Linear Constant Coefficient Difference Equations (LCCDE)

ECE 3640 Discrete-Time Signals and Systems

Jake Gunther

Linear Constant Coefficient Difference Equations (LCCDE)

$$\sum_{k=0}^{N} a[k]y[n-k] = \sum_{m=0}^{M} b[m]x[n-m]$$

(both sides look like convolution)

(output) (input)

To compute output, solve for $y[n]$.

$$a[0]y[n] = \sum_{m=0}^{M} b[m]x[n-m] - \sum_{k=1}^{N} a[k]y[n-k]$$

(recursive formula)

(usually $a[0] = 1$)

Note that current output $y[n]$ depends on:

- Current input: $x[n]$
- Past inputs: $x[n-1], \cdots, x[n-M]$
- Past outputs: $y[n-1], \cdots, y[n-N]$

(initial conditions)

Linear Constant Coefficient Difference Equations (LCCDE)

Forward recursion leads to causal output signal.

$$a[0]y[n] = \sum_{m=0}^{M} b[m]x[n-m] - \sum_{k=1}^{N} a[k]y[n-k]$$

Backward recursion leads to anti-causal output signal.

$$a[N]y[n-N] = \sum_{m=0}^{M} b[m]x[n-m] - \sum_{k=0}^{N-1} a[k]y[n-k]$$

- For LCCDE to represent LTI system, need zero initial conditions.

- All LTI systems representable by difference equations.

Linear Constant Coefficient Difference Equations (LCCDE)

$$\sum_{k=0}^{N} a[k]y[n-k] = \sum_{m=0}^{M} b[m]x[n-m]$$

Finite Impulse Response (FIR)

$a[0] = 1, N = 0, M < \infty$

$$y[n] = \sum_{k=0}^{M} b[m]x[n-m]$$

- No feedback
- Feed forward
- Convolution
- `conv` command in Matlab

Infinite Impulse Response (IIR)

$N > 0$

$$\sum_{k=0}^{N} a[k]y[n-k] = \sum_{m=0}^{M} b[m]x[n-m]$$

- Feedback
- Recursive
- Difference equation
- `filter` command in Matlab

Linear Constant Coefficient Difference Equations in Matlab

$$\sum_{k=0}^{N} a[k]y[n-k] = \sum_{m=0}^{M} b[m]x[n-m]$$

Finite Impulse Response (FIR)	Infinite Impulse Response (IIR)
$\mathbf{a} = 1$	$\mathbf{a} = [a_0, a_1, \cdots, a_N]$
$\mathbf{b} = [b_0, b_1, \cdots, b_M]$	$\mathbf{b} = [b_0, b_1, \cdots, b_M]$
$\mathbf{x} = [x_0, x_1, \cdots, x_L]$	$\mathbf{x} = [x_0, x_1, \cdots, x_L]$
$\mathbf{y} = [y_0, y_1, \cdots, y_L]$	$\mathbf{y} = [y_0, y_1, \cdots, y_L]$
$\mathbf{y} = \texttt{filter}(\mathbf{b}, 1, \mathbf{x})$	$\mathbf{y} = \texttt{filter}(\mathbf{b}, \mathbf{a}, \mathbf{x})$
$\mathbf{y} = \texttt{conv}(\mathbf{b}, \mathbf{x})$	

Example: Accumulator

Consider an LTI system having impulse response $u[n]$. The output is

$$y[n] = x[n] * u[n]$$

$$= \sum_{k=-\infty}^{\infty} u[k]x[n-k] = \sum_{k=0}^{\infty} x[n-k]$$

$$= \sum_{k=-\infty}^{\infty} x[k]u[n-k] = \sum_{k=-\infty}^{n} x[k] \quad \text{(accumulation)}$$

The output $y[n]$ is the accumulated sum of the current and all past inputs.

