Lecture I213E – Class 12

Discrete Signal Processing

Sakriani Sakti

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY
1990

Course Materials

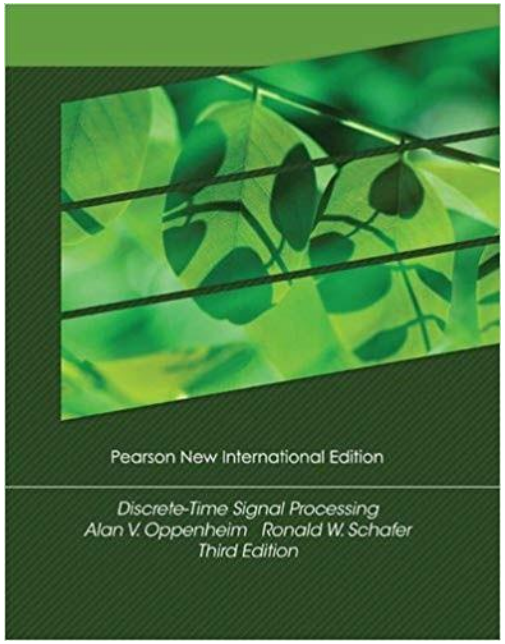
- **Materials**

  → Lecture notes will be uploaded before each lecture
  https://jstorage-2018.jaist.ac.jp/s/PGXRrC7iFmN2FWo
  Pass: dsp-i213e-2022
  (Slide Courtesy of Prof. Nak Young Chong)

- **References**

  → Chi-Tsong Chen:
  Linear System Theory and Design, 4th Ed.,
  Oxford University Press, 2013.

  → Alan V. Oppenheim and Ronald W. Schafer:
  Discrete-Time Signal Processing, 3rd Ed.,
  Pearson New International Ed., 2013.



Related Courses & Prerequisite

- **Related Courses**
  - → I212 Analysis for Information Science
  - → I114 Fundamental Mathematics for Information Science

- **Prerequisite**
  - → None

Evaluation

- **Viewpoint of evaluation**

  → Students are able to understand:

  - Basic principles in modeling and analysis of linear time-invariant systems
  - Applications of mathematical methods and tools to different signal processing problems.

- **Evaluation method**

  → Homework, term project, midterm exam, and final exam

- **Evaluation criteria**

  → Homework/labs (30%), term project (30%)
  midterm exam (15%), and final exam (25%)

Contact

- **Lecturer**

  → Sakriani Sakti

- **TA**

  Tutorial hours & Term project

  → WANG Lijun (s2010026)

  → TANG Bowen (s2110411)

  Homework

  → PUTRI Fanda Yuliana (s2110425)

- **Contact Email**

  → dsp-i213e-2022@ml.jaist.ac.jp

Schedule

- **December 8th, 2022 – February 9th, 2023**

- **Lecture Course Term 2-2**

  → Tuesday 9:00 — 10:40

  → Thursday 10:50 — 12:30

- **Tutorial Hours**

  → Tuesday 13:30-15:10

Schedule

Dec

Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Jan

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	✖	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	✖	25	26	27	28
29	30	31				

Feb

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				

Lecture:
Tuesday 9:00 — 10:40
Thursday 10:50 — 12:30

Additional Lecture:
Monday 17:10 — 18:50

Tutorial:
Tuesday 13:30 — 15:10

Course review &
term project evaluation
(on tutorial hours)

Midterm & final exam
Thursday 10:50 — 12:30

Syllabus

Class	Date	Lecture Course Tue 9:00 — 10:40 / Thr 10:50 — 12:30	Tutorial Hours Tue 13:30 — 15:10
1	12/08	Introduction to Linear Systems with Applications to Signal Processing	
2	12/13	State Space Description	○
3	12/15	Linear Algebra	
4	12/20	Quantitative Analysis (State Space Solutions) and Qualitative Analysis (Stability)	○
5	12/22	Discrete-time Signals and Systems	
X	01/05		
6	01/10	Discrete-time Fourier Analysis	
7	01/10*	Review of Discrete-time Linear Time-Invariant Signals and Systems (on Tutorial Hours)	
	01/12	Midterm Exam	
8	01/17	Sampling and Reconstruction of Analog Signals	○
9	01/19	z-Transform	
X	01/24		○
10	01/31	Discrete Fourier Transform	○
11	02/02	FFT Algorithms	
12	02/06	Implementation of Digital Filters	
13	02/07	Digital Signal Processors and Design of Digital Filters	
14	02/07*	Review of the Course and Term Project Evaluation (on Tutorial Hours)	
	02/09	Final exam	

Class 12
Implementation of Digital Filters

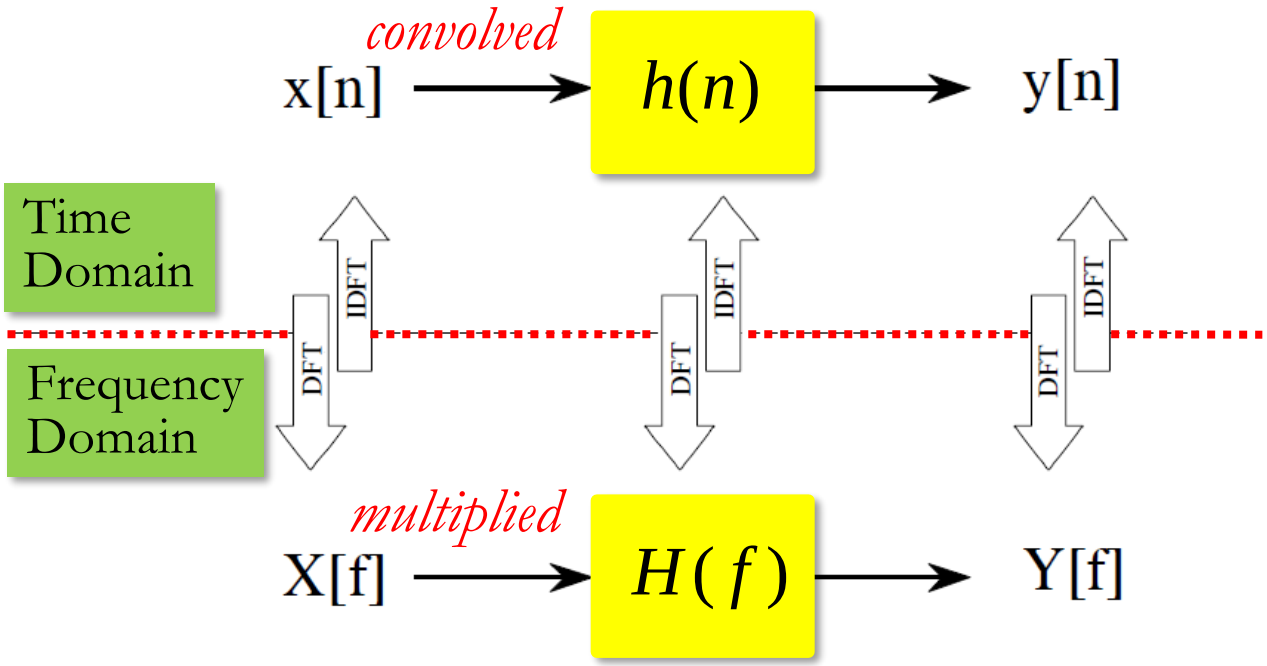
Linear Time-invariant (LTI) / Linear Shift-invariant (LSI) System

LTI/LSI System

- **LTI/LSI**

  → They are basically equivalent: the linear time invariant systems refers to an analog system and shift-invariant system refers to a discrete-time system.

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



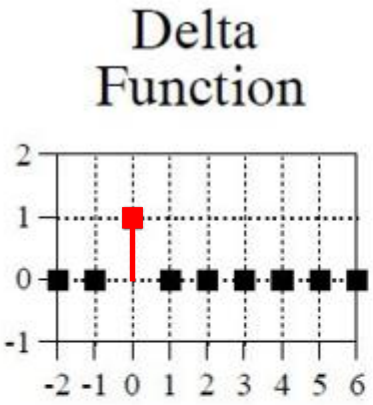
$$Y[f] = X[f]H[f]$$
$$Y[\omega] = X[\omega]H[\omega]$$

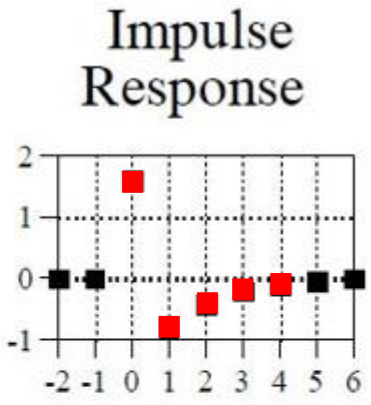
Impulse Input and Response

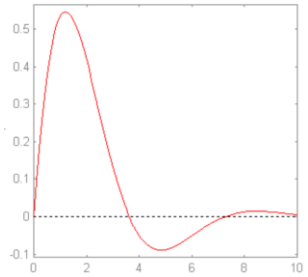
- **Impulse (Delta Function)**

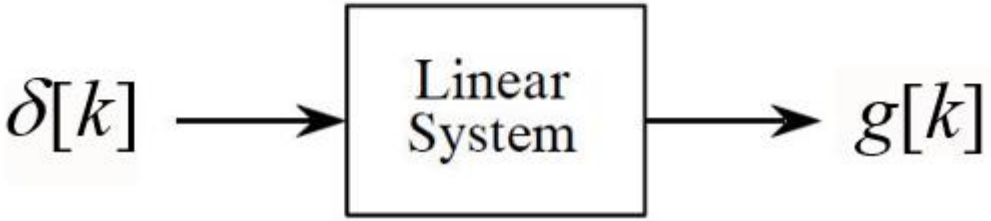
  → Impulse input

  → Impulse response









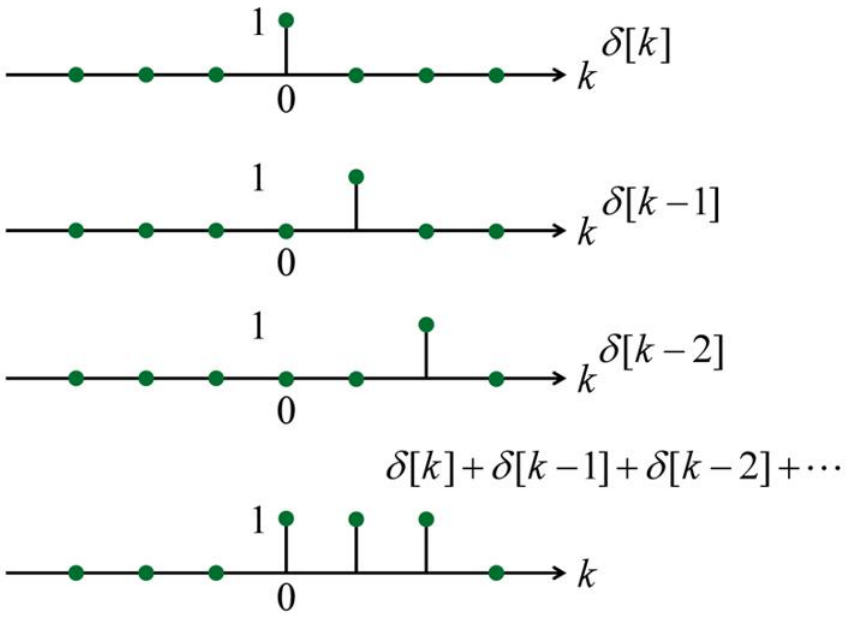
The impulse response is the signal that exits a system when a delta function (unit impulse) is the input

Step Function and Response

Step Function

→ Step function is a running sum of time-delayed impulses:

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



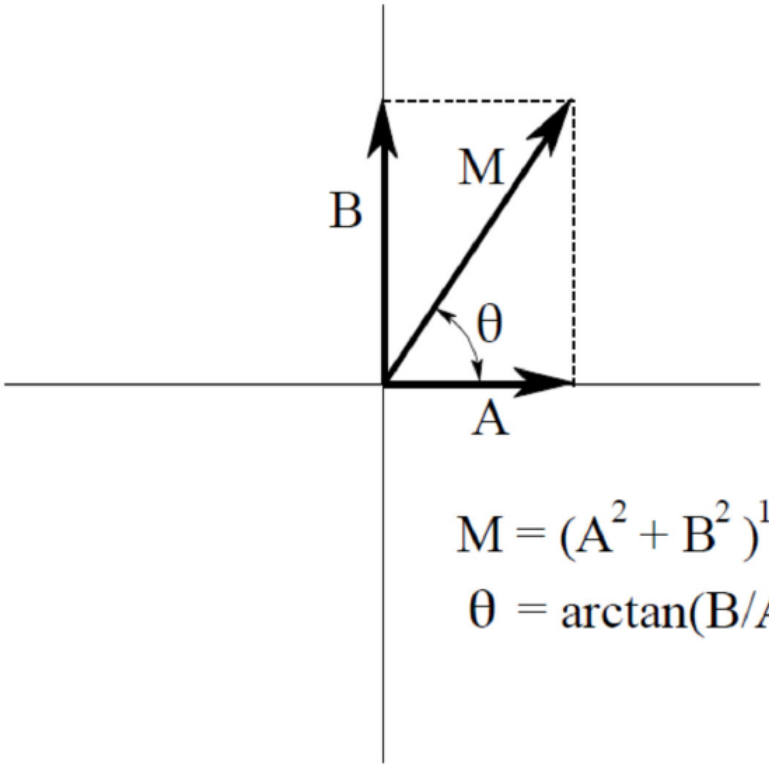
Step Response

→ step response is a running sum of time-delayed impulse responses:

$$s(n) = \sum_{k=0}^{\infty} h(n-k) \quad \Rightarrow \quad \boxed{h(n) = s(n) - s(n-1)}$$

Frequency Response: Magnitude & Phase

- **Rectangular-to-Polar Conversion**



$$M = (A^2 + B^2)^{1/2}$$

$$\theta = \arctan(B/A)$$

$$MagX[k] = (\mathrm{Re}\,X[k]^2 + \mathrm{Im}\,X[k]^2)^{1/2}$$

$$PhaseX[k] = \arctan\left(\frac{\mathrm{Im}\,X[k]}{\mathrm{Re}\,X[k]}\right)$$

$$\mathrm{Re}\,X[k] = MagX[k]\cos(PhaseX[k])$$

$$\mathrm{Im}\,X[k] = MagX[k]\sin(PhaseX[k])$$

Magnitude Distortion

Magnitude Distortion

→ If the system provides unequal amount of amplification to different frequency components available in input signal

$x(t) = \sin \omega_1 t + \sin \omega_2 t$

$\omega_1 \neq \omega_2$

$y(t) = k_1 \sin \omega_1 t + k_2 \sin \omega_2 t$

$k_1 \neq k_2$

Phase Distortion

Phase Distortion

→ If the system provides unequal amount of time delays to different frequency components available in input signal

$x(t) = \sin \omega_1 t + \sin \omega_2 t$

$\omega_1 \neq \omega_2$

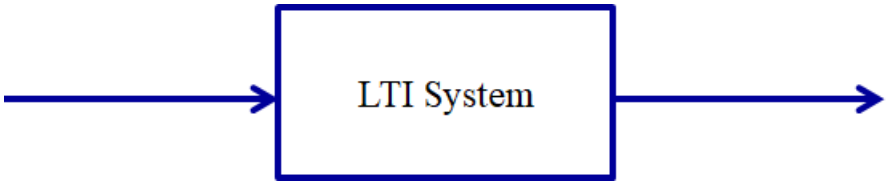
$y(t) = \sin[\omega_1(t - t_1)] + \sin[\omega_2(t - t_2)]$

$t_1 \neq t_2$

Distortionless LTI System

- **Distortionless**

  → No magnitude and phase distortion



$x(t) = \sin \omega_1 t + \sin \omega_2 t$ $\quad$ $y(t) = k_1 \sin[\omega_1 (t - t_1)] + k_2 \sin[\omega_2 (t - t_2)]$

$\omega_1 \neq \omega_2$ $\quad$ $k_1 = k_2$ $\quad$ $t_1 = t_2$

$$y(t) = kx(t - t_0)$$

$$Y(j\omega) = kX(j\omega)e^{-j\omega t_0}$$

Group Delay & Phase Delay

Phase Response

→ The relationship between the phase of input signal and processed output signal

$$Y(j\omega) = kX(j\omega)e^{-j\omega t_0}$$

$$\frac{Y(j\omega)}{X(j\omega)} = H(j\omega) = ke^{-j\omega t_0}$$

Magnitude: $|H(j\omega)| = k$

Phase: $\angle H(j\omega) = -\omega t_0$

Group Delay

→ The time delay of amplitude envelopes of these sinusoidal components
(All the multiple frequency components experience the same delay)

$$y(t) = \left|H(i\omega)\right| a(t - \tau_g) \cos\left(\omega(t - \tau_\phi) + \theta\right) \Rightarrow \tau_g = -\frac{\mathrm{d}\angle H(j\omega)}{d\omega}$$

Phase Delay

→ The time delay of phase of various sinusoidal components present in a signal

$$y(t) = \left|H(i\omega)\right| a(t - \tau_g) \cos\left(\omega(t - \tau_\phi) + \theta\right) \Rightarrow \tau_\varphi = -\frac{\angle H(j\omega)}{\omega}$$

