCT: Real→Real; Freq range $[0, 0.5]$; Well localized $\forall f$; $|X_C[k]| \propto k^{-2}$ for $2Nf < k < N$

Big Ideas

- Discrete Fourier Transform (DFT)
  - For finite signals assumed to be zero outside of defined length
  - N-point DFT is sampled DTFT at N points
  - DFT properties inherited from DFS, but circular operations!
- Fast Convolution Methods
  - Use circular convolution (i.e DFT) to perform fast linear convolution
    - Overlap-Add, Overlap-Save
- DCT useful for frame rate compression of large signals

Admin

- Project 1
  - Due Monday 4/5 (can turn in by 4/9 with no penalty)
- HW 7 out now