Example: Accumulator

Consider an LTI system having impulse response $u[n]$. The output is

$$y[n] = x[n] * u[n]$$

$$= \sum_{k=-\infty}^{\infty} u[k]x[n-k] = \sum_{k=0}^{\infty} x[n-k]$$

$$= \sum_{k=-\infty}^{\infty} x[k]u[n-k] = \sum_{k=-\infty}^{n} x[k] \quad \text{(accumulation)}$$

The output $y[n]$ is the accumulated sum of the current and all past inputs.

$$y[n] = \sum_{k=0}^{\infty} x[n-k]$$

$$\sum_{k=0}^{N} a[k]y[n-k] = \sum_{m=0}^{M} b[m]x[n-m]$$

$$a[0] = 1, N = 0$$

$$b[n] = 1 \text{ for all } n \geq 0, M = \infty$$

Example: Accumulator

$$y[n] = \sum_{k=-\infty}^{n} x[k]$$

$$= x[n] + \sum_{k=-\infty}^{n-1} x[k]$$

$$= x[n] + y[n-1]$$

$$y[n] - y[n-1] = x[n]$$

(diff. eq. representation is more compact)

$$\sum_{k=0}^{N} a[k]y[n-k] = \sum_{m=0}^{M} b[m]x[n-m]$$

$$a[0] = 1, a[1] = -1, N = 1$$

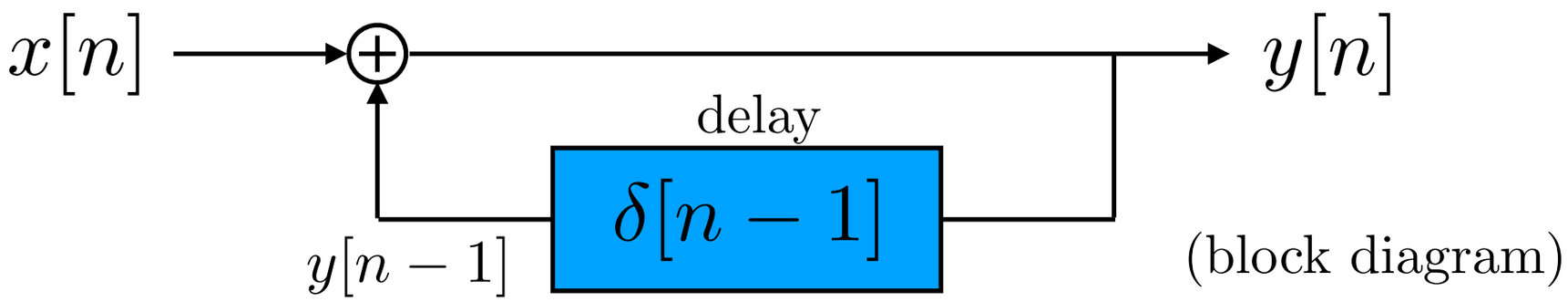
$$b[0] = 1, M = 0$$

```
y = filter(1,[1,-1],x)
```

(Matlab command)

Example: Accumulator

$$y[n] = x[n] + y[n-1]$$



(block diagram)

Example: Moving Average

(impulse response) $$b[n] = \begin{cases} \frac{1}{L}, & n = 0, 1, \cdots, L-1 \\ 0, & \text{otherwise} \end{cases}$$

$$= \left(\frac{1}{L}\right) (u[n] - u[n-L])$$

$$= \left(\frac{1}{L}\right) (\delta[n] - \delta[n-L]) * u[n]$$

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} x[n-k]$$

This representation uses L adds.