Group Delay & Phase Delay

▪ Distortionless LTI System

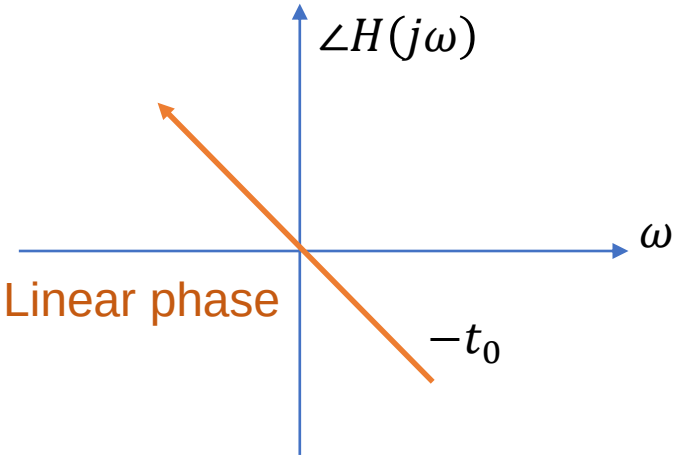
$$Y(j\omega) = kX(j\omega)e^{-j\omega t_0}$$

$$\frac{Y(j\omega)}{X(j\omega)} = H(j\omega) = ke^{-j\omega t_0}$$

→ Magnitude: $|H(j\omega)| = k$

→ Phase: $\angle H(j\omega) = -\omega t_0 = -t_0.\omega$+0 ➡ Phase delay:

$$t_0 = -\frac{\angle H(j\omega)}{\omega}$$



In Distortionless LTI system

$$-\frac{\angle H(j\omega)}{\omega} = -\frac{d\angle H(j\omega)}{d\omega}$$

→ Slope: $\frac{d\angle H(j\omega)}{d\omega} = -t_0$ ➡ Group delay:

$$t_0 = -\frac{d\angle H(j\omega)}{d\omega}$$

Digital Filters

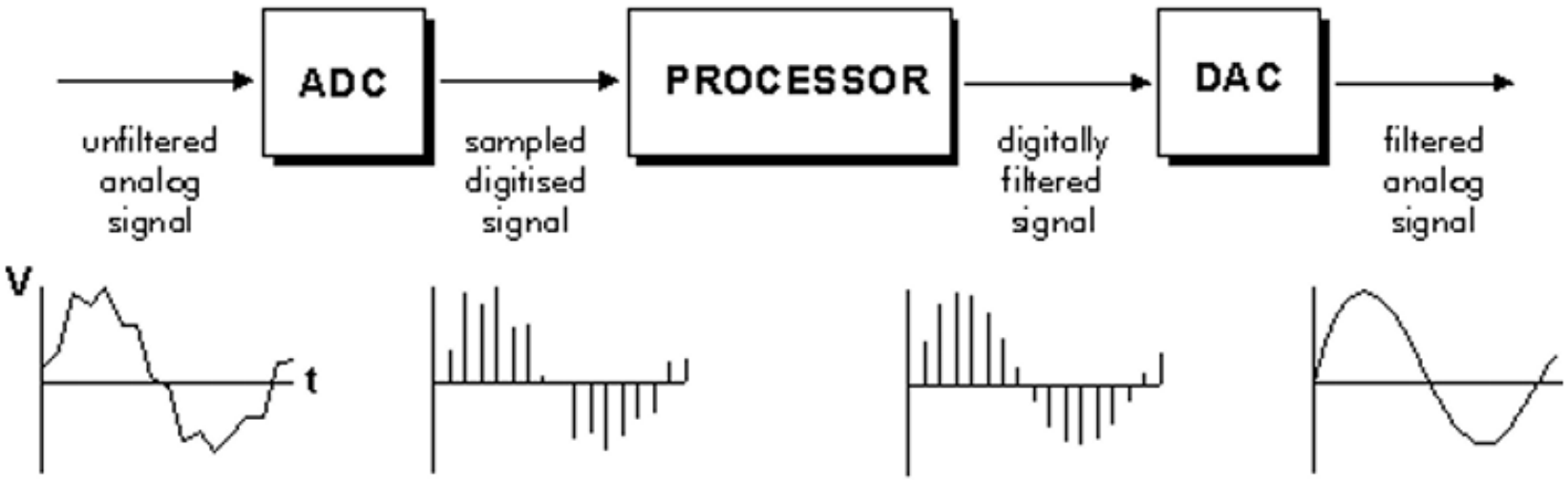
Digital Filters

- **Usage**

  (1) Signal separation: separation of signals that have been combined-contaminated with interference, noise, or other signals

  (2) Signal restoration: restoration of signals that have been distorted

  Every linear filter has an impulse response and a frequency response



How Information is Represented in Signals

▪ Information represented in Time-domain

→ When something occurs and what the amplitude of the occurrence is
→ Each sample contains information that is interpretable without reference to any other sample.
→ Example: The light output from the sun. The light output is measured and recorded once each second.

➡ Step Response: how information represented in the *time domain* is being modified by the system.

▪ Information represented in Frequency-domain

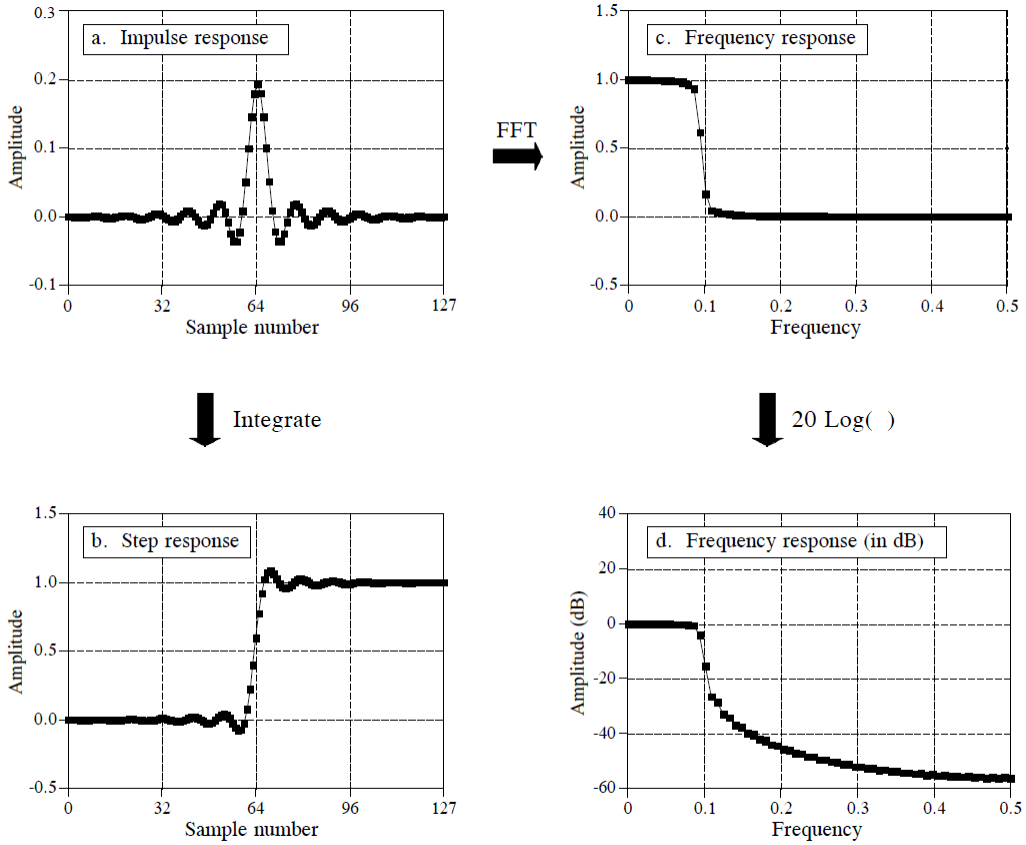
→ Many things show periodic motion
→ A single sample, in itself, contains no information about the periodic motion
→ The information is contained in the relationship between many points in the signal.
→ Example: the pendulum of a grandfather clock swings back and forth

➡ Frequency Response: how information represented in the frequency domain is being changed.

Filter Basic

Filter Basic Time-domain Filter

→ Time-domain Filter: Step Response
→ Frequency-domain Filter: Frequency Response

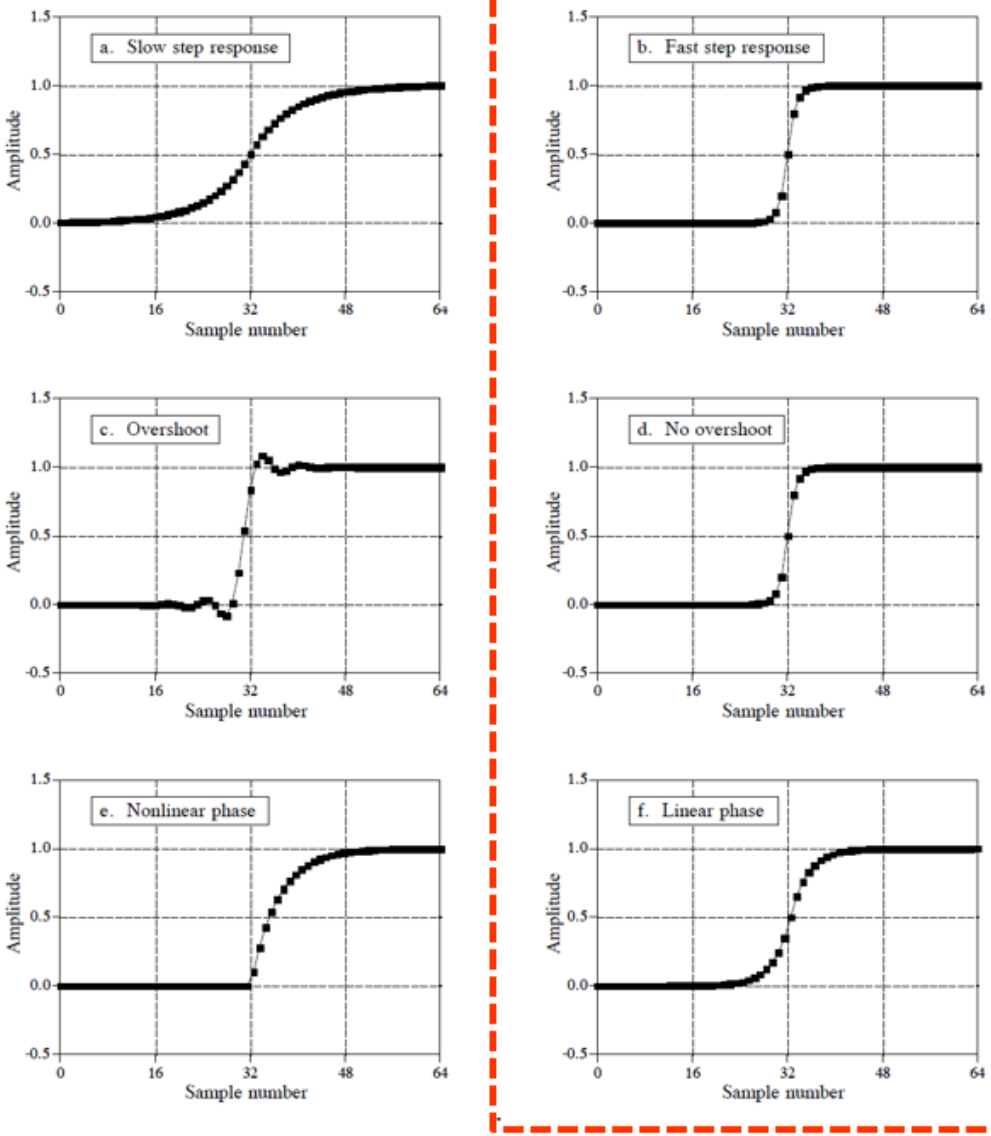


It is not possible to optimize a filter for both applications.
Good performance in the time domain results in poor performance in the frequency domain, and vice versa.

Time Domain Parameters

POOR

GOOD



distinguish events: transition speed (rise time)

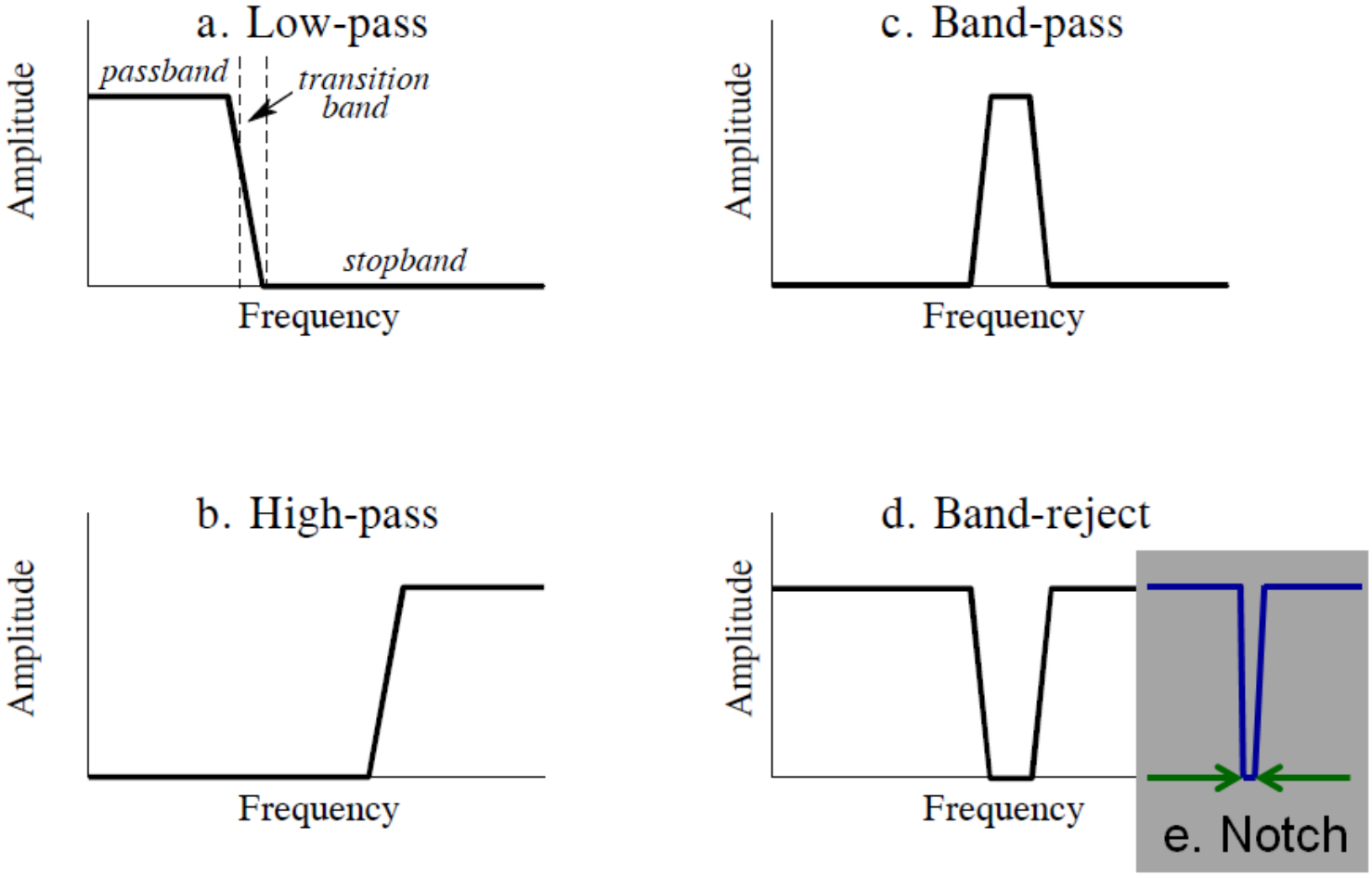
changes in the amplitude of samples

symmetry between the top and bottom halves of the step

Frequency Domain Filters

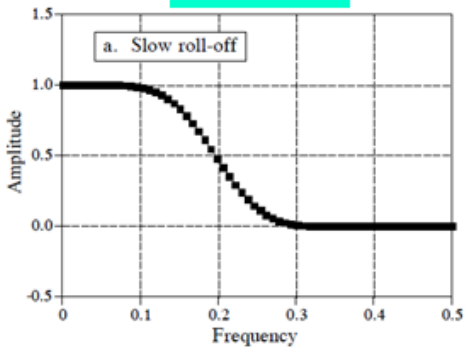
- **Usage**

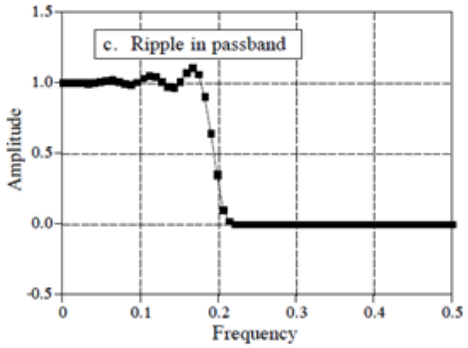
→ Pass certain frequencies (the passband), while blocking others (the stopband).

