

Digital Signal Processing: Convolution

CPSC 501: Advanced Programming Techniques
Fall 2020

Jonathan Hudson, Ph.D
Instructor
Department of Computer Science
University of Calgary

Wednesday, November 23, 2022

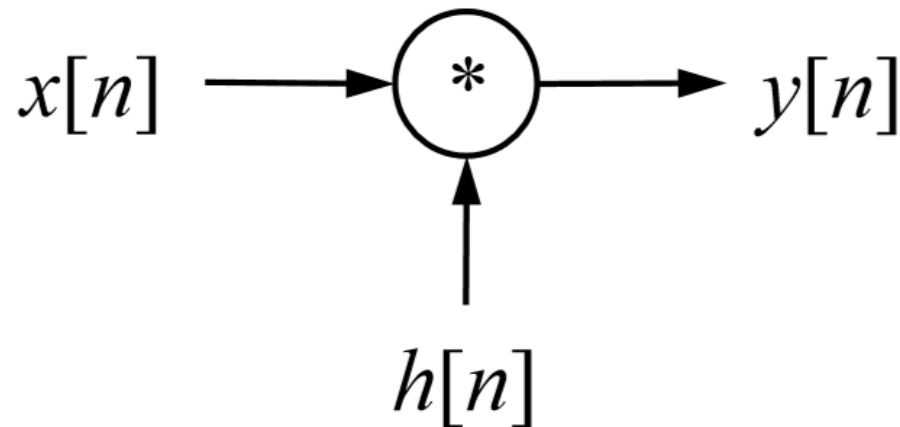


UNIVERSITY OF
CALGARY

Convolution

Time-Domain Convolution

- Convolution is mathematical operation that takes 2 input signals, $x[n]$ and $h[n]$, to produce a third signal, $y[n]$
 - Represented mathematically as:
 $x[n] * h[n] = y[n]$
 - Note: $*$ means convolution, not multiplication
 - Represented graphically:



Time-Domain Convolution

- Input Side Algorithm (array implementation):

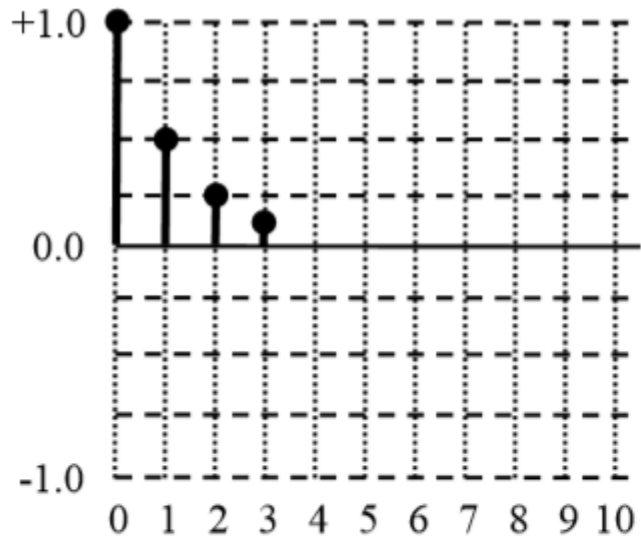
```
void convolve(float x[], int N, float h[], int M, float y[], int P) {  
    int n, m;  
    /* Outer loop: process each input value x[n] in turn */  
    for (n = 0; n < N; n++) {  
        /* Inner loop: process x[n] with each sample of h[n] */  
        for (m = 0; m < M; m++)  
            y[n+m] += x[n] * h[m];  
    }  
}
```

Time-Domain Convolution

- Note:
 - N , M , and P are the number of elements in each array
 - P must equal $N + M - 1$
 - The signals $x[n]$, $h[n]$, and $y[n]$ are f.p. numbers scaled to the range:
 - -1.0 to +1.0
 - You may need to convert to/from signed integers
 - This algorithm can be adapted to deal with samples stored in audio files

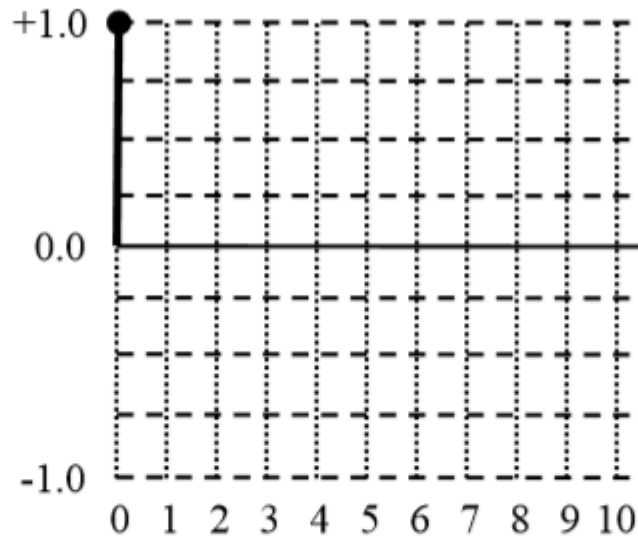
Examples

Example convolutions:



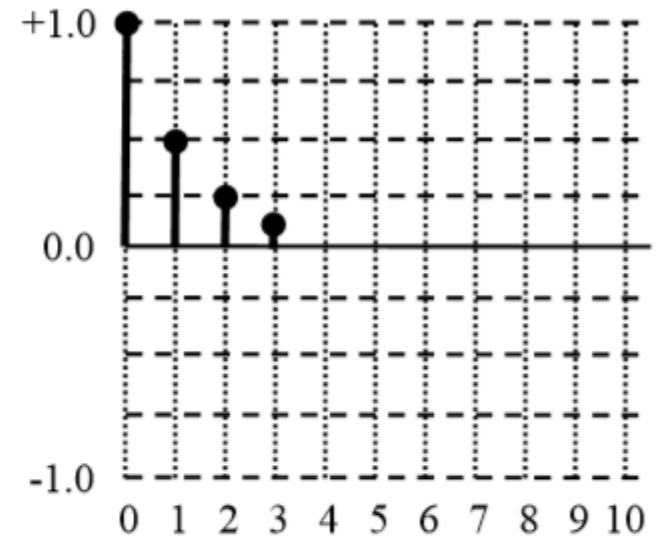
$x[n]$

*



$h[n]$

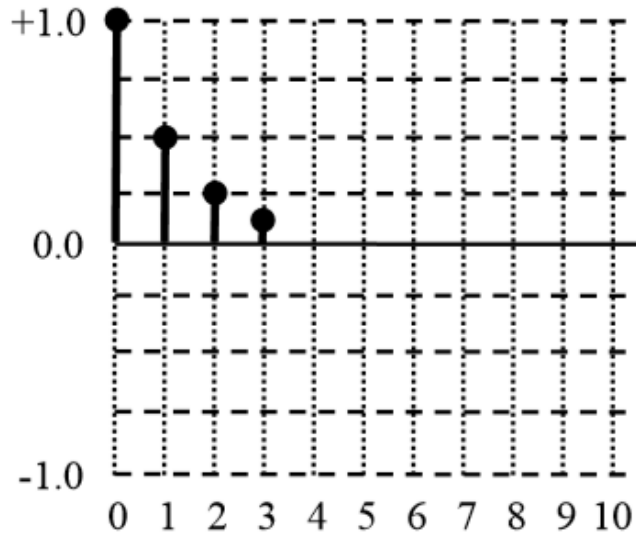
=



$y[n]$

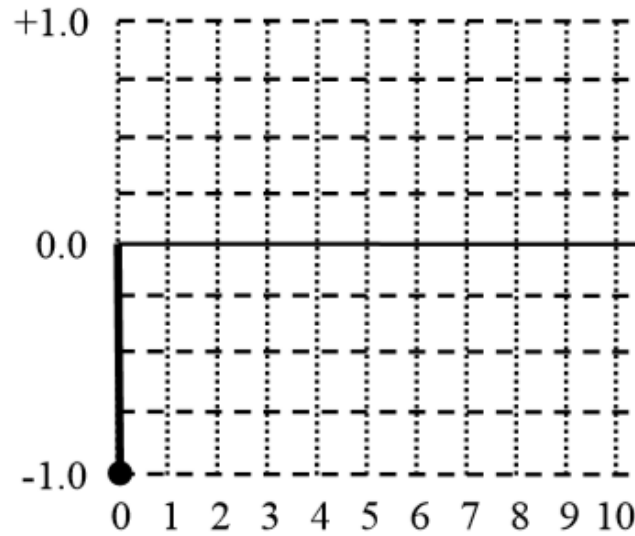
“identity”
impulse response

Example convolutions:



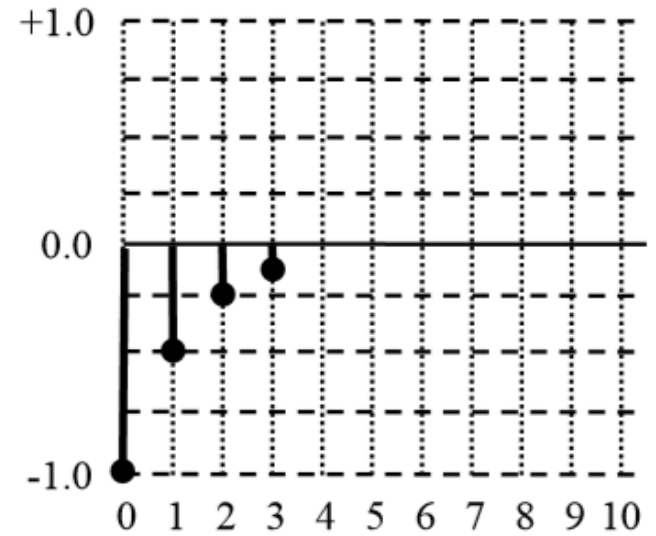
$x[n]$

*



$h[n]$

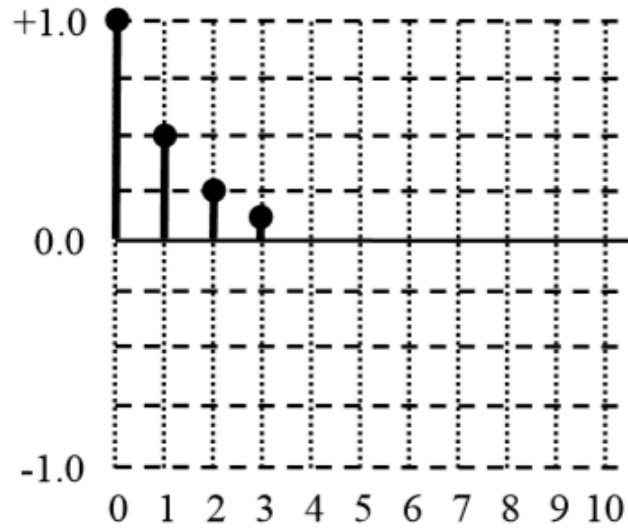
=



$y[n]$

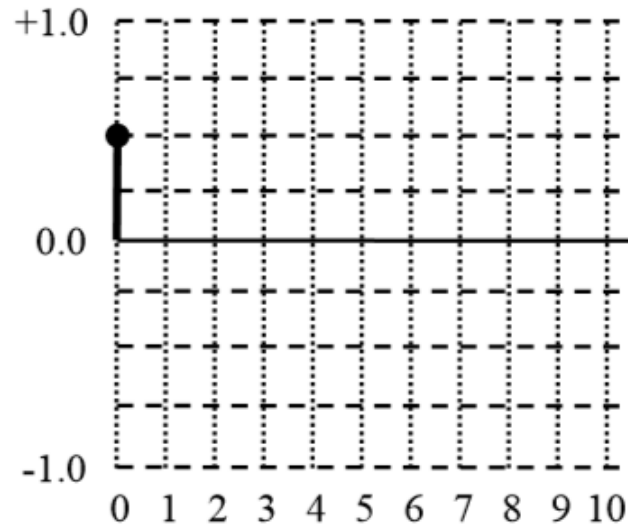
“inverse” IR

Example convolutions:



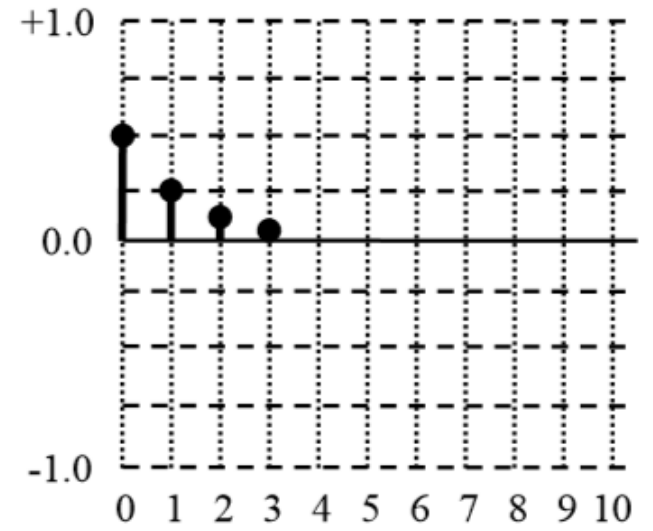
$x[n]$

*



$h[n]$

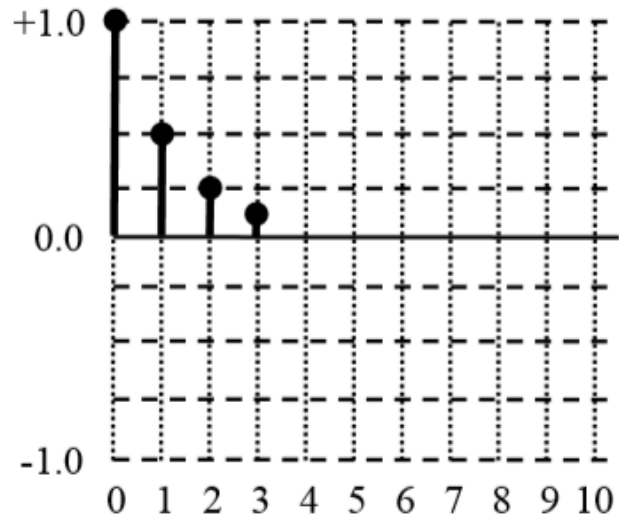
=



$y[n]$

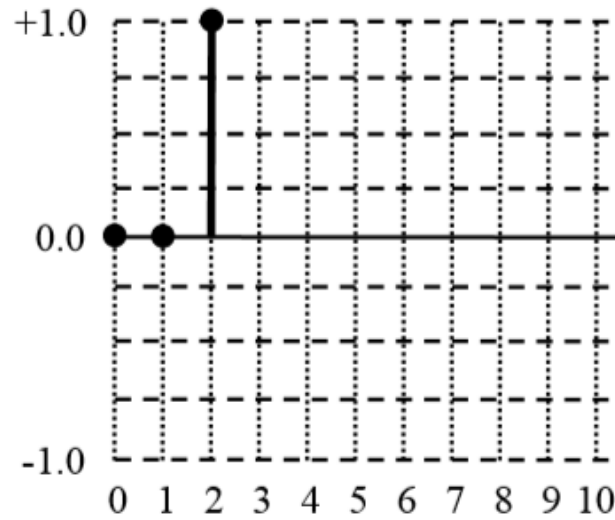
“scaling” IR

Example convolutions:



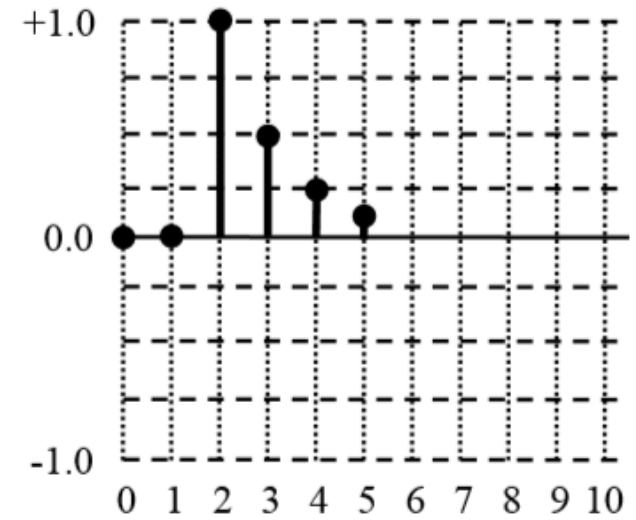
$x[n]$

*



$h[n]$

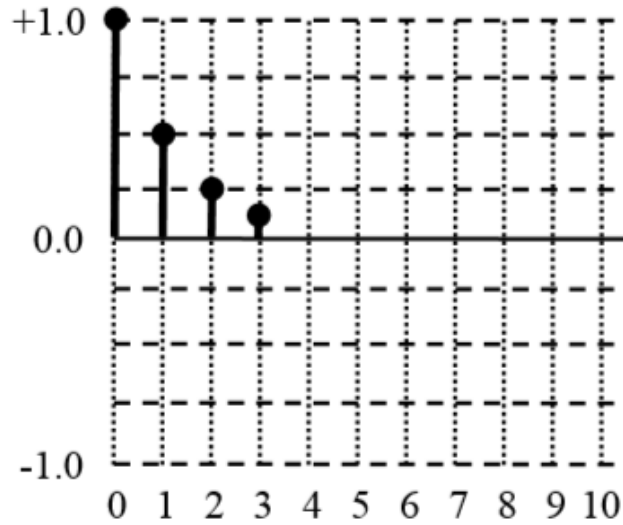
=



$y[n]$

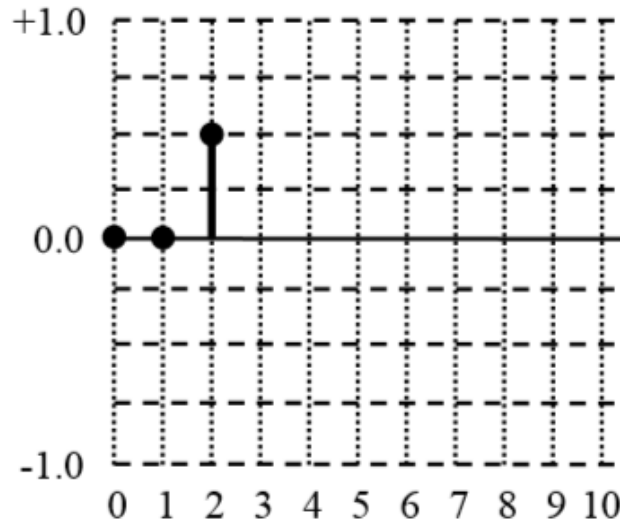
“delay” IR

Example convolutions:



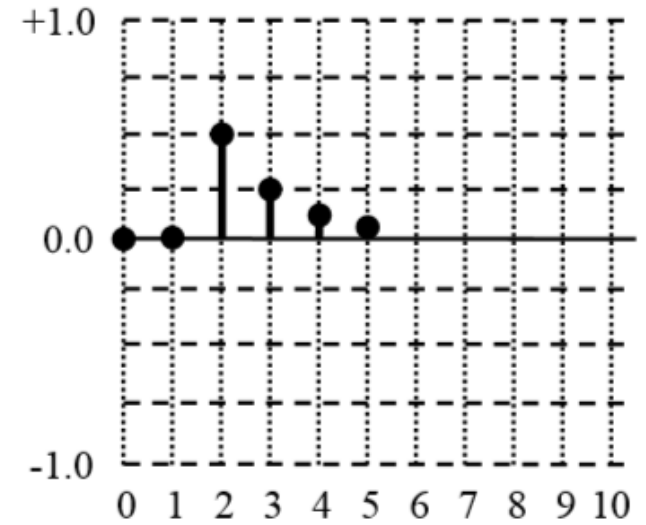
$x[n]$

*



$h[n]$

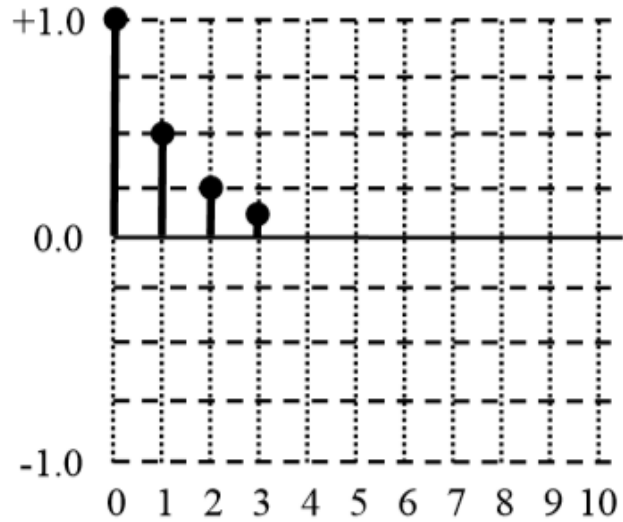
=



$y[n]$

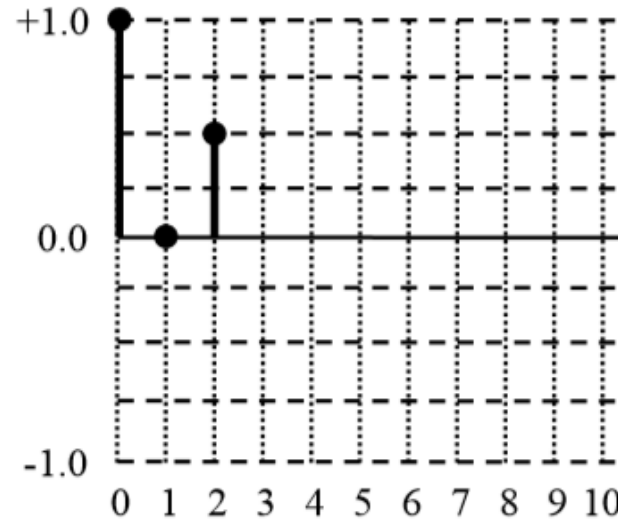
“delay &
scaling” IR

Example convolutions:



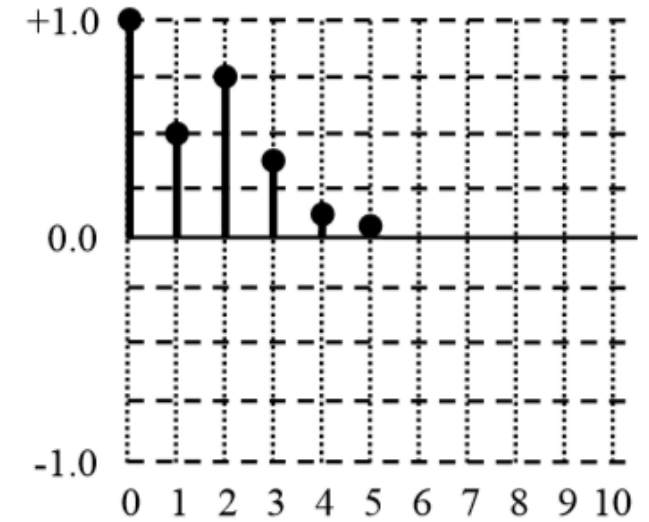
$x[n]$

*



$h[n]$

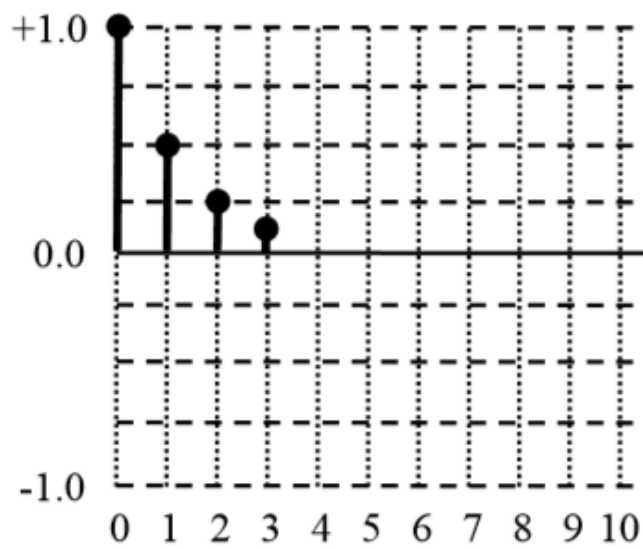
=



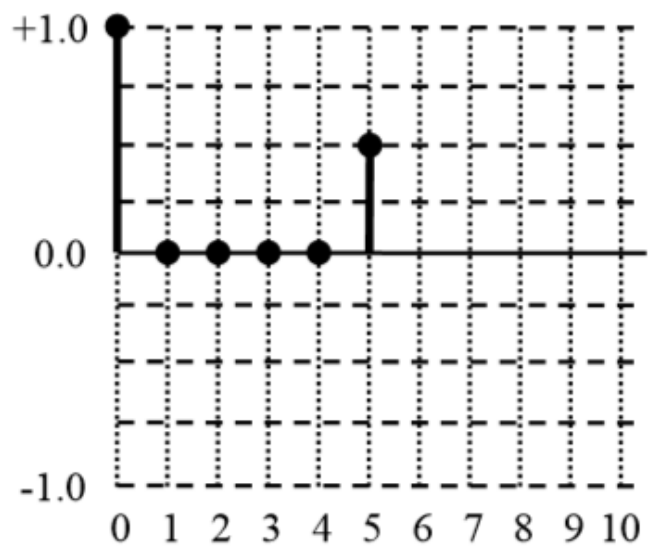
$y[n]$

“overlapping
echo” IR

Example convolutions:



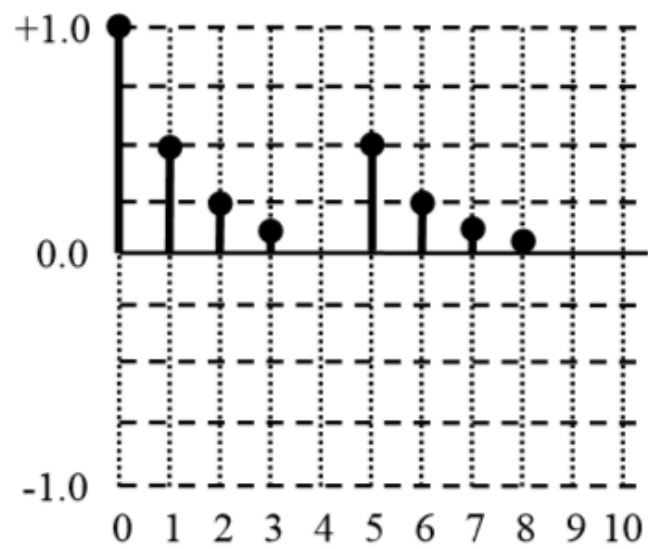
$x[n]$



$h[n]$

*

=

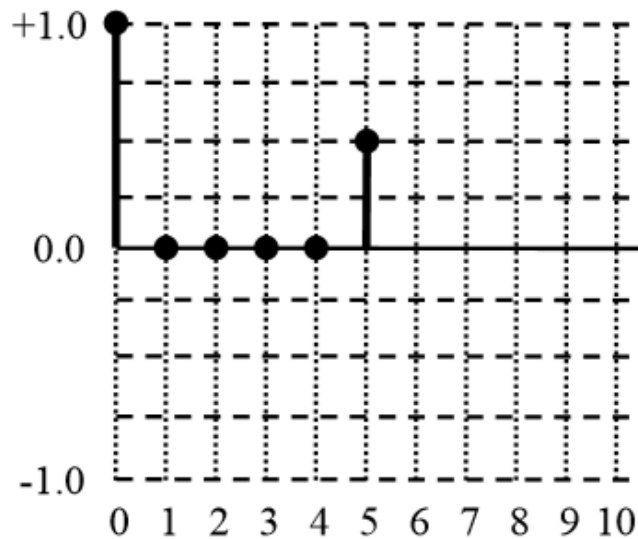


$y[n]$

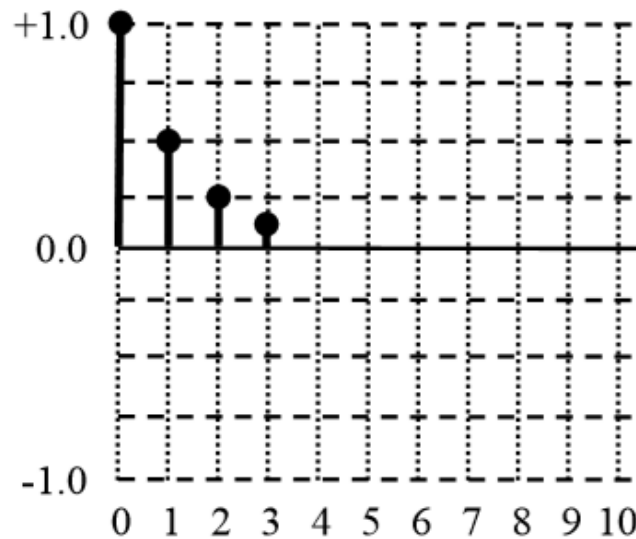
“non-overlapping
echo” IR

Example convolutions:

- Convolution is a commutative operation
 - That is: $x[n] * h[n] = h[n] * x[n] = y[n]$