This means no feedback, i.e. FIR.

$$N = 0, \ \mathbf{a} = 1$$

$$M = L - 1, \ \mathbf{b} = \left[\frac{1}{L}, \cdots, \frac{1}{L}\right]$$

L length

`y = filter(b,a,x)` in Matlab

`y = conv(b,x)` in Matlab

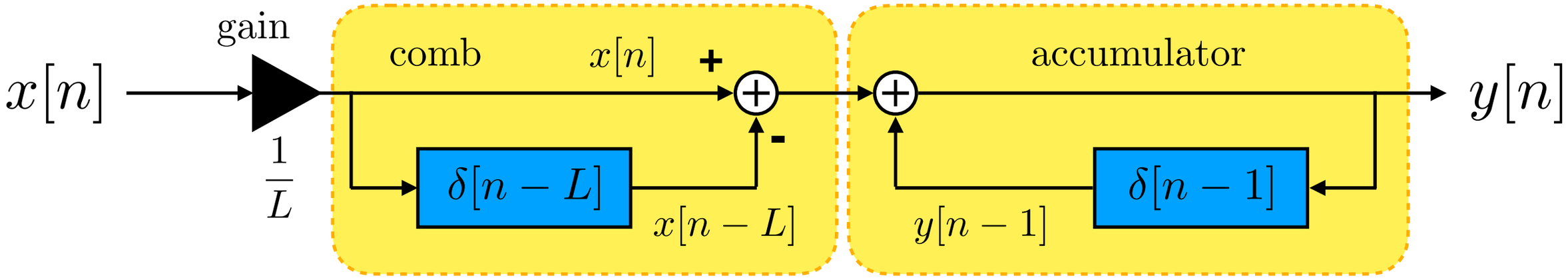
(What is the difference?)

Example: Moving Average

$$b[n] = \begin{cases} \frac{1}{L}, & n = 0, 1, \cdots, L-1 \\ 0, & \text{otherwise} \end{cases}$$

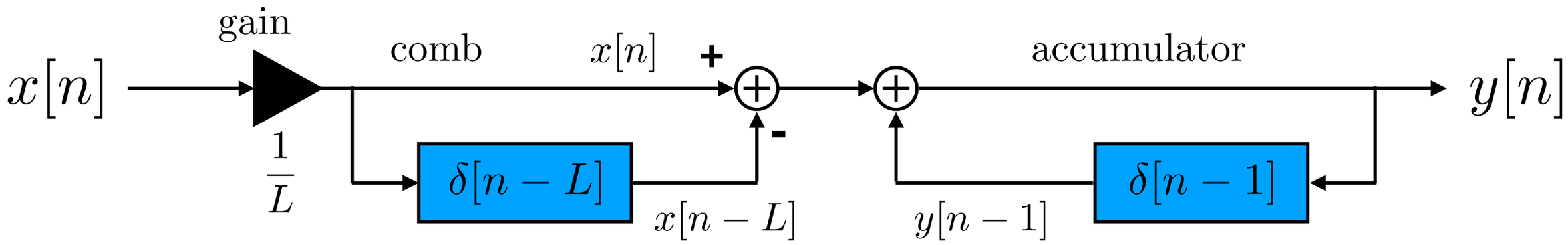
$$= \left(\frac{1}{L}\right)(u[n] - u[n-L])$$

$$= \left(\frac{1}{L}\right)(\delta[n] - \delta[n-L]) * u[n]$$



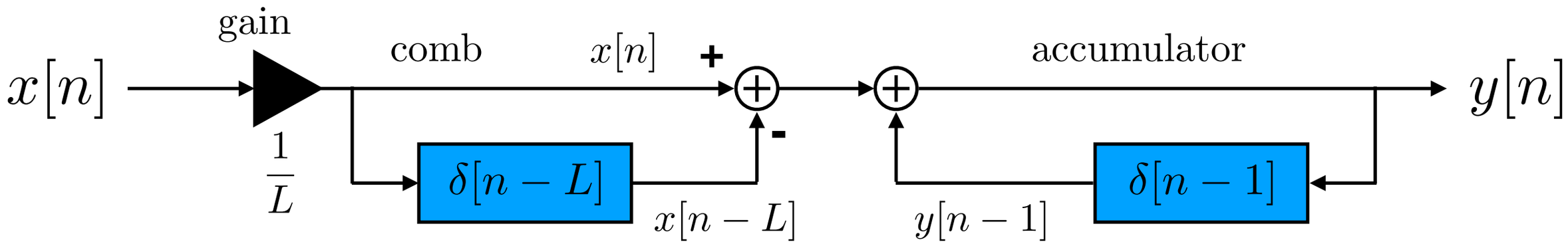
This representation uses less arithmetic: 2 adds.