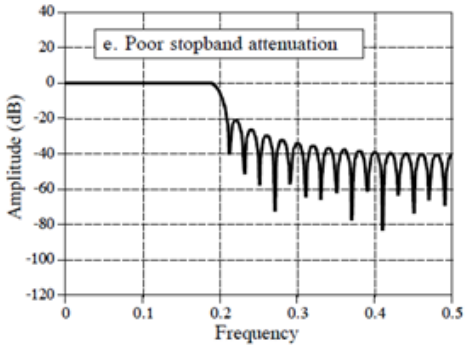


Frequency Domain Parameters: Low-Pass

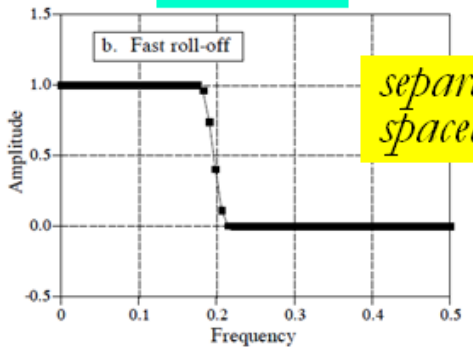
POOR



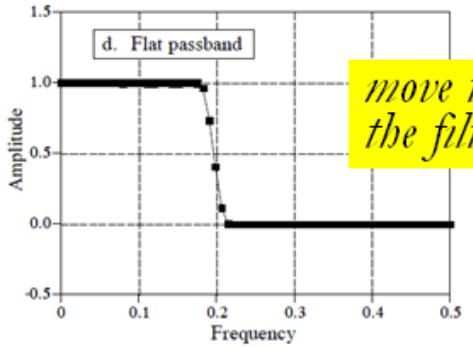




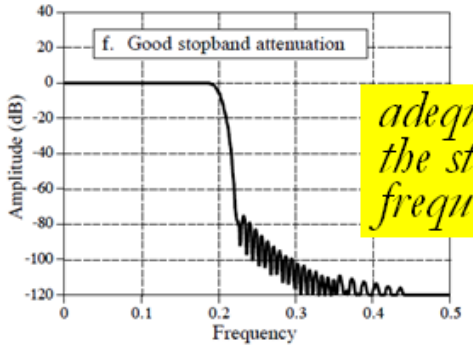
GOOD



separate closely spaced frequencies



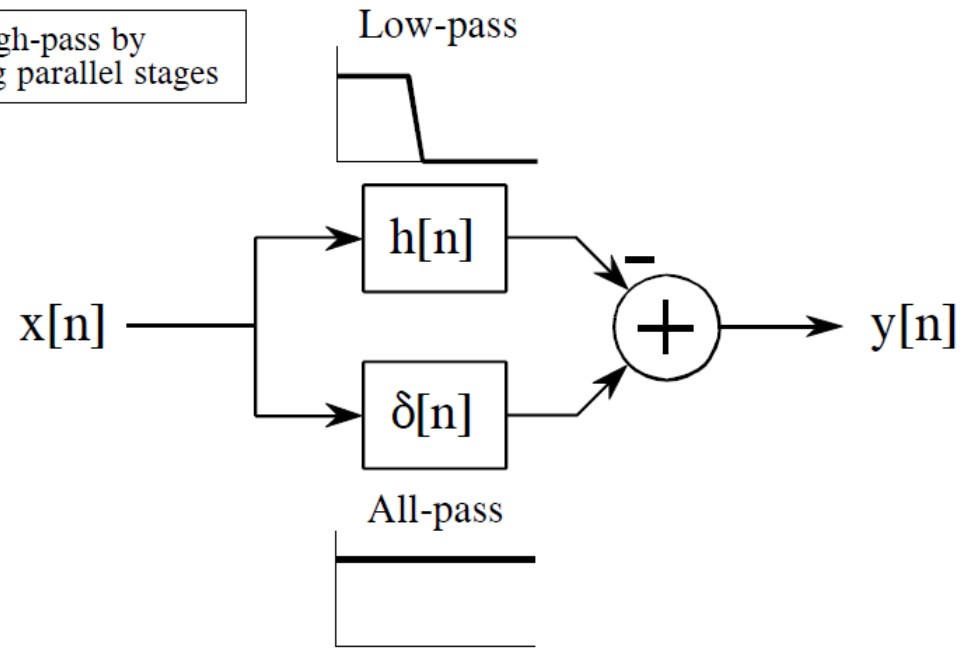
move through the filter unaltered



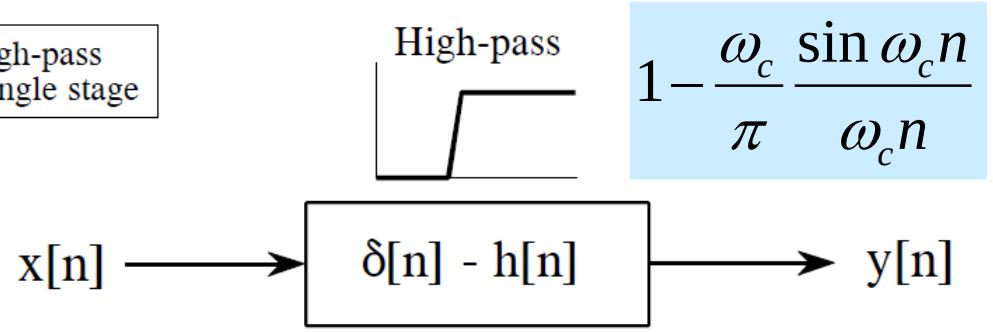
adequately block the stopband frequencies

High-Pass: Converting from Low-Pass

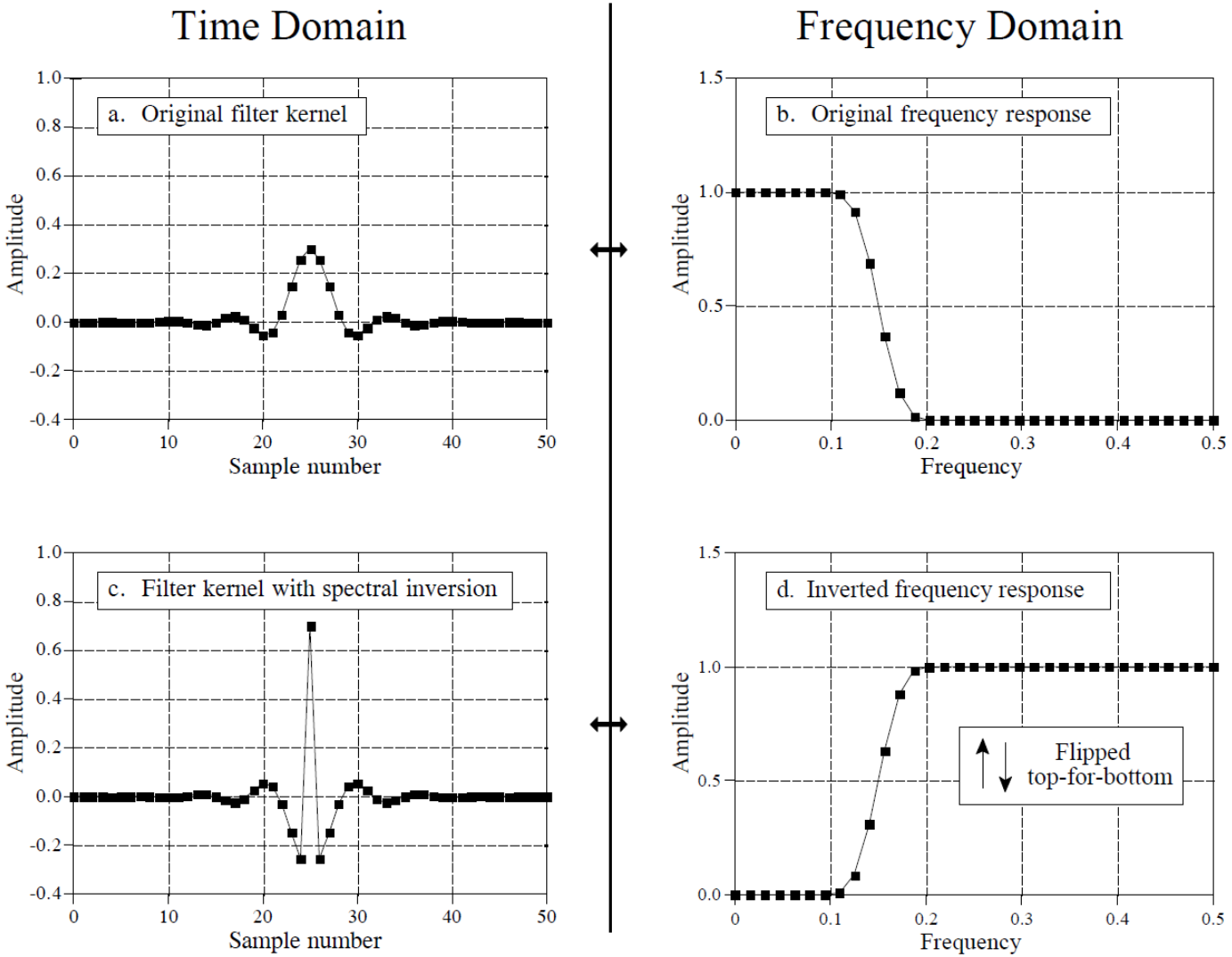
a. High-pass by adding parallel stages



b. High-pass in a single stage

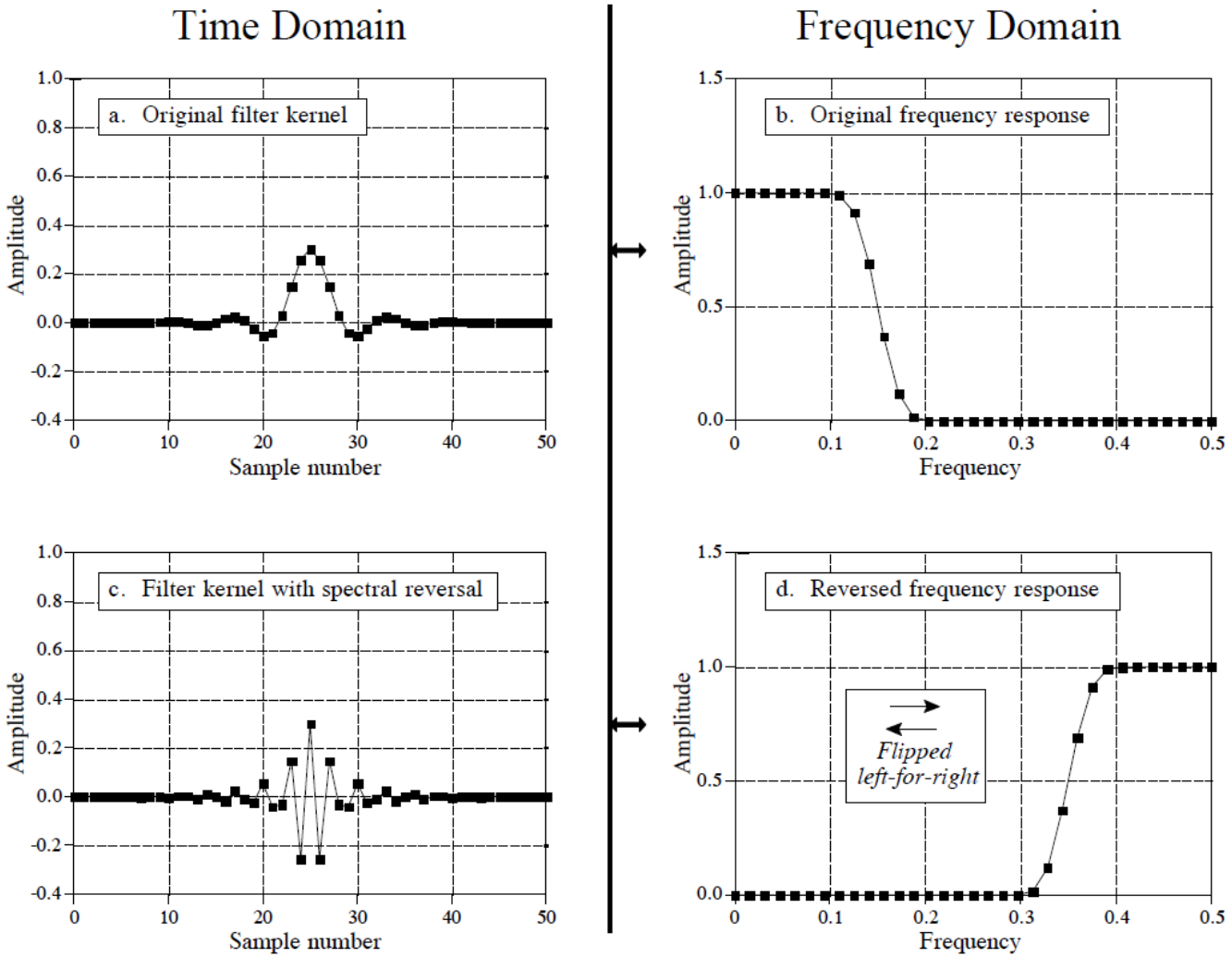


High-Pass: Converting from Low-Pass I



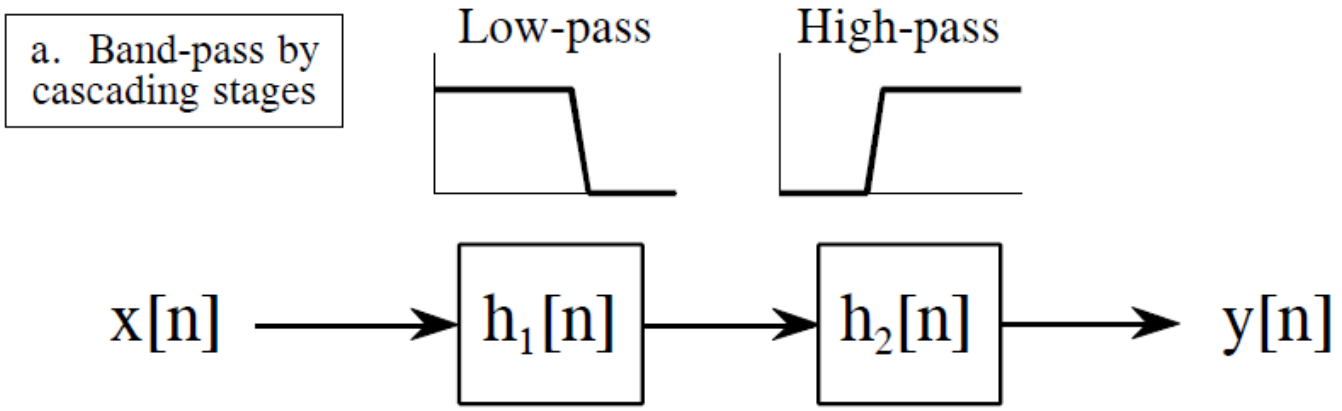
flips the frequency response top-for-bottom, changing the passbands into stopbands, and the stopbands into passbands.