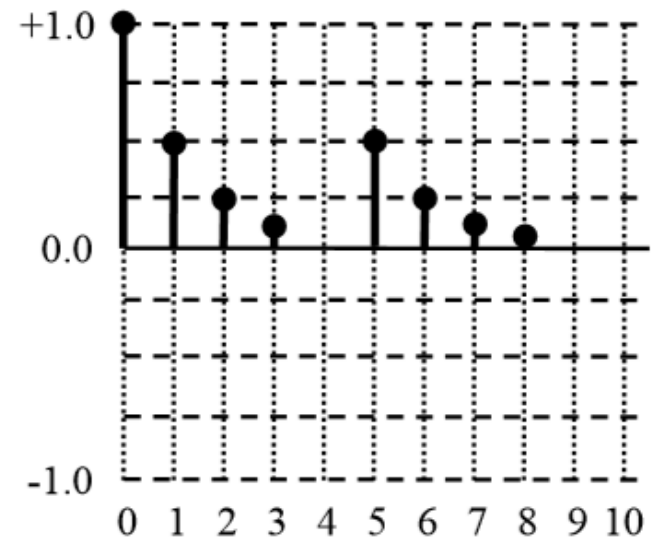
$h[n]$



$x[n]$

*

=



$y[n]$



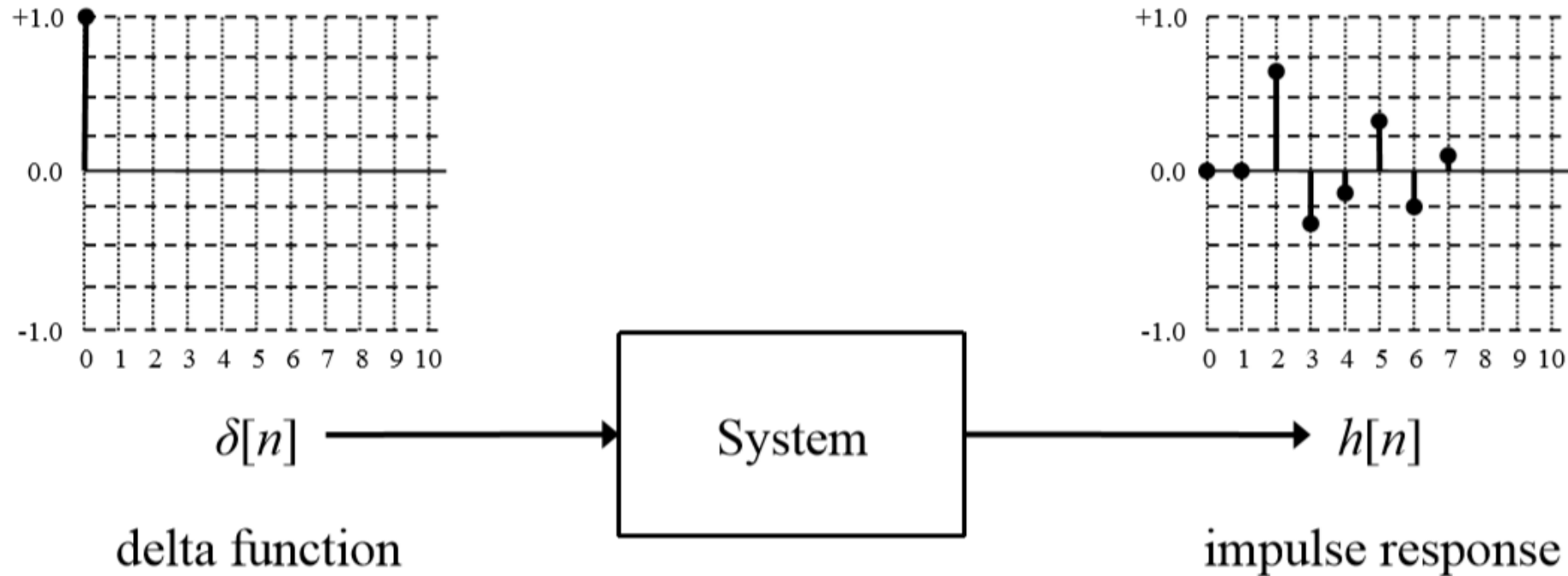
Time Domain Convolution

Time-Domain Convolution

- The signals $x[n]$ and $h[n]$ can be any arbitrary kind of signal
 - However, $x[n]$ is usually an input signal that you want to process
 - And, $h[n]$ is the impulse response (IR) of a system you want mode

Impulse Response

- The IR of a system is found by inputting a normalized impulse (delta function) into the system, and measuring or calculating the output

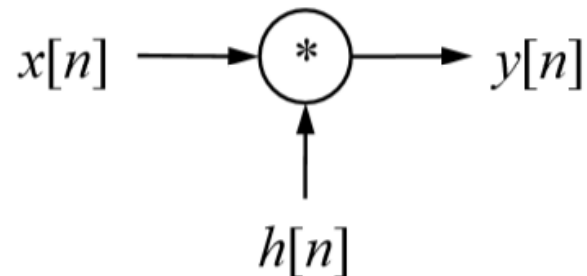
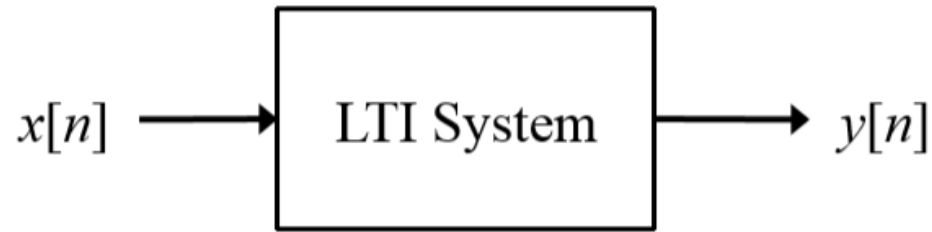