Example: Moving Average



$$y[n] = y[n-1] + \frac{1}{L}\left(x[n] - x[n-L]\right)$$

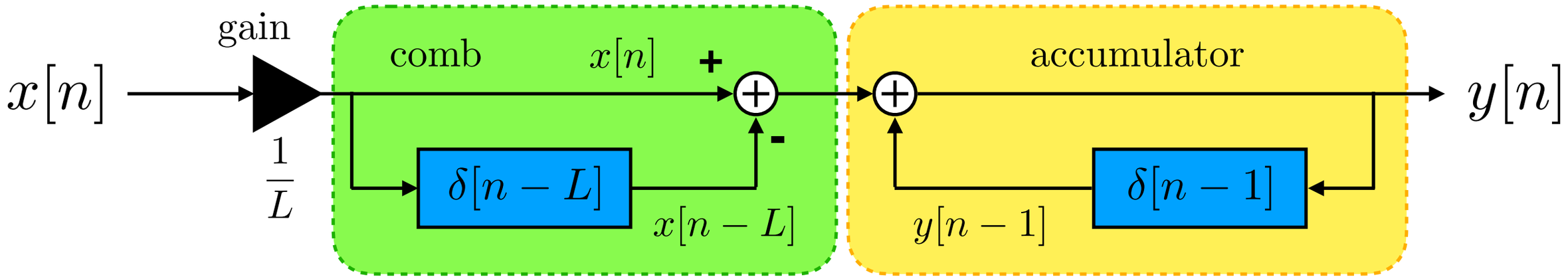
Example: Moving Average



$$y[n] = y[n-1] + \frac{1}{L}\left(x[n] - x[n-L]\right)$$

$$y[n] - y[n-1] = \frac{1}{L}\left(x[n] - x[n-L]\right)$$

Example: Moving Average



$$y[n] = y[n-1] + \frac{1}{L}\left(x[n] - x[n-L]\right)$$

$$y[n] - y[n-1] = \frac{1}{L}\left(x[n] - x[n-L]\right)$$

$$N = 1, \quad \mathbf{a} = [1, -1]$$

$$M = L - 1, \quad \mathbf{b} = \left[\frac{1}{L}, 0, \cdots, 0, -\frac{1}{L}\right]$$