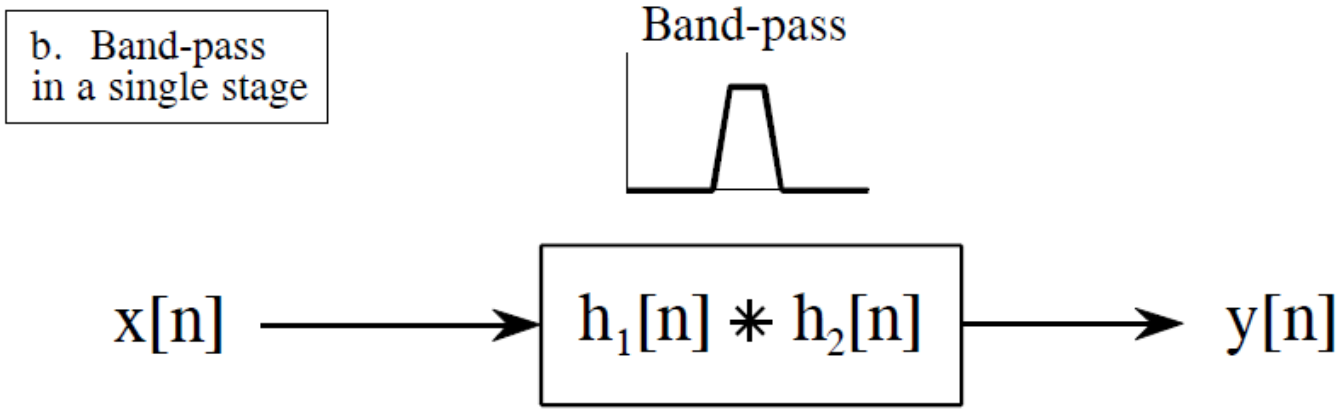
High-Pass: Converting from Low-Pass II



flips the frequency response left-for-right

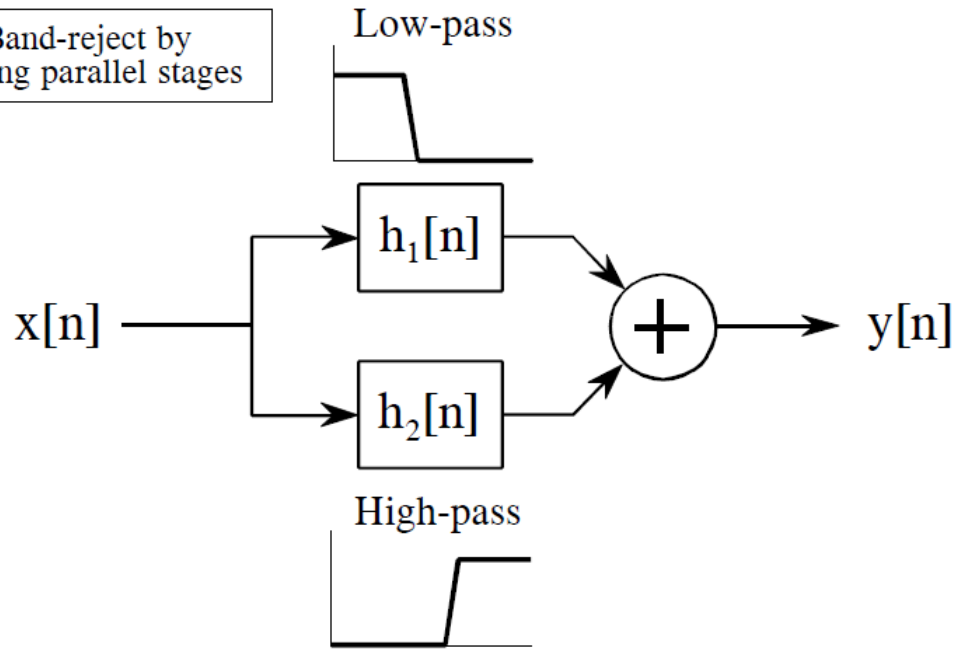
Band-Pass: Low-Pass + High-Pass



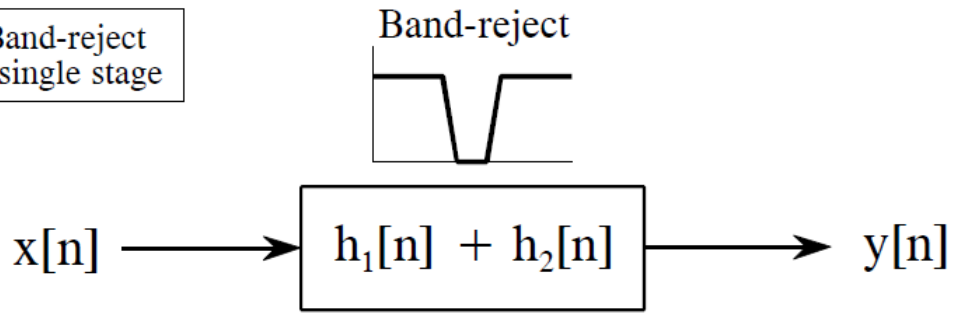


Band-Reject: Low-Pass + High-Pass

a. Band-reject by adding parallel stages



b. Band-reject in a single stage

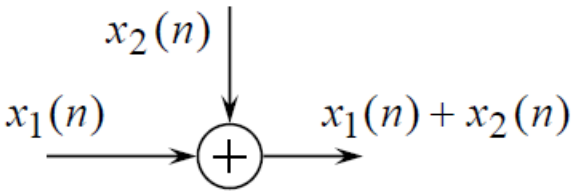


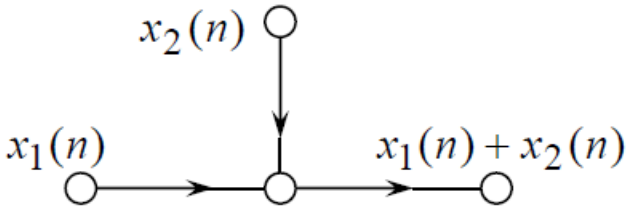
Signal Flow Graph

- Filters can be described in terms 3 basic operations

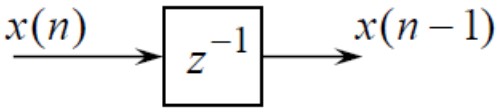
block diagram *signal flow*

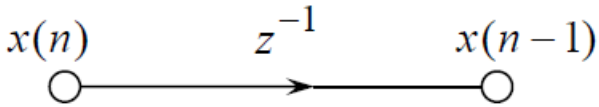
- addition



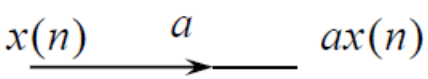


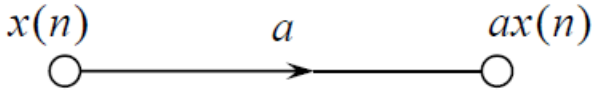
- delay





- multiplication by a constant





Type of Digital Filters

- **Finite Impulse Response (FIR) Filters**

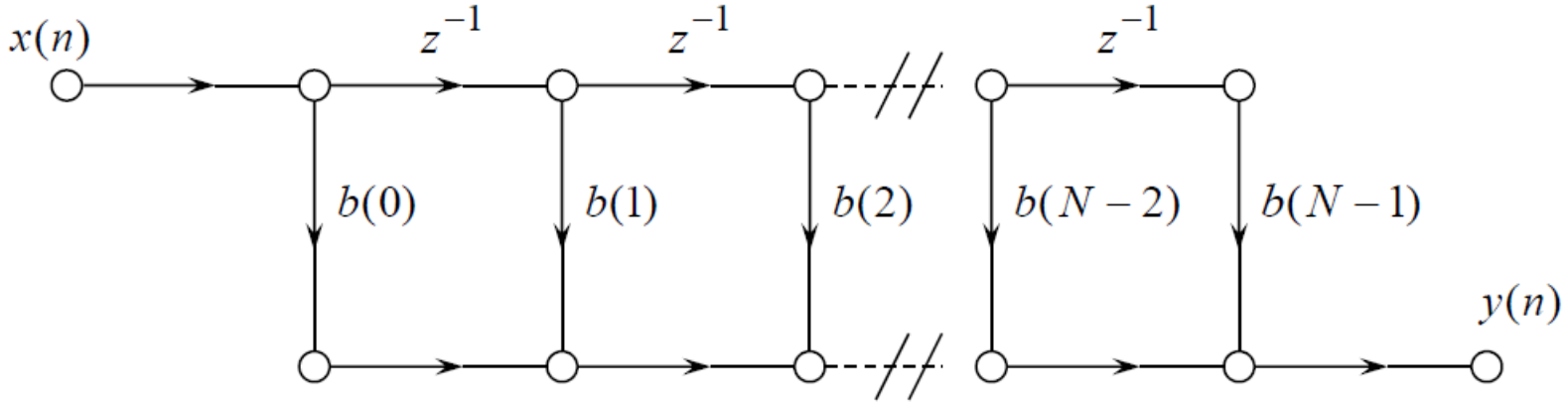
→ Difference Equation of Input-Output:

$$y(n) = \sum_{k=0}^{M} b_k x(n-k)$$

→ System Function:

$$H(z) = \sum_{k=0}^{M} b_k z^{-k}$$

→ Signal Flow



Type of Digital Filters

- **Infinite Impulse Response (IIR) Filters**

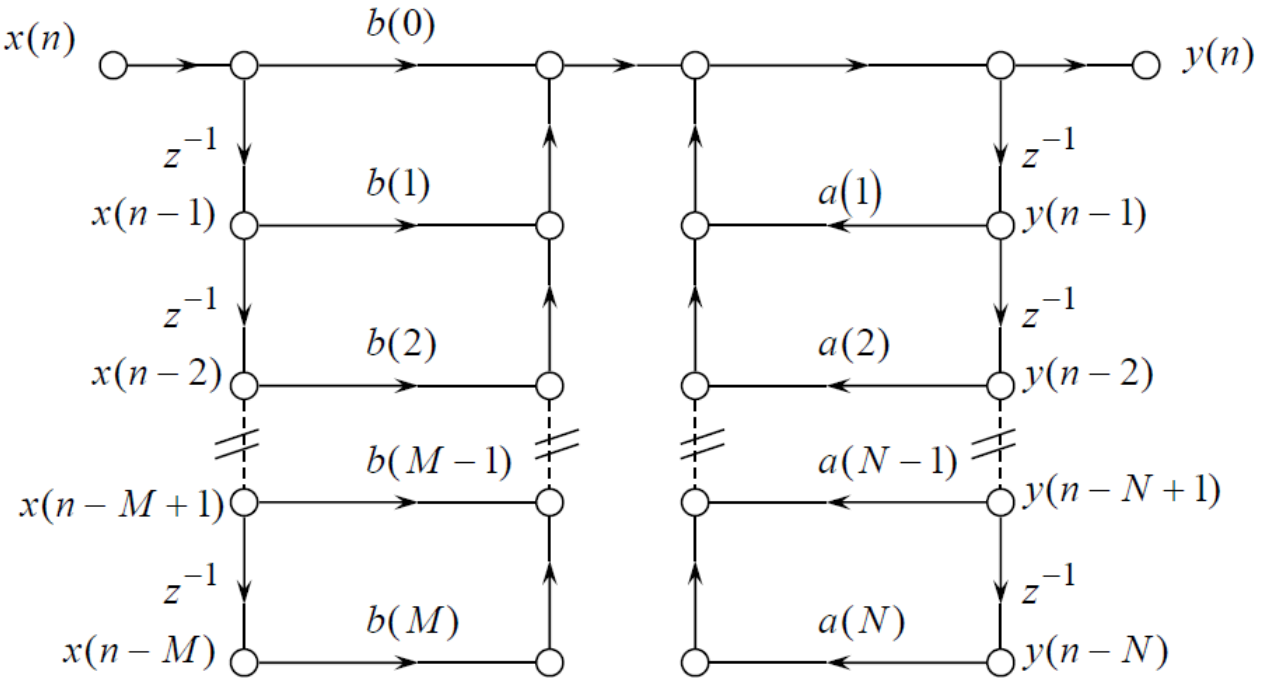
→ Difference Equation of Input-Output:

$$y(n) = \sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

→ System Function:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

→ Signal Flow (Form1):



Type of Digital Filters

- **Infinite Impulse Response (IIR) Filters**

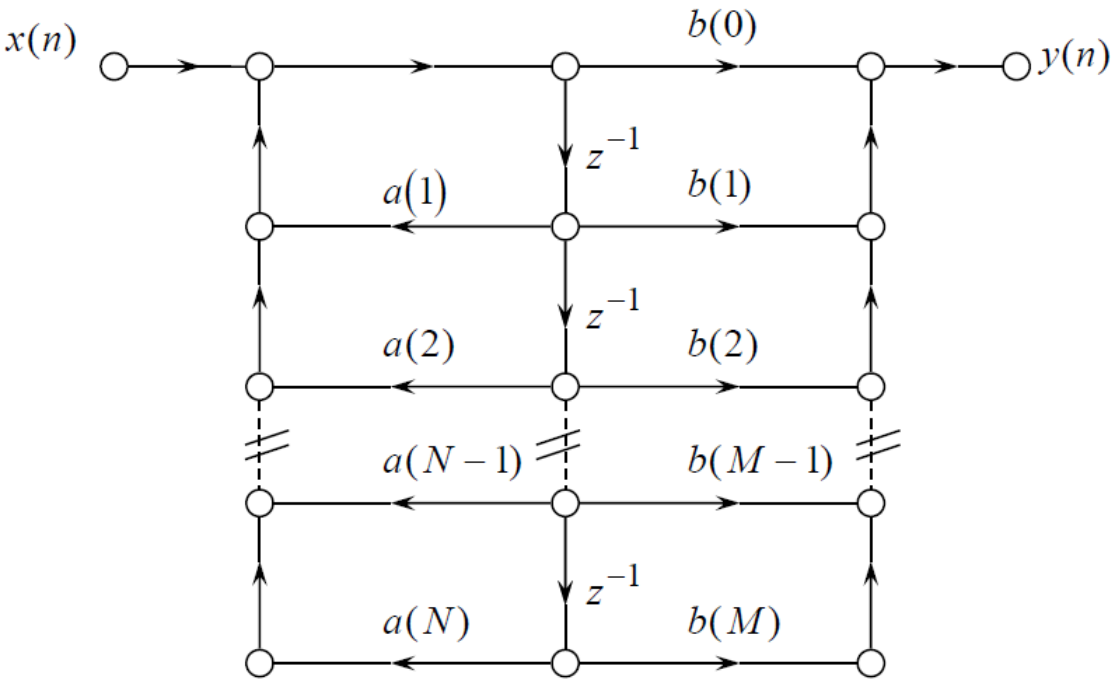
→ Difference Equation of Input-Output:

$$y(n) = \sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

→ System Function:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

→ Signal Flow (Form2):



Filter Classification

FILTER IMPLEMENTED BY:

FILTER USED FOR:	Convolution *Finite Impulse Response (FIR)*	Recursion *Infinite Impulse Response (IIR)*
Time Domain *(smoothing, DC removal)*	Moving average	Single pole
Frequency Domain *(separating frequencies)*	Windowed-sinc	Chebyshev
Custom *(Deconvolution)*	FIR custom	Iterative design

FIR (Time-Domain): Moving Average

Filter Classification

FILTER IMPLEMENTED BY:

FILTER USED FOR:	Convolution *Finite Impulse Response (FIR)*	Recursion *Infinite Impulse Response (IIR)*
Time Domain *(smoothing, DC removal)*	Moving average	Single pole
Frequency Domain *(separating frequencies)*	Windowed-sinc	Chebyshev
Custom *(Deconvolution)*	FIR custom	Iterative design

Moving Average Filter

- **Moving Average**

  → Averaging a number of points from the input signal

$$y(n) = \frac{1}{M} \sum_{i=0}^{M-1} x(n+i) \qquad y(n) = \frac{1}{M} \sum_{i=-(M-1)/2}^{(M-1)/2} x(n+i)$$

- **Example**

  → A 5-point moving average filter

$$y(80) = \frac{x(80) + x(81) + x(82) + x(83) + x(84)}{5}$$

$$y(80) = \frac{x(78) + x(79) + x(80) + x(81) + x(82)}{5}$$

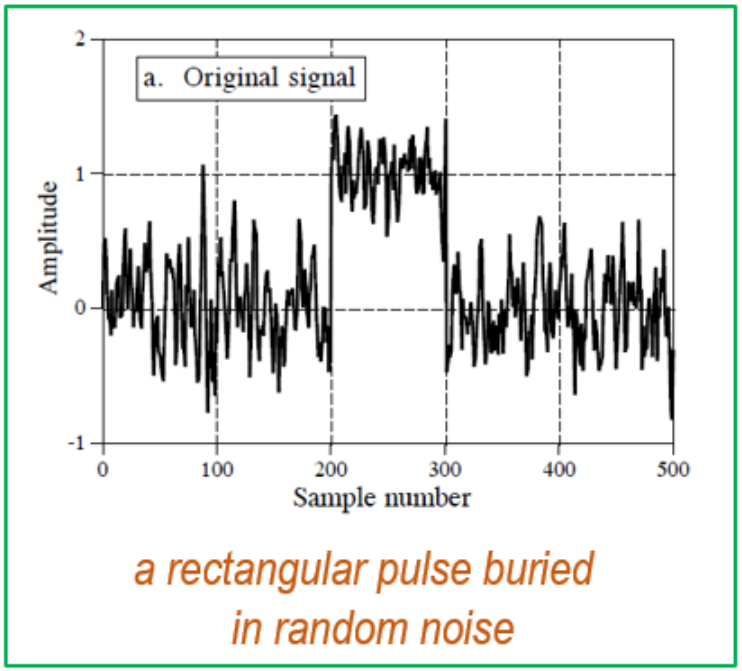
→ Filter Kernel

$$\ldots, 0, 0, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, 0, 0, \ldots$$

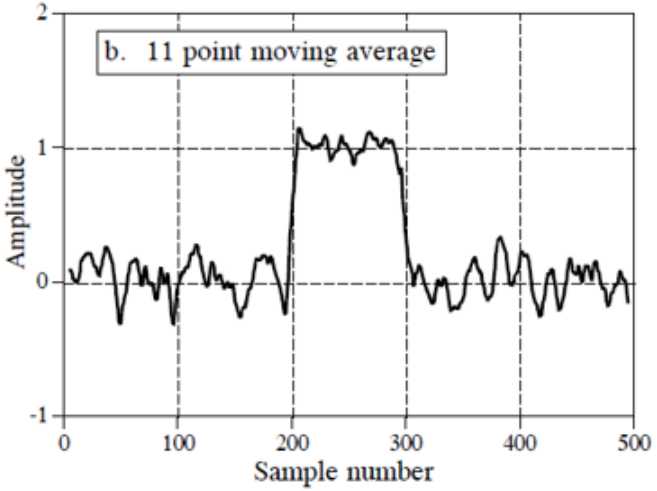
Convolution of the input signal with a rectangular pulse having an area of 1.