Model the System as it applied to signal change

- If the system is linear and time-invariant (LTI), the impulse response perfectly characterizes the system
 - That is, we know how the system will respond to any input it receives

Simulate the system as convolution

- We can thus simulate how a system would respond to an input signal by convolving the input signal with the system's IR
- That is, the following are equivalent:



Application: Concert Hall

Time-Domain Convolution

- The IR of a concert hall can be found by firing a starting pistol (an approximate delta function), and using a microphone to record to the system's response
 - Convolvering a “dry” recording of an instrument with the IR results in an output signal where it sounds like the instrument is playing in the hall
 - This is the basis for a convolution reverb digital effect

Convolution using the FFT

- Time-domain convolution can be very slow, especially if $h[n]$ is large
 - Is $O(N \times M)$, where N and M are the sizes of $x[n]$ and $h[n]$
- High-speed or FFT convolution uses the principle that convolution in the time domain corresponds to multiplication of spectra in the frequency domain

Convolution using the FFT

- This is the convolution theorem, and is expressed mathematically as:

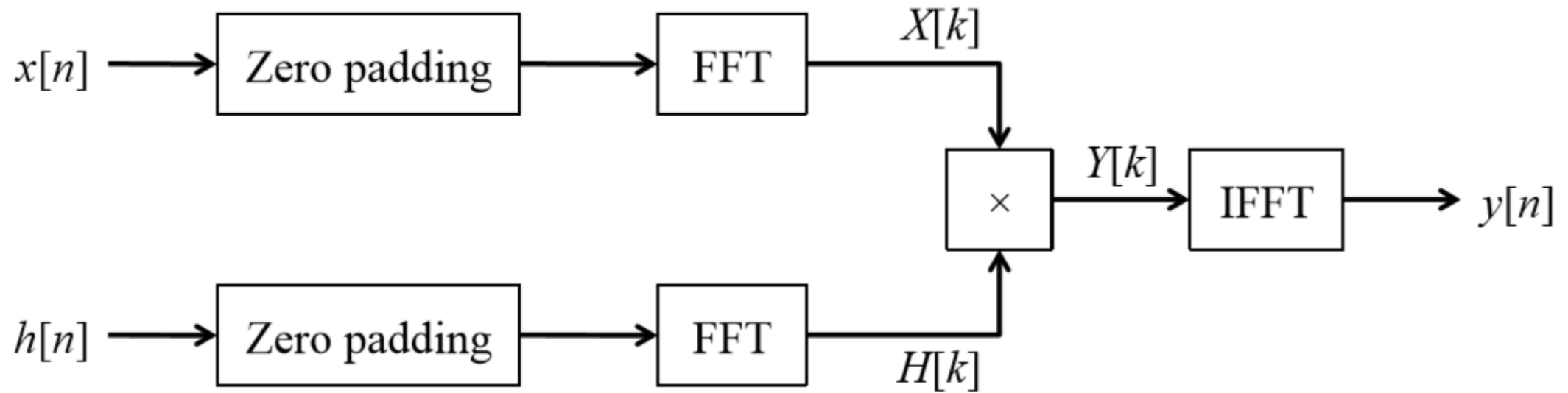
$$\begin{array}{ccc} & DFT & \\ x[n] * h[n] & \xleftrightarrow{\quad} & X[k] \cdot H[k] \\ & IDFT & \\ \text{Time domain} & \text{---} & \text{Frequency domain} \end{array}$$

- Where $X[k]$ and $H[k]$ are the spectra of $x[n]$ and $h[n]$:
 - $X[k] = \text{DFT}[x[n]]$
 - $H[k] = \text{DFT}[h[n]]$

Convolution using the FFT

- We can calculate the convolution by:
 - Transforming both $x[n]$ and $h[n]$ to the frequency domain using the FFT
 - Note: zero-padding will have to be used so that $x[n]$ and $h[n]$ have the same length, the length is a power of 2 (needed for the FFT), and is long enough to avoid circular convolution (wrap around)
 - Multiplying $X[k]$ by $H[k]$ point by point
 - Note: this will be complex multiplication (imaginary/real)
 - Converting the result back to the time domain using the inverse FFT (IFFT)

Convolution using the FFT



Convolution using the FFT

- Note: the output from either the FFTs or the IFFT (but NOT both) will have to be scaled by dividing by each data point by N , if the FFT/IFFT algorithm you are using doesn't do scaling
- E.g.

```
for (k = 0, i = 0; k < N; k++, i += 2) {  
    /* Scale the real and imaginary parts of a data point */  
    x[i] /= (double)N;  
    x[i+1] /= (double)N;  
}
```

Onward to ... next topic.

Jonathan Hudson
jwhudson@ucalgary.ca
<https://pages.cpsc.ucalgary.ca/~hudsonj/>



UNIVERSITY OF
CALGARY