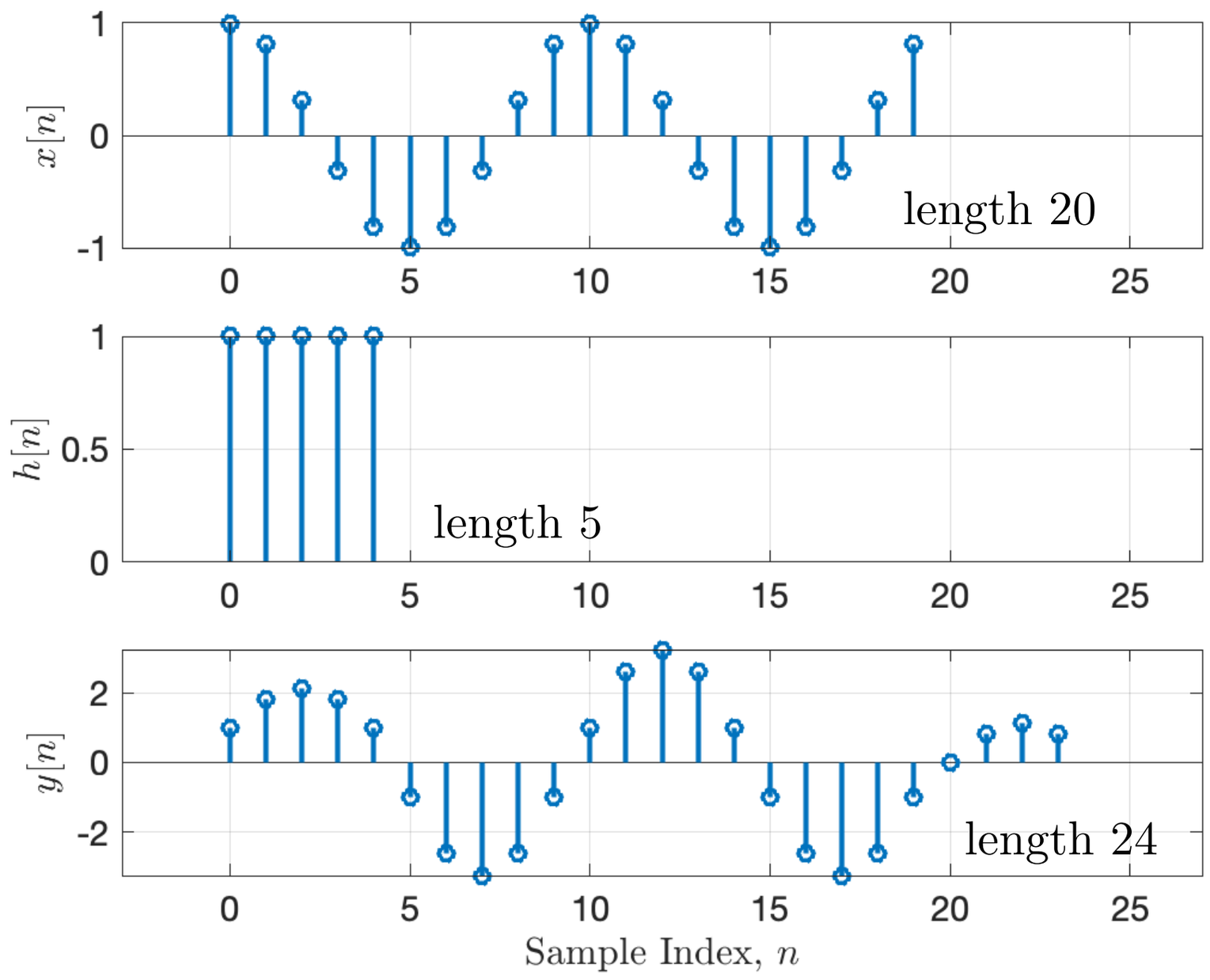
$L-1$ zeros

$L+1$ length

`y = filter(b,a,x)` in Matlab

Convolution and Filtering in Matlab

```
x = cos(2*pi*0.1*[0:19]);
h = ones(1,5);
y = conv(h,x);
```