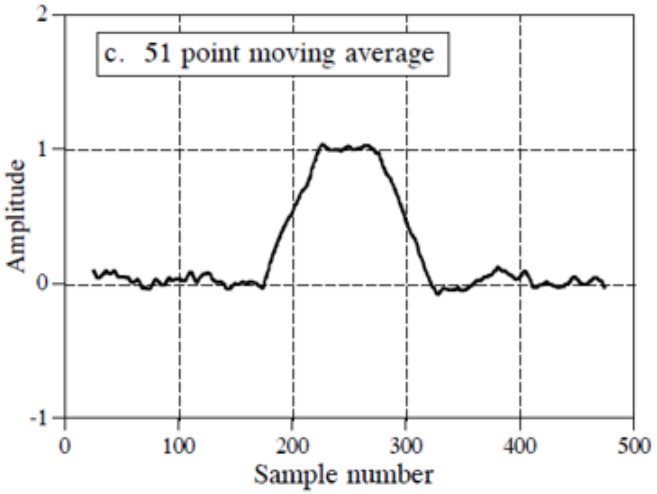
Example Moving Average Filter

- **Rectangular Pulse (Time-Domain)**



a rectangular pulse buried in random noise

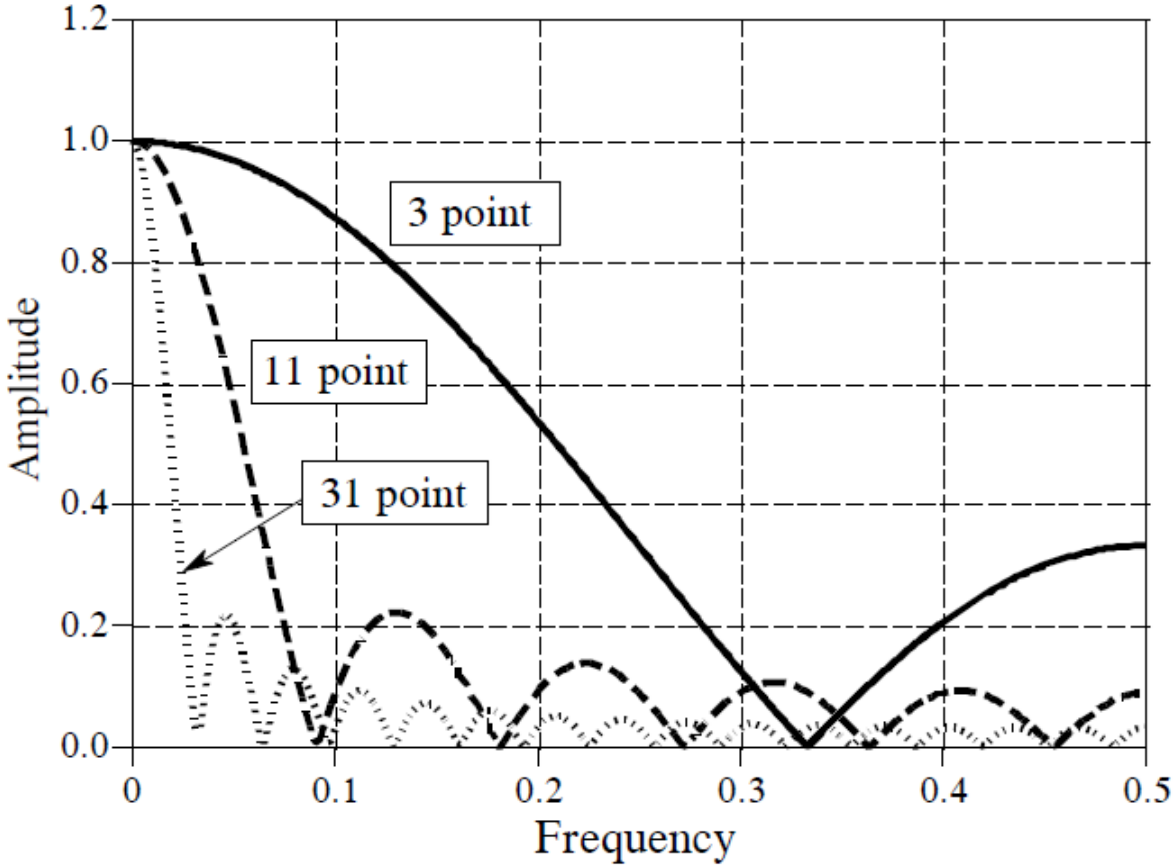




Good performance in the time domain
Good smoothing filter

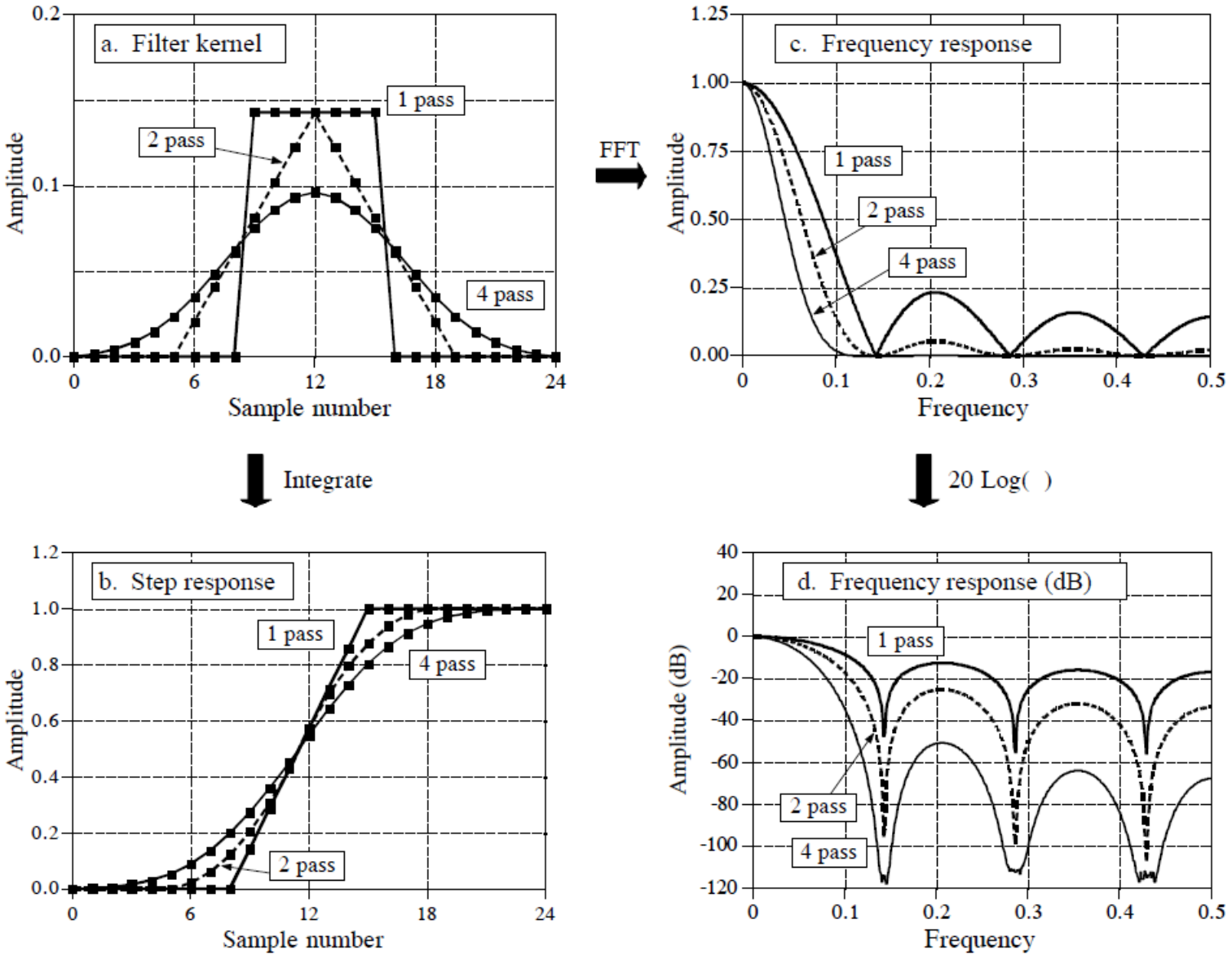
Example Moving Average Filter

- **Frequency Response (Fourier Transform of Rectangular Pulse)**



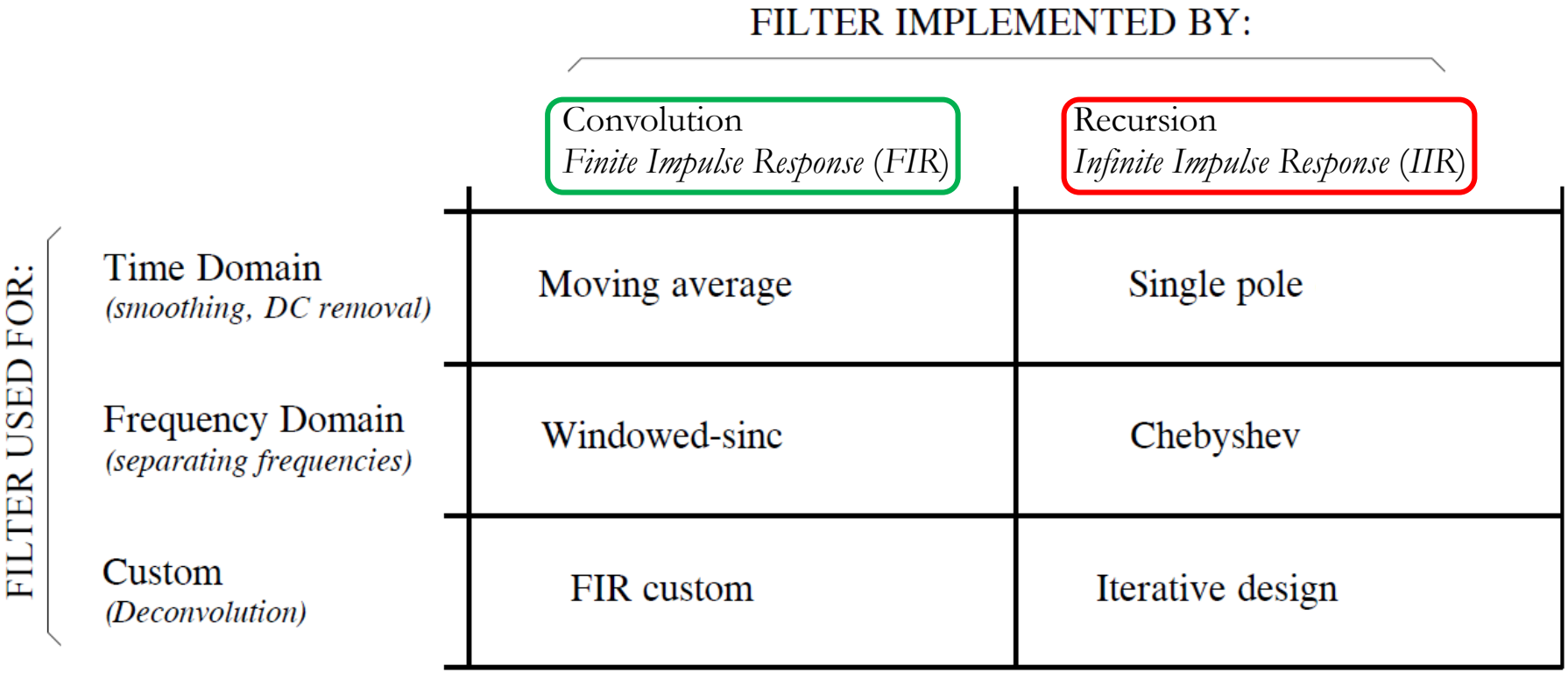
Poor performance in the frequency domain
Bad low-pass filter

Multi-pass Moving Average Filter



FIR (Frequency-Domain): Windowed-Sinc

Filter Classification



Windowed-Sinc Filter

- **Sinc**

  → Windowed-sinc filters are used to separate one band of frequencies from another

  → Sinc Function:

$$h[i] = \frac{\sin(2\pi f_c i)}{i\pi}$$

A perfect low-pass filter!
But sinc function continues infinity without dropping to zero amplitude.

- **Windowed-Sinc**

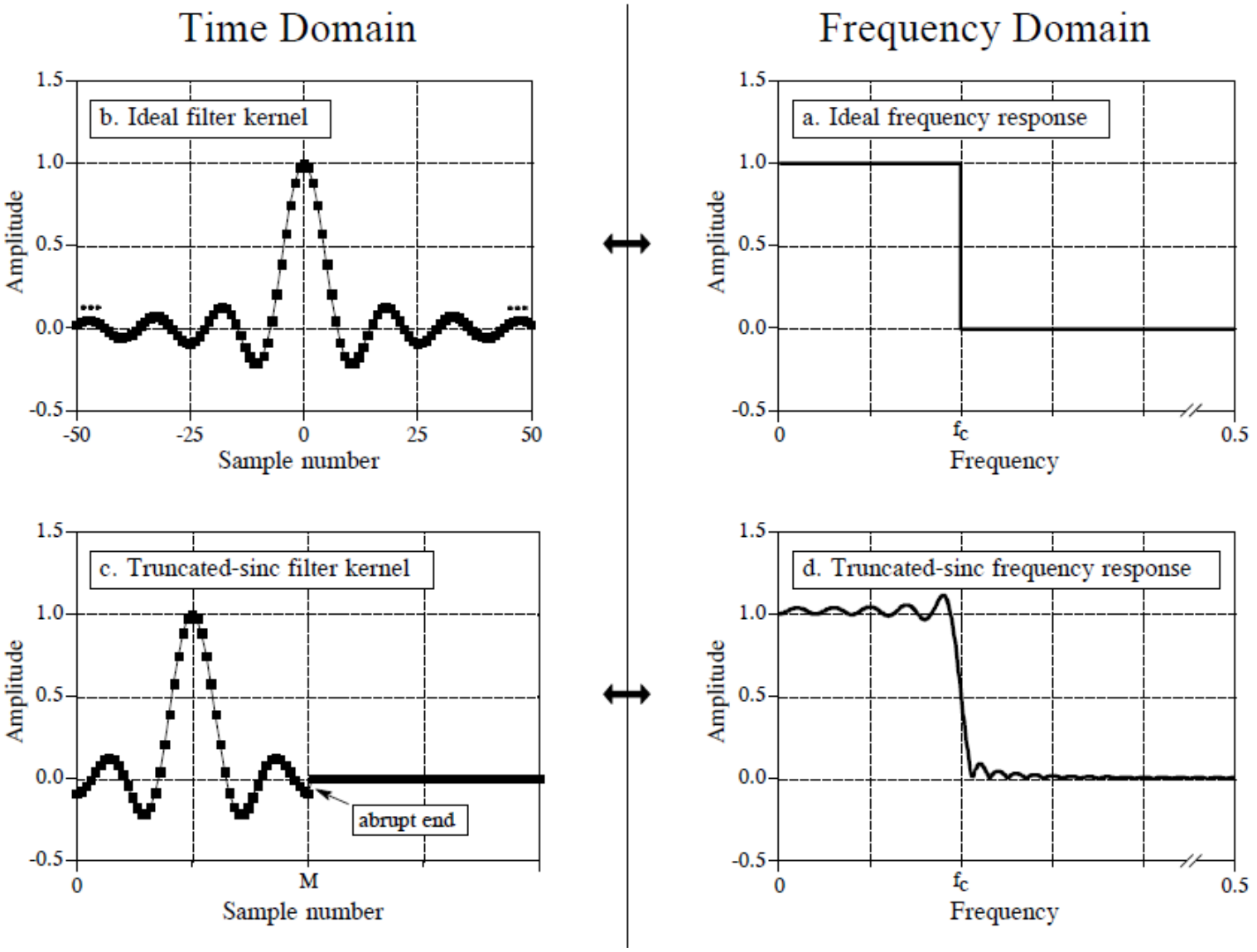
  → Multiply with Hamming window or Blackman window

$$w[i] = 0.54 - 0.46\cos(2\pi i/M)$$

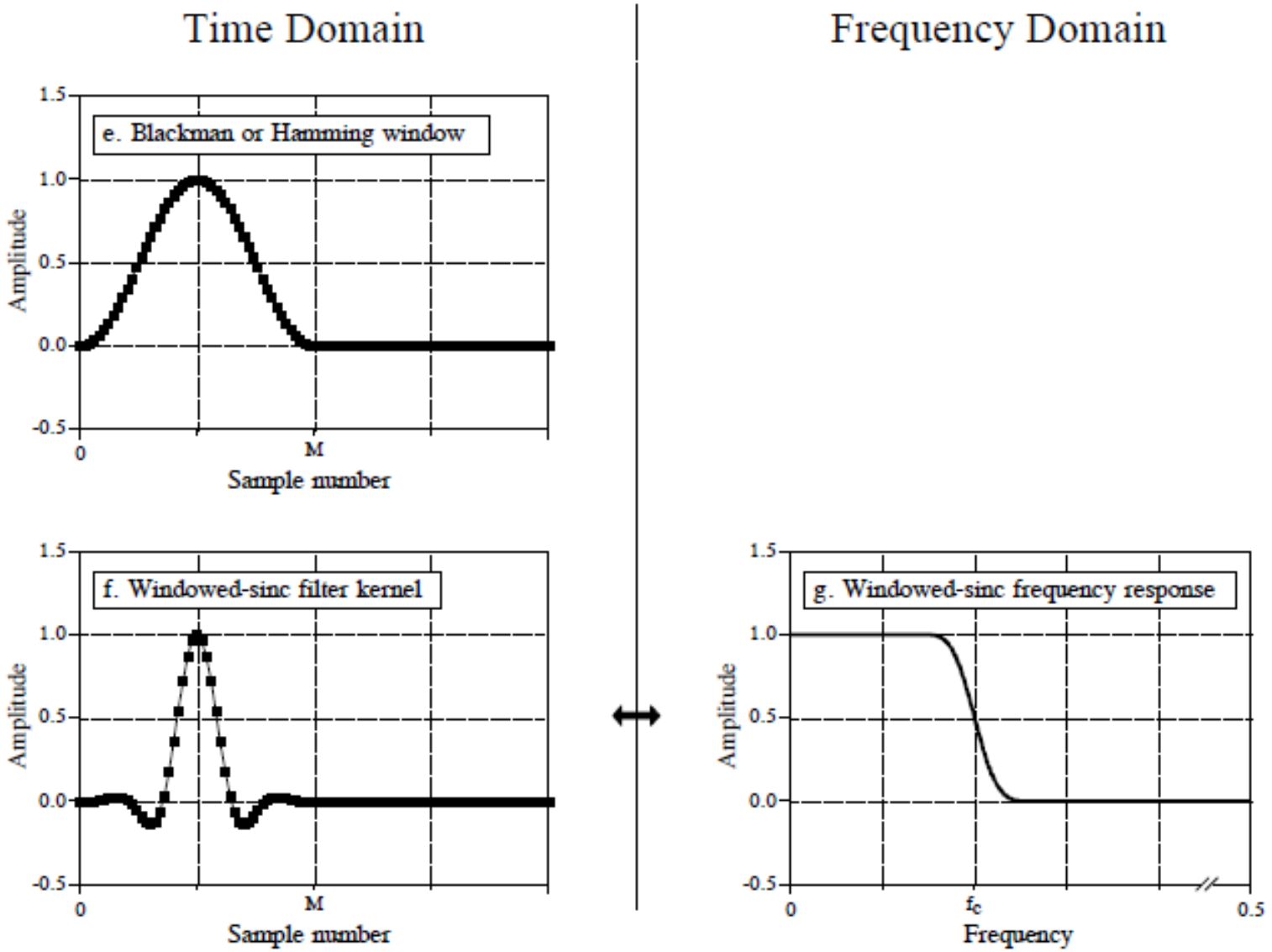
$$w[i] = 0.42 - 0.5\cos(2\pi i/M) + 0.08\cos(4\pi i/M)$$

Windowed-sinc filters are used to separate one band of frequencies from another

Sinc Filter

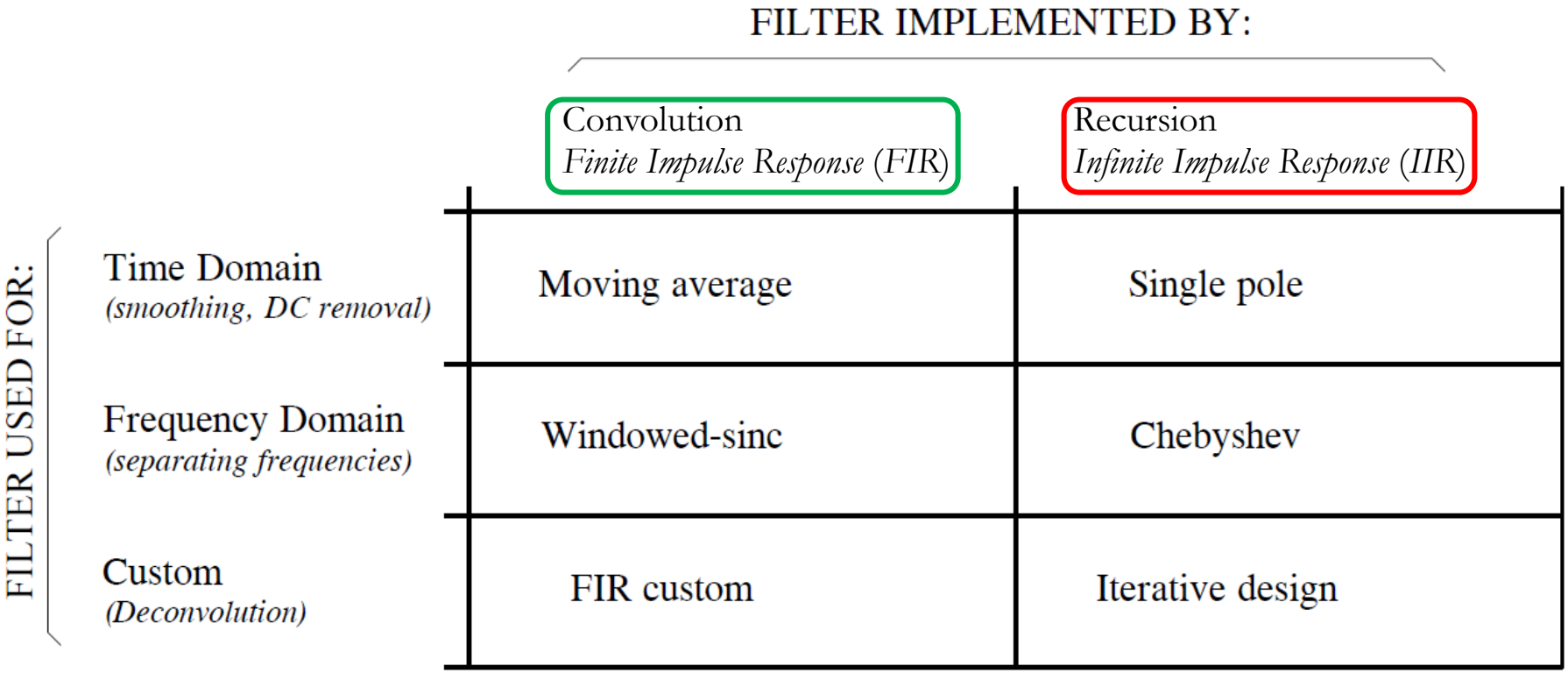

Time Domain
Frequency Domain
b. Ideal filter kernel
a. Ideal frequency response
c. Truncated-sinc filter kernel
d. Truncated-sinc frequency response
abrupt end
Amplitude
Sample number
Frequency
1.5
1.0
0.5
0.0
-0.5
-50
-25
0
25
50
M
f_c