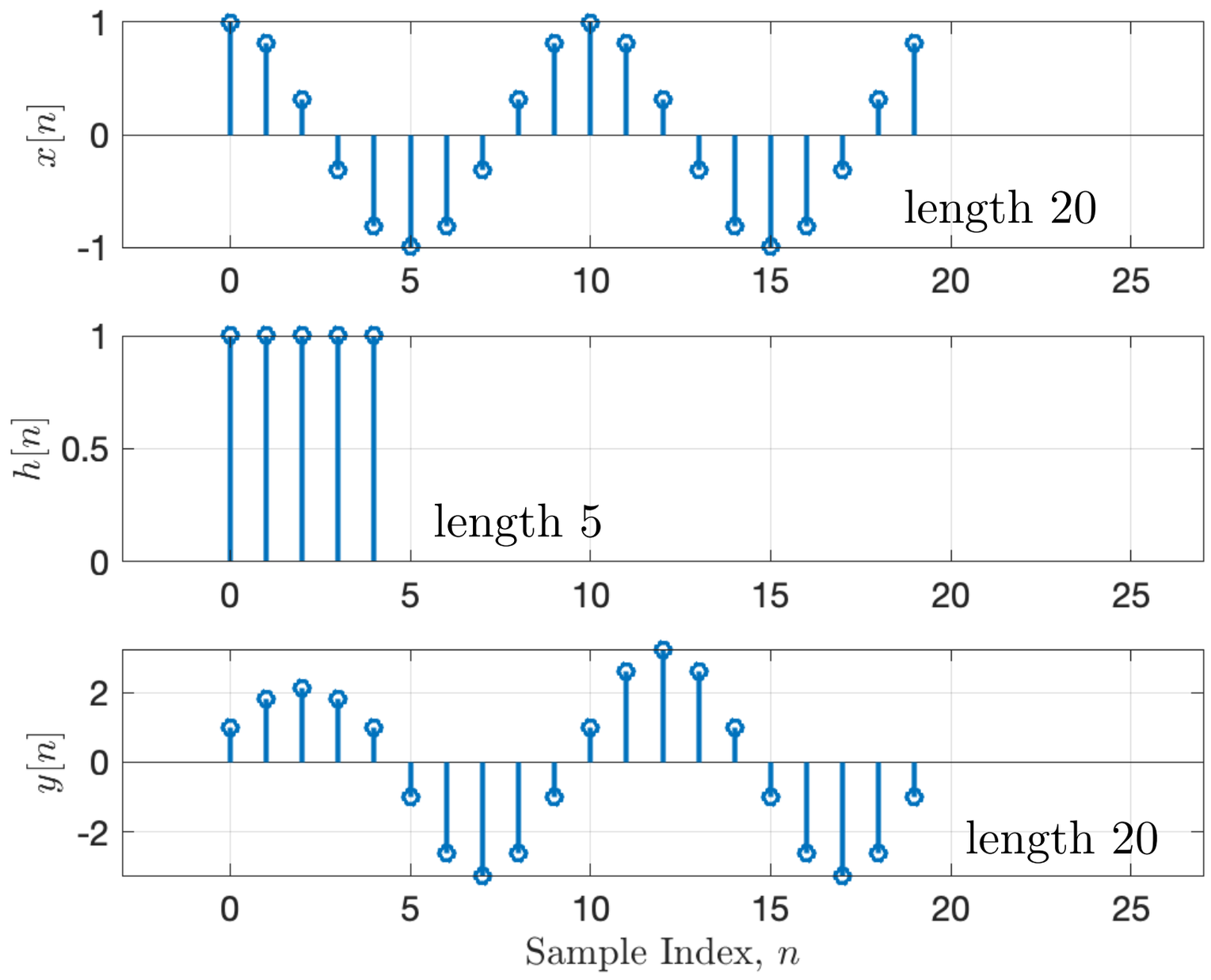


Length of convolution result:

$$L_y = L_x + L_h - 1$$

Convolution and Filtering in Matlab

```
x = cos(2*pi*0.1*[0:19]);
h = ones(1,5);
y = filter(h,1,x);
```



Length of filtering result:

$$L_y = L_x$$

Convolution and Filtering in Matlab

```
x = [cos(2*pi*0.1*[0:19]),zeros(1,4)];
h = ones(1,5);
y = filter(h,1,x);
```