Windowed-Sinc Filter



FIR (Custom)

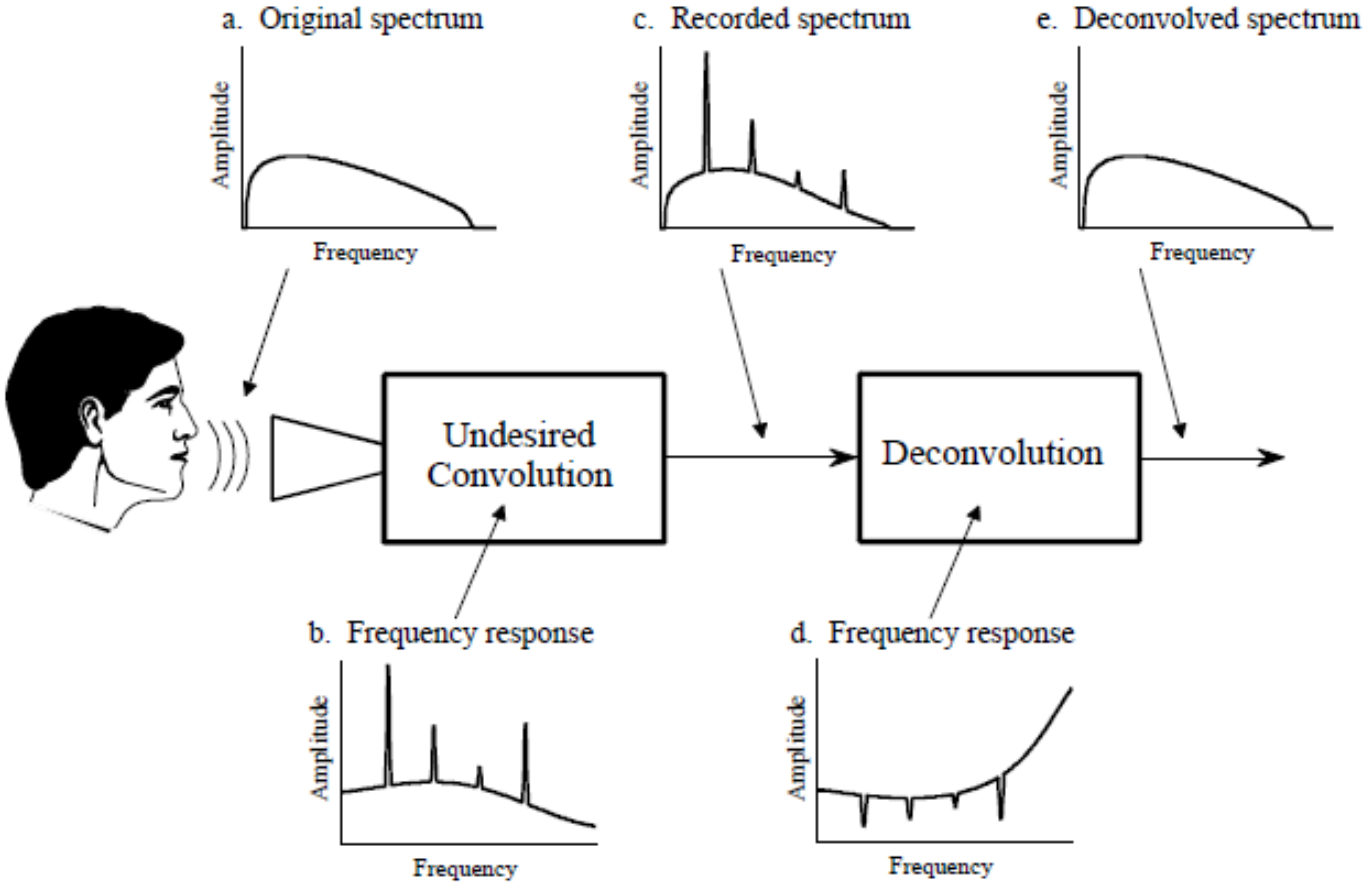
Filter Classification



Custom Filter: Deconvolution

- **Deconvolution**

  → A way of restoring signals that have undergone an unwanted convolution



IIR (Time-Domain): Single Pole

Filter Classification

FILTER USED FOR:	FILTER IMPLEMENTED BY: Convolution *Finite Impulse Response (FIR)*	Recursion *Infinite Impulse Response (IIR)*
Time Domain *(smoothing, DC removal)*	Moving average	Single pole
Frequency Domain *(separating frequencies)*	Windowed-sinc	Chebyshev
Custom *(Deconvolution)*	FIR custom	Iterative design

IIR: Recursive Filter

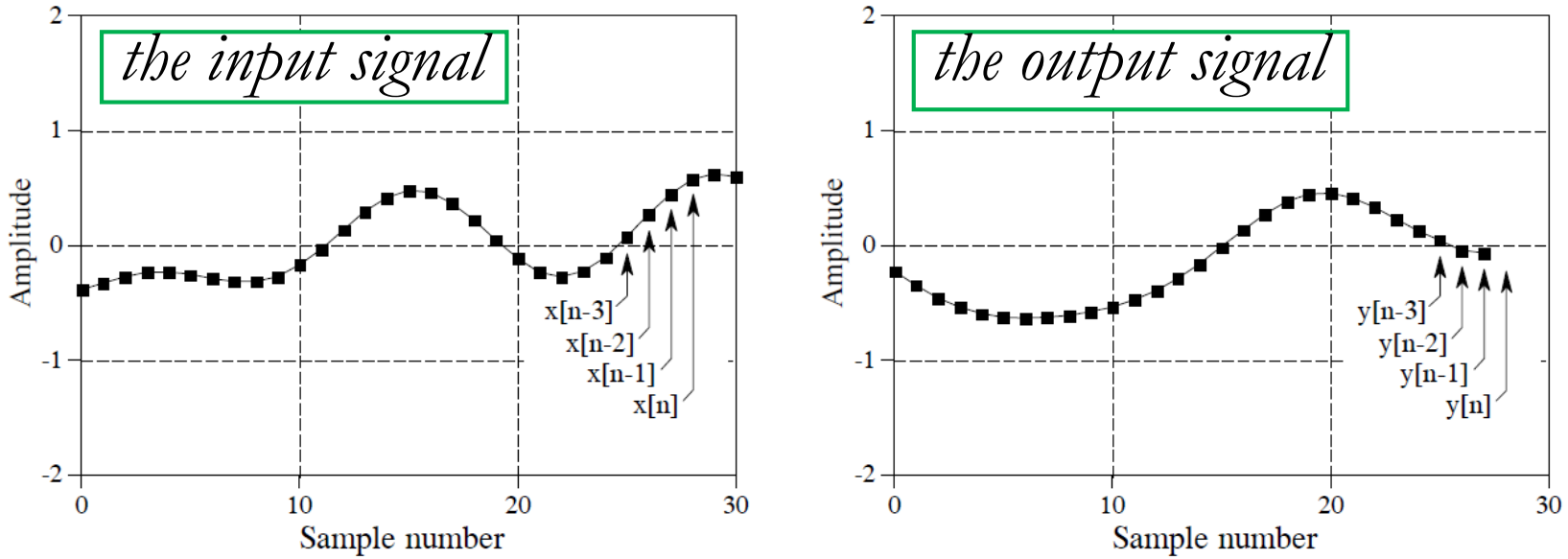
- **Recursive Filter**

→ Achieving a long impulse response, without having to perform a long convolution

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 (n-2) + b_3 (n-3) + \cdots$$

$$+ a_1 y(n-1) + a_2 y(n-2) + a_3 y(n-3) + \cdots$$

previously calculated values of the output signal



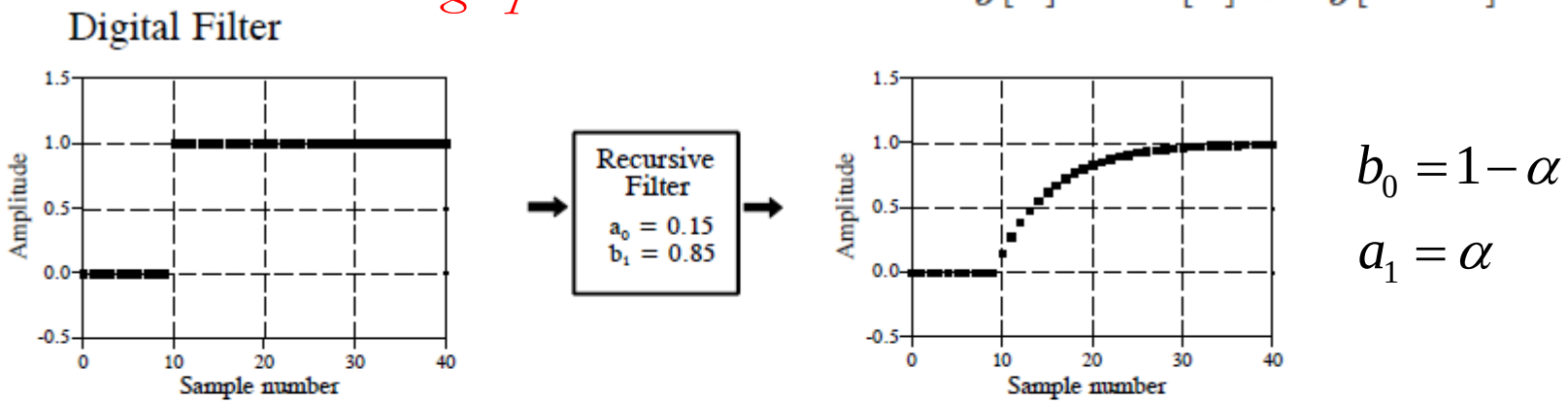
In actual practice, no more than about a dozen recursion coefficients can be used or the filter becomes unstable.

IIR: Recursive Filter

- **Single-Pole Recursive Filter**

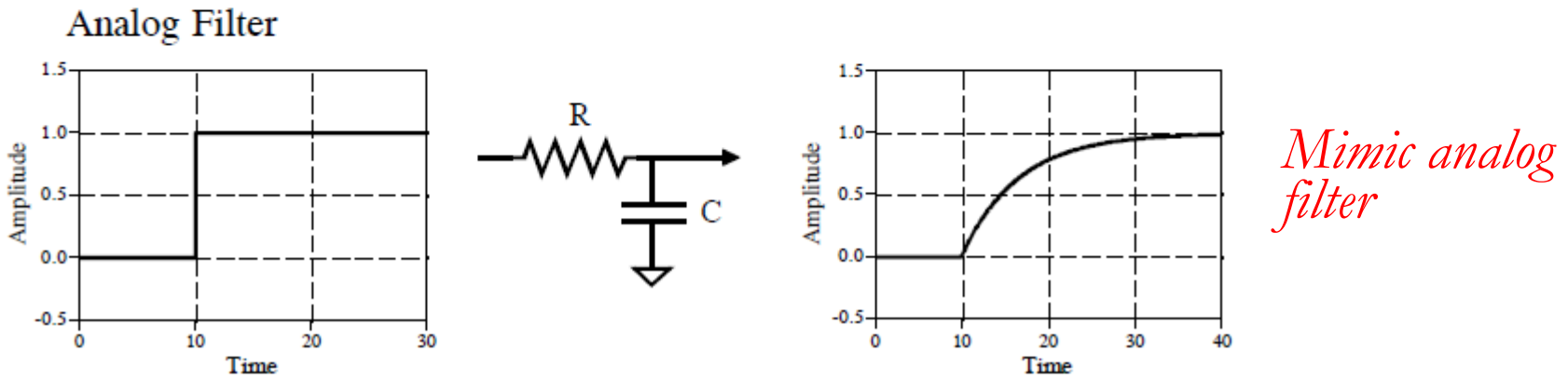
  → Controlled by a parameter α, a value between zero and one, the amount of decay between adjacent samples

Single-pole Low Pass Filter $y[n] = bx[n] + ay[n-1]$



$b_0 = 1 - \alpha$

$a_1 = \alpha$



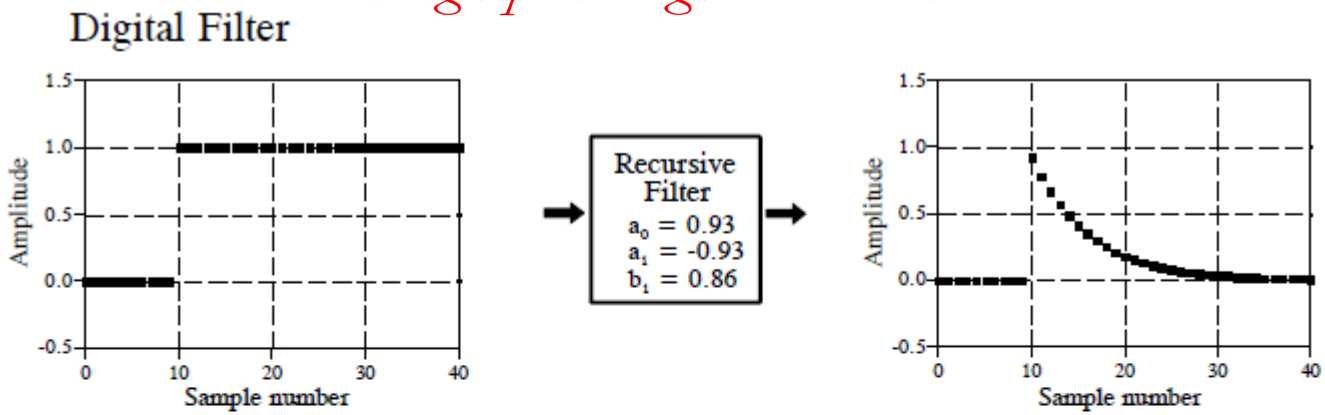
Mimic analog filter

IIR: Recursive Filter

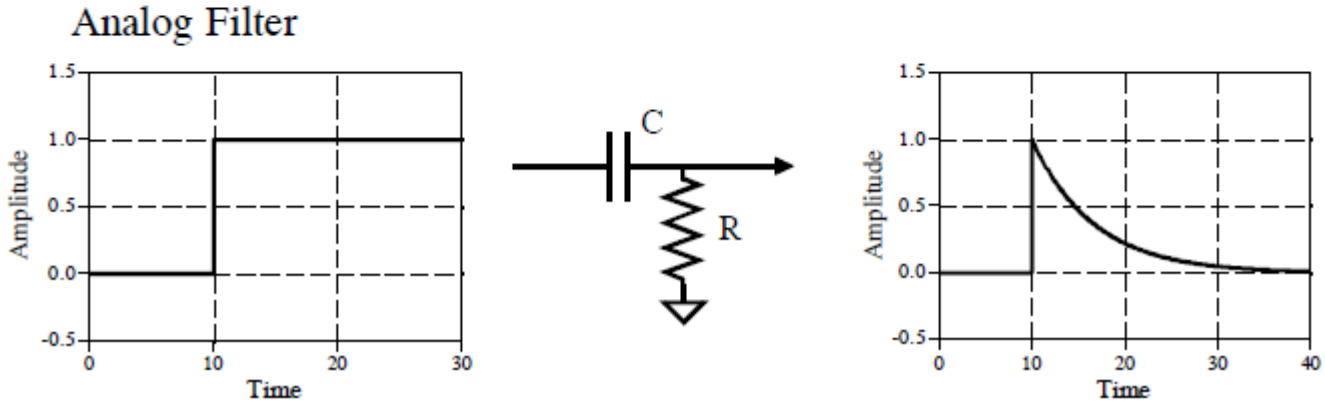
- **Single-Pole Recursive Filter**

  → Controlled by a parameter α, a value between zero and one, the amount of decay between adjacent samples

Single-pole High Pass Filter



$$b_0 = (1+\alpha)/2$$
$$b_1 = -(1+\alpha)/2$$
$$a_1 = \alpha$$

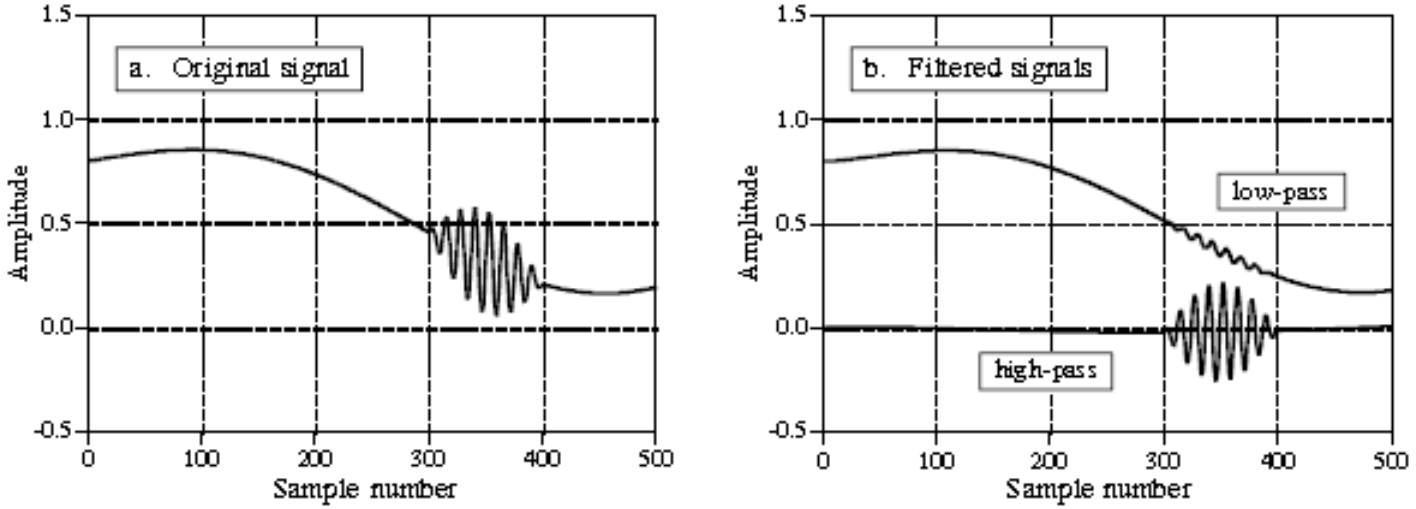


Mimic analog filter

IIR: Recursive Filter

- **Single-Pole Recursive Filter**

  → Controlled by a parameter α, a value between zero and one, the amount of decay between adjacent samples



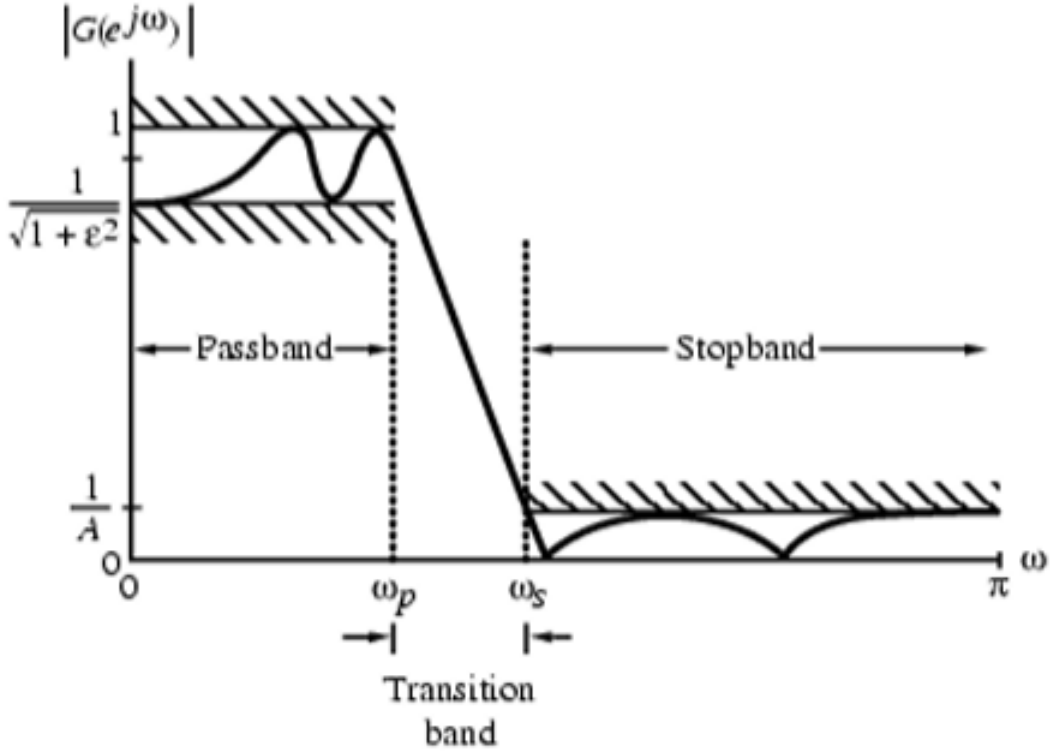
Single pole low-pass and high-pass filters are used to separate the two components.

IIR (Frequency-Domain): Chebyshev

IIR Filter Specification

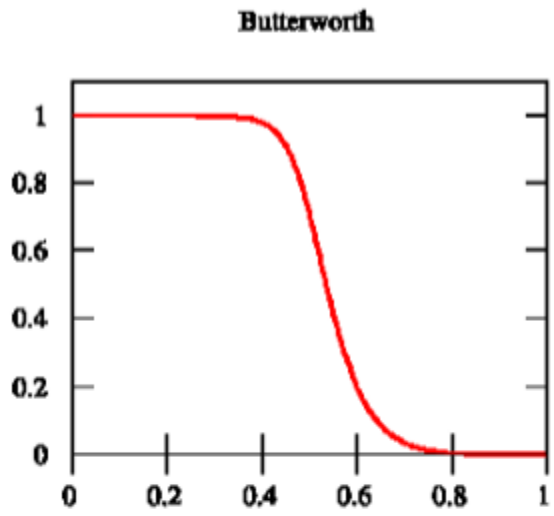
- **Magnitude Specification**

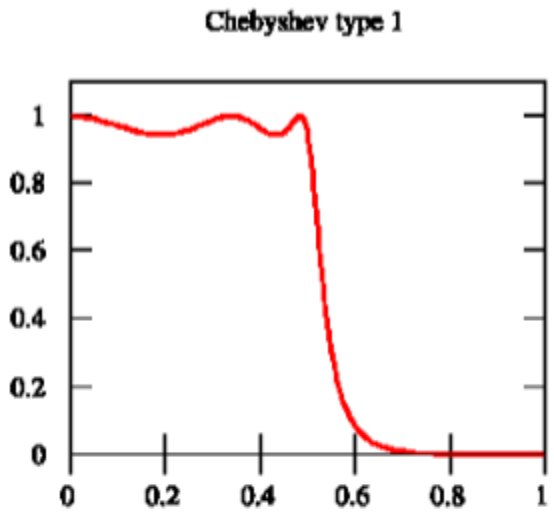
Magnitude specifications may alternately be given in a normalized form as indicated below

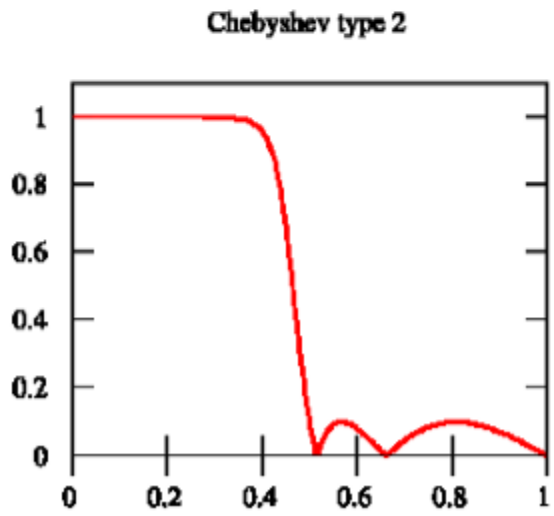


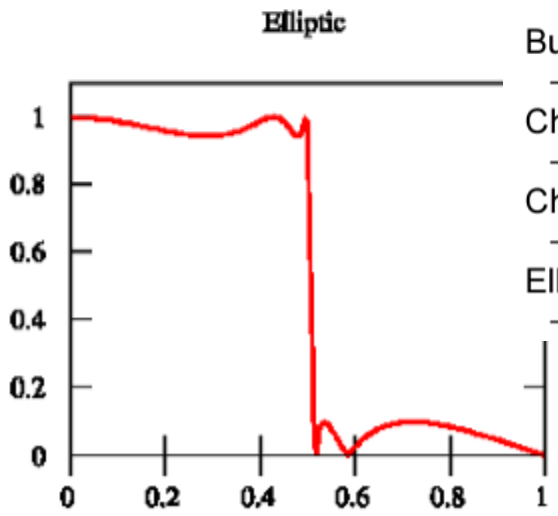
IIR Filter Specification

- **Mimic Analog Filter**









Butterworth filter
- no gain ripple in pass band and stop band, slow cutoff

Chebyshev filter (Type I)
- no gain ripple in stop band, moderate cutoff

Chebyshev filter (Type II)
- no gain ripple in pass band, moderate cutoff

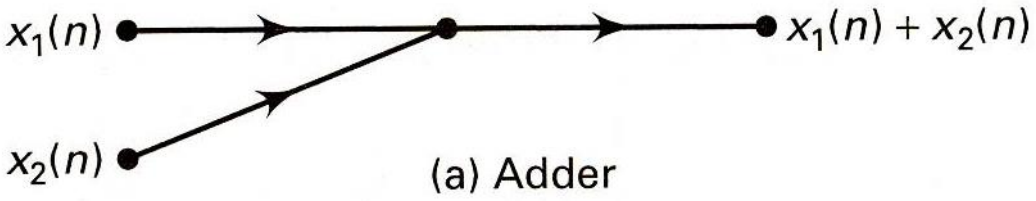
Elliptic filter
- gain ripple in pass and stop band, fast cutoff

IIR Filter Structures

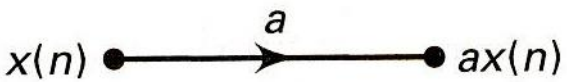
Why different structures?

computational complexity
memory requirements
finite-word-length effects

Three Basic Elements

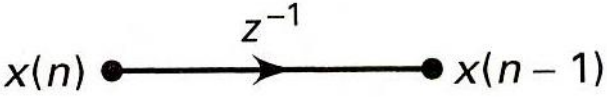


(a) Adder



(b) Multiplier

gain



(c) Delay element

shifter or memory

Our filters are **LTI** *systems.*

IIR Filter Structure

- **Three Different Structures**

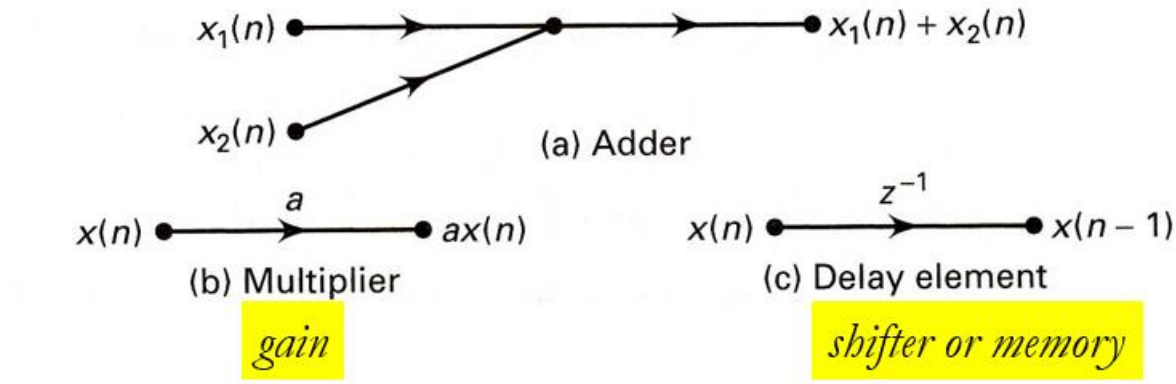
  → Different ways of representing the system w.r.t. memory storage

  Direct form (I and II)
  Cascade form
  Parallel form

- **Why Different Structures?**

computational complexity
memory requirements
finite-word-length effects

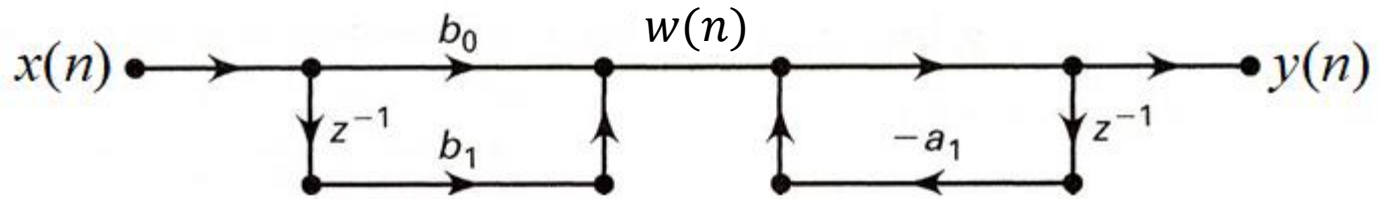
Three Basic Elements



IIR Filter Structure

- **The First-order System**

$$y(n) = -a_1 y(n-1) + b_0 x(n) + b_1 x(n-1)$$



- Separate delays (memory) for both the input and output signal samples
- Can be viewed as two linear time-invariant systems in cascade

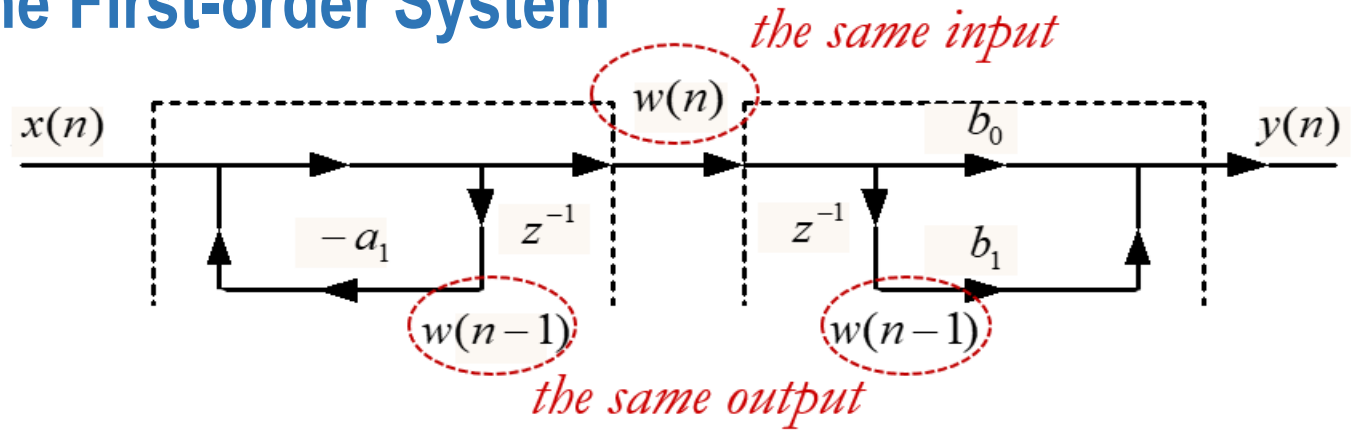
$w(n) = b_0 x(n) + b_1 x(n-1)$ a nonrecursive system

$y(n) = -a_1 y(n-1) + v(n)$ a recursive system

If we interchange the order of the cascaded linear time-invariant systems, the overall system response remains the same.

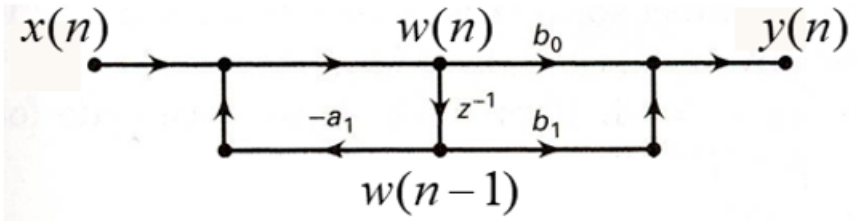
IIR Filter Structure

- **The First-order System**



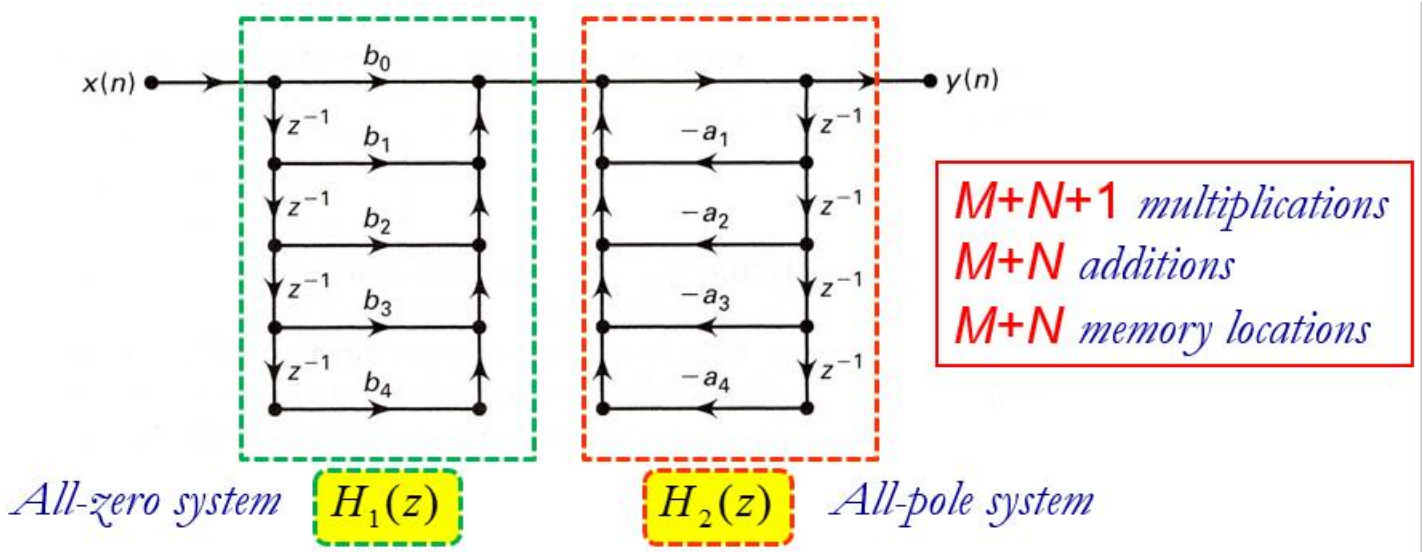
We interchange the order of the recursive and nonrecursive systems.

$$w(n) = -a_1 w(n-1) + x(n)$$
$$y(n) = b_0 w(n) + b_1 w(n-1)$$



Requires only one delay, hence more efficient in terms of memory requirements.