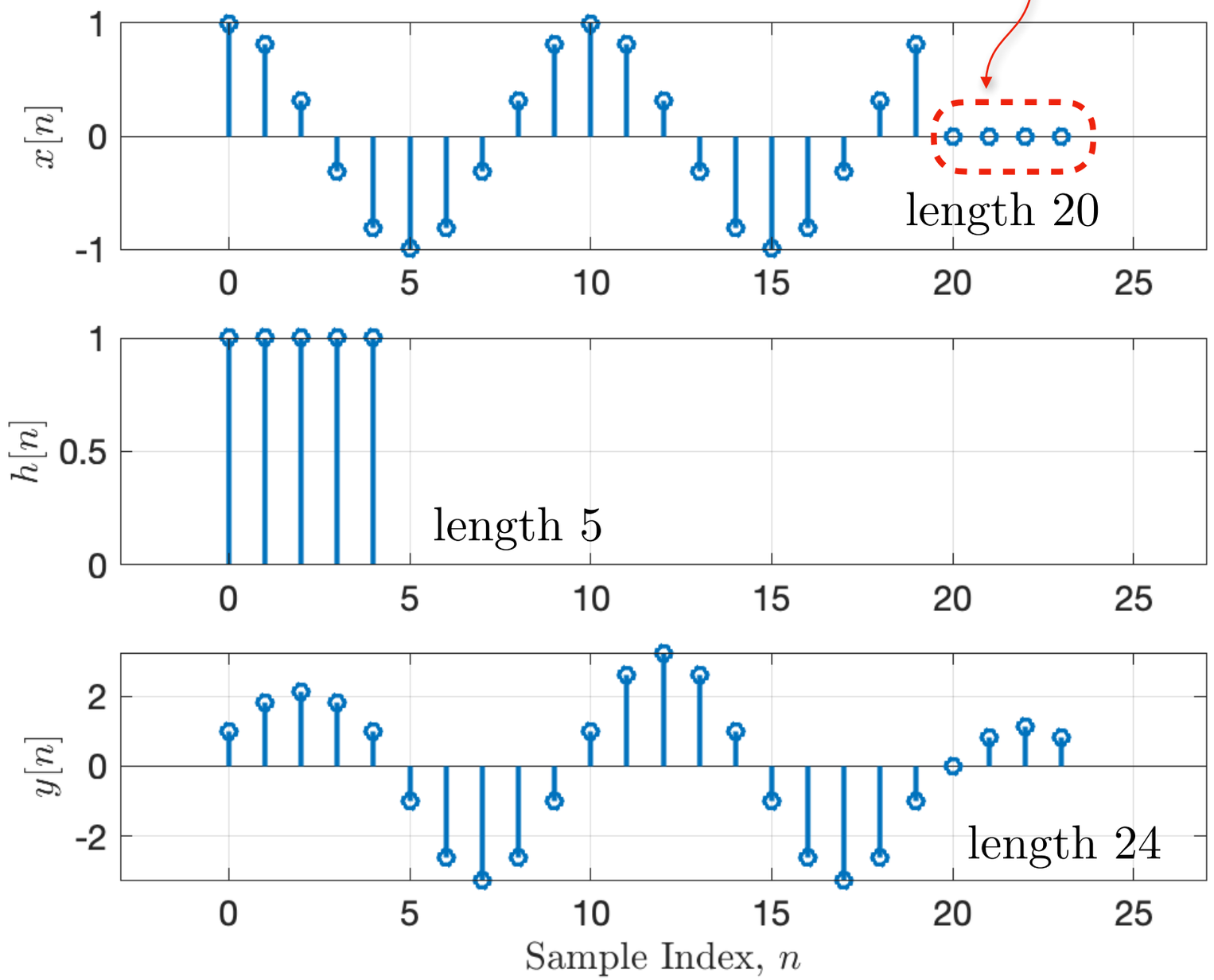
(zero padding)

Make filtering perform convolution by zero padding the input.

Zero pad by $L_h - 1$.

Length of convolution result:

$$L_y = L_x + L_h - 1$$



Impulse Response

Q: Given a difference equation, how do you compute the impulse response?

$$a[0]y[n] = \sum_{m=0}^{M} b[m]x[n-m] - \sum_{k=1}^{N} a[k]y[n-k]$$

A: Put in an impulse and see what comes out.

$$\mathbf{a} = [a_0, a_1, \cdots, a_N]$$

$$\mathbf{b} = [b_0, b_1, \cdots, b_M]$$

$$\boldsymbol{\delta} = [1, 0, 0, \cdots, 0] \quad \text{(Kronecker delta sequence)}$$

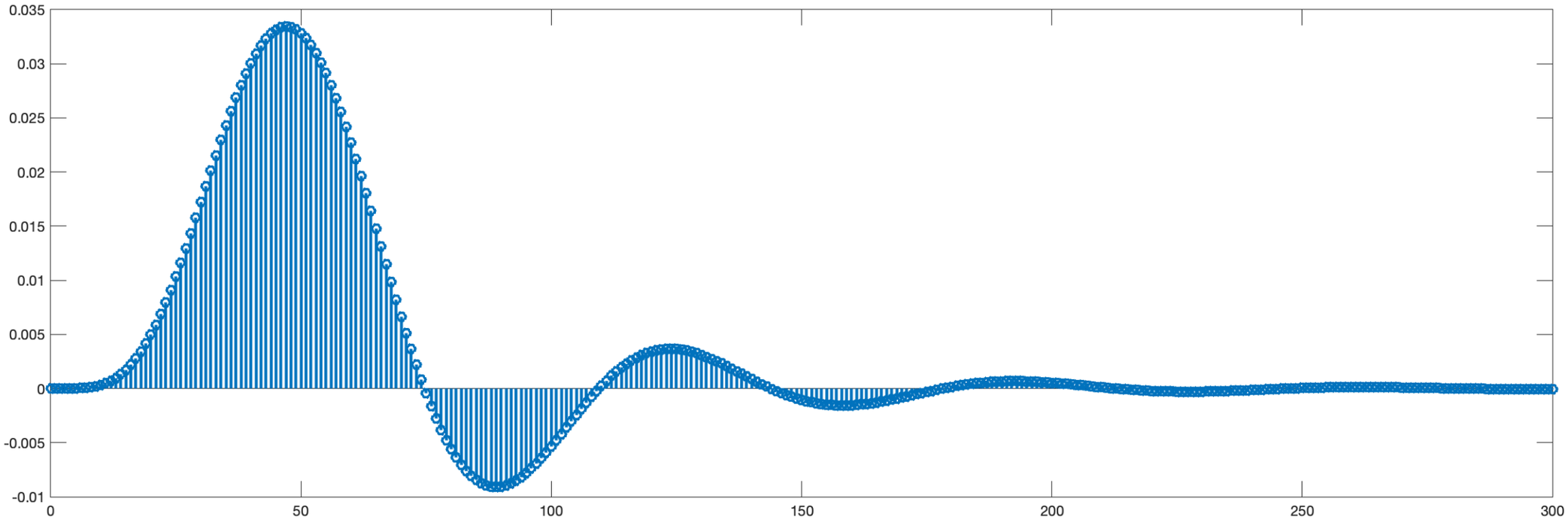
$$\mathbf{h} = [h_0, h_1, \cdots, h_L] \quad \text{(Impulse response)}$$

$$\mathbf{h} = \texttt{filter}(\mathbf{b}, \mathbf{a}, \boldsymbol{\delta})$$

Note: This only computes the first L samples of the impulse response. To compute the entire impulse response sequence, we will need the z-transform.

Butterworth Filter Impulse Response

```
[b,a]=butter(6,0.03); % LPF
h = filter(b,a,[1,zeros(1,300)]);
stem([0:300],h,'LineWidth',2);
```



- This gives the first 301 samples of the impulse response, which continues forever.
- At some point the impulse response amplitude becomes negligible.
- Truncation gives an approximate FIR filter.
- Difference equation realization more computationally efficient than convolution with truncated impulse response.