IIR: Direct Form Structure I



$$M = N = 4$$

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + b_3 x(n-3) + b_4 x(n-4)$$
$$- a_1 y(n-1) - a_2 y(n-2) - a_3 y(n-3) - a_4 y(n-4)$$

$$H(z) = H_1(z) H_2(z)$$

$$H_1(z) = \sum_{n=0}^{M} b_n z^{-n}, H_2(z) = \frac{1}{1 + \sum_{n=1}^{N} a_n z^{-n}}$$

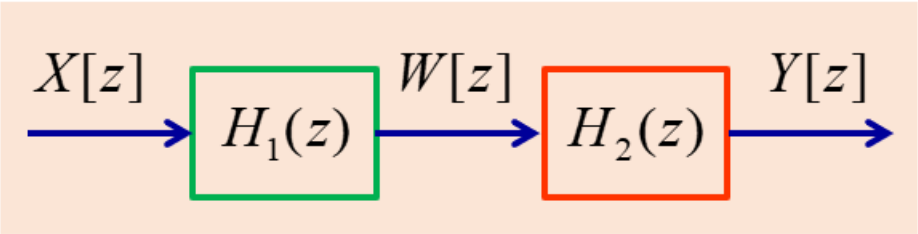
IIR: Direct Form Structure I

$$H(z)=\frac{Y(z)}{X(z)}=\frac{b_0+b_1z^{-1}+\cdots+b_Mz^{-M}}{1+a_1z^{-1}+\cdots+a_Nz^{-N}}=\frac{B(z)}{A(z)}$$

$$Y(z)=X(z)H(z)$$

$$=X(z)\frac{B(z)}{A(z)}$$

$$=X(z)B(z)\frac{1}{A(z)}$$



$$W(z)=X(z)B(z)=X(z)\left(b_0+b_1z^{-1}+\cdots+b_Mz^{-M}\right)$$

$$w(n)=b_0x(n)+b_1x(n-1)+\cdots+b_Mx(n-M)$$

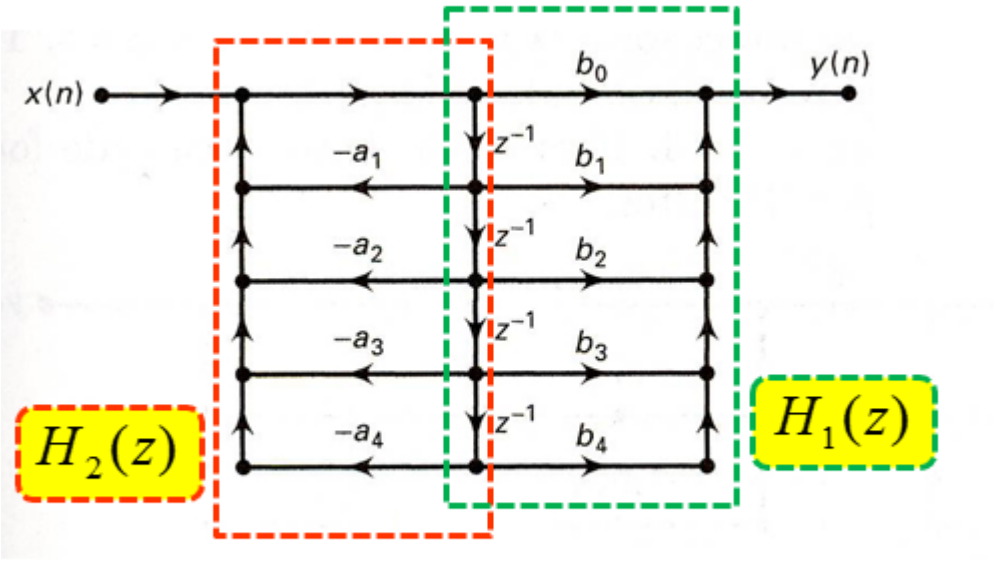
$$Y(z)=\frac{W(z)}{A(z)}=\frac{W(z)}{1+a_1z^{-1}+\cdots+a_Nz^{-N}}$$

$$y(n)+a_1y(n-1)+\cdots+a_Ny(n-N)=w(n)$$

$$y(n)=w(n)-a_1y(n-1)-\cdots-a_Ny(n-N)$$

IIR: Direct Form Structure II

the commutative law of the convolution



one delay line removed

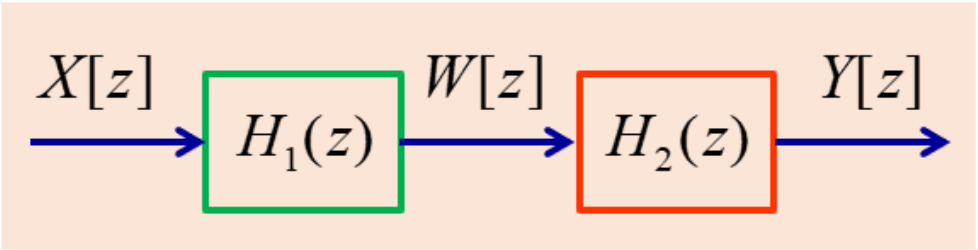
M+N+1 multiplications
M+N additions
Maximum of {M, N} memory locations

IIR: Direct Form Structure II

$$H(z)=\frac{Y(z)}{X(z)}=\frac{b_0+b_1z^{-1}+\cdots+b_Mz^{-M}}{1+a_1z^{-1}+\cdots+a_Nz^{-N}}=\frac{B(z)}{A(z)}$$

$$Y(z)=X(z)H(z)$$

$$=X(z)\frac{1}{A(z)}B(z)$$



$$X(z)=W(z)A(z)=W(z)\left(1+a_1z^{-1}+\cdots+a_Nz^{-N}\right)$$

$$x(n)=w(n)+a_1w(n-1)+\cdots+a_Nw(n-N)$$

$$x(n)-a_1w(n-1)-\cdots-a_Nw(n-N)=w(n)$$

$$Y(z)=W(z)B(z)=W(z)\left(b_0+b_1z^{-1}+\cdots+b_Mz^{-M}\right)$$

$$y(n)=b_0w(n)+b_1w(n-1)+\cdots+b_Mw(n-M)$$

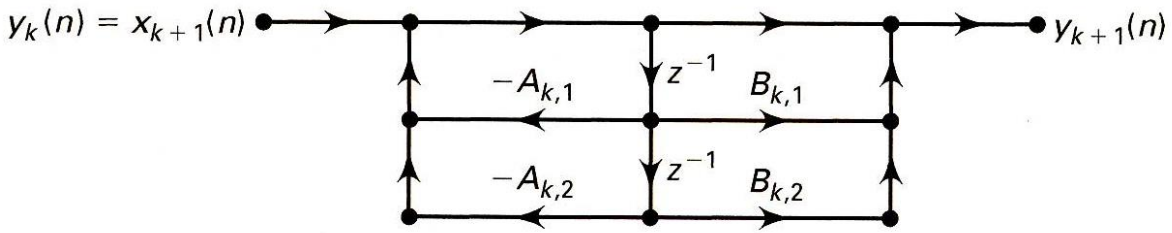
Cascade Form Structure

Cascade Form

→ Quantization can produce unstable filters

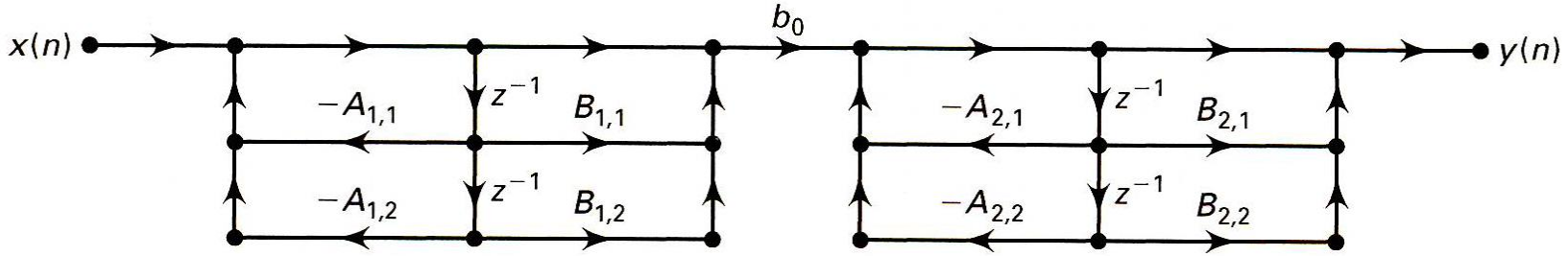
→ Higher-order filters: typically implemented as serially-cascaded biquad sections

Biquad section structure



The input to the k-th biquad section is the output from the (k-1)-th biquad section, while the output from the k-th biquad is the input to the (k+1)-th biquad.

Cascade form structure for N=4



Cascade Form Structure

```
function [b0,B,A] = dir2cas(b,a)
% DIRECT-form to CASCADE-form conversion (cplxpair version)
% ------------------------------------------------------------------------
% [b0,B,A] = dir2cas(b,a)
% b0 = gain coefficient
%  B = K by 3 matrix of real coefficients containing bk's
%  A = K by 3 matrix of real coefficients containing ak's
%  b = numerator polynomial coefficients of DIRECT form
%  a = denominator polynomial coefficients of DIRECT form

% compute gain coefficient b0
b0 = b(1); b = b/b0; a0 = a(1); a = a/a0; b0 = b0/a0;
%
M = length(b); N = length(a);
if N > M
b = [b zeros(1,N-M)];
elseif M > N
a = [a zeros(1,M-N)]; N = M;
else
NM = 0;
end
%
K = floor(N/2); B = zeros(K,3); A = zeros(K,3);
if K*2 == N;
b = [b 0]; a = [a 0];
end
%
broots = cplxpair(roots(b)); aroots = cplxpair(roots(a));
for i=1:2:2*K
Brow = broots(i:1:i+1,:); Brow = real(poly(Brow));
B(fix(i+1)/2),:) = Brow;
Arow = aroots(i:1:i+1,:); Arow = real(poly(Arow));
A(fix((i+1)/2),:) = Arow;
end
```

SP Toolbox tf2sos
(*transfer function to second-order section*)

a similar operation.

Cascade Form Structure

```
function y = casfiltr(b0,B,A,x);
% CASCADE form realization of IIR and FIR filters
% ------------------------------------------------------------
% y = casfiltr(b0,B,A,x);
%  y = output sequence
% b0 = gain coefficient of CASCADE form
%  B = K by 3 matrix of real coefficients containing bk's
%  A = K by 3 matrix of real coefficients containing ak's
%  x = input sequence
%
[K,L] = size(B);
N = length(k); w = zeros(K+1,N); w(1,:) = x;
for i=1:1:K
      w(i+1,:) = filter(B(i,:),A(i,:),w(i,:));
end
y = b0*w(K+1,:)
```

Multiplication of several second-order polynomials

```
function [b,a] = cas2dir(b0,B,A);
% CASCADE-to-DIRECT form conversion
% ------------------------------------------------
% [b,a] = cas2dir(b0,B,A)
%  b = numerator polynomial coefficients of DIRECT form
%  a = denominator polynomial coefficients of DIRECT form
% b0 = gain coefficient
%  B = K by 3 matrix of real coefficients containing bk's
%  A = K by 3 matrix of real coefficients containing ak's
%
[K,L] = size(B);
b = [1]; a = [1];
for i=1:1:K
b=conv(b,B(i,:)); a=conv(a,A(i,:));
end
b = b*b0
```

SP Toolbox sos2tf

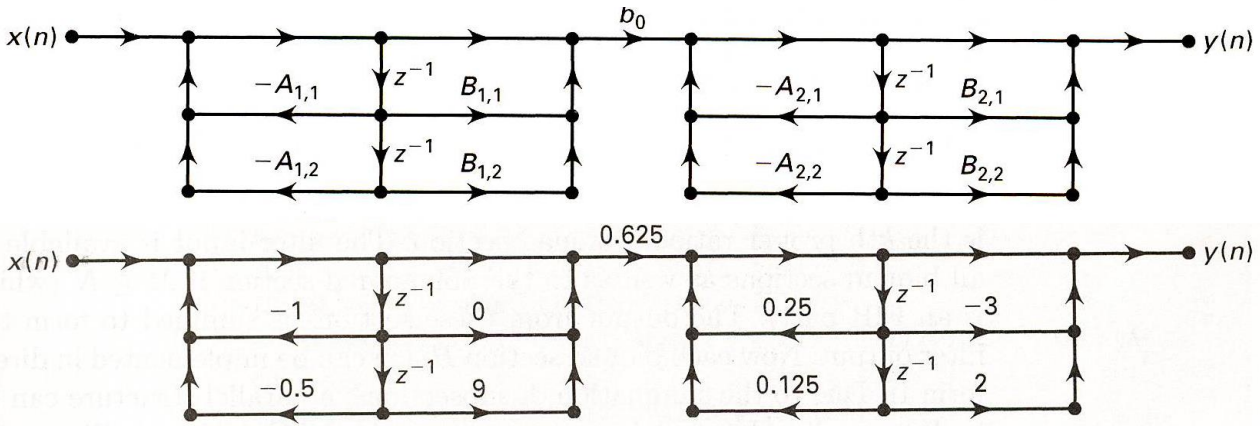
Example:

A filter is described by the following difference equation:

$$16y(n)+12y(n-1)+2y(n-2)-4y(n-3)-y(n-4)$$
$$=x(n)-3x(n-1)+11x(n-2)-27x(n-3)+18x(n-4)$$

Determine its cascade form structure.

```
>> b=[1 -3 11 -27 18]; a=[16 12 2 -4 -1];
>> [b0,B,A]=dir2cas(b,a)
b0 = 0.0625
 B =
      1.0000      -0.0000       9.0000
      1.0000      -3.0000       2.0000
A =
      1.0000       1.0000       0.5000
      1.0000      -0.2500      -0.1250
```

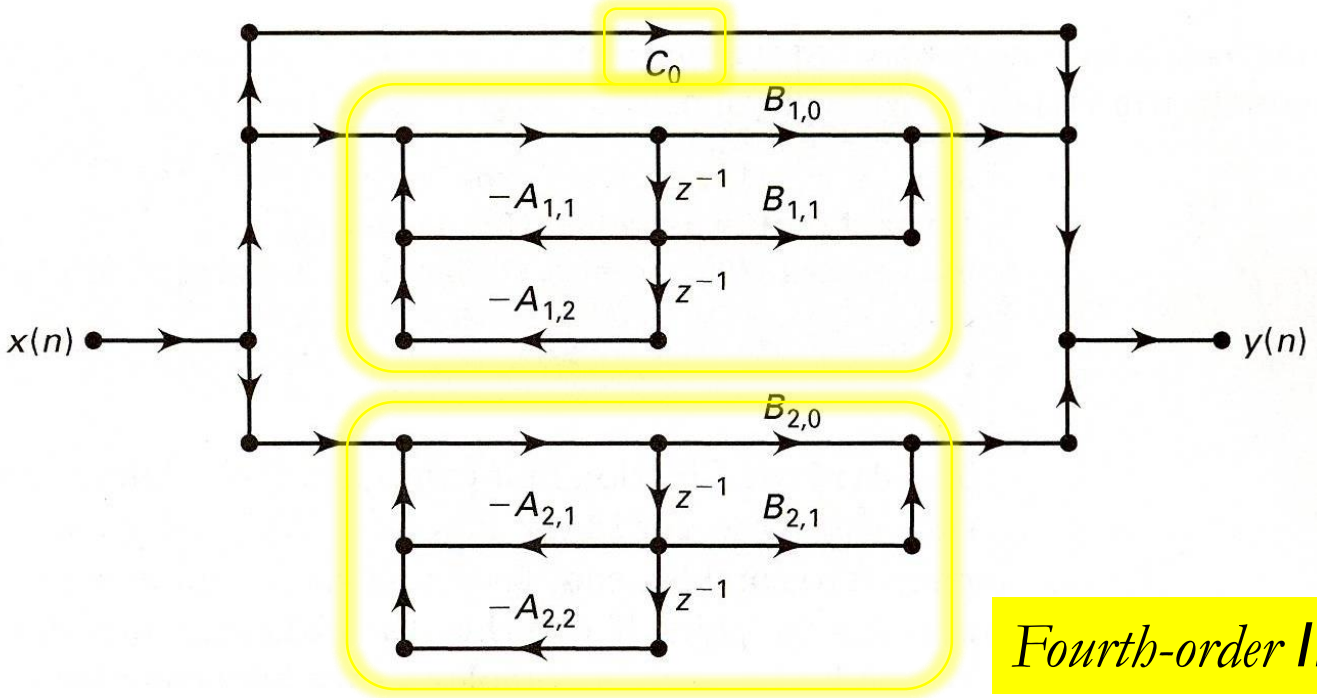


Parallel Form Structure

- **Parallel Form**

  → The filter input is available to all biquad sections as well as to the polynomial section.

Parallel Form Structure for M=N=4



Fourth-order IIR filter

Parallel Form Structure

```
function [C,B,A] = dir2par(b,a)
% DIRECT-form to PARALLEL-form conversion
% ---------------------------------------------------------
% [C,B,A] = dir2par(b,a)
%  C = Polynomial part when length(b) >= length(a)
%  B = K by 2 matrix of real coefficients containing bk's
%  A = K by 3 matrix of real coefficients containing ak's
%  b = numerator polynomial coefficients of DIRECT form
%  a = denominator polynomial coefficients of DIRECT form
%
M = length(b); N = length(a);

[r1,p1,C] = residuez(b,a);
p = cplxpair(p1,10000000*eps); I = cplxcomp(p1,p); r = r1(I);

K = floor(N/2); B = zeros(K,2); A = zeros(K,3);
if K*2 == N; %N even, order of A(z) odd, one factor is first order
 for i=1:2:N-2
 Brow = r(i:1:i+1,:); Arow = p(i:1:i+1,:);
 [Brow,Arow] = residuez(Brow,Arow,[]);
 B(fix((i+1)/2),:) = real(Brow); A(fix((i+1)/2),:) = real(Arow);
 end
 [Brow,Arow] = residuez(r(N-1),p(N-1,[]);
 B(K,:) = [real(Brow) 0]; A(K,:) = [real(Arow) 0];
 else
 for i=1:2:N-1
 Brow = r(i:1:i+1,:); Arow = p(i:1:i+1,:);
 [Brow,Arow] = residuez(Brow,Arow,[]);
 B(fix((i+1)/2),:) = real(Brow); A(fix((i+1)/2),:) = real(Arow);
 end
 end
```

Parallel Form Structure

```
function I = cplxcomp(p1,p2)
%  I = cplxcomp(p1,p2)
% Compares two complex pairs which contain the same scalar elements
%    but (possibly) at different indices.  This routine should be
%    used after CPLXPAIR routine for rearranging pole vectors and its
%    corresponding residue vector.
%         p2 =cplxpair(p1)
%
I = [];
for j=1:1:length(p2)
     for i=1:1:length(p1)
          if (abs(p1(i)-p2(j)) < 0.0001)
            I=[I,i];
          end
     end
end
I=I';
```

```
function y = parfiltr(C,B,A,x);
% PARALLEL form realization of IIR filters
% ------------------------------------------------
%    [y] = parfiltr(C,B,A,x);
%    y = output sequence
%    C = polynomial (FIR) part when M >= N
%    B = K by 2 matrix of real coefficients containing bk's
%    A = K by 3 matrix of real coefficients containing ak's
%    x = input sequence
%
[K,L] = size(B); N = length(x); w = zeros(K+1,N);
w(1,:) = filter(C,1,x);
for i = 1:1:K
          w(i+1,:) = filter(B(i,:),A(i,:),x);
end
y = sum(w);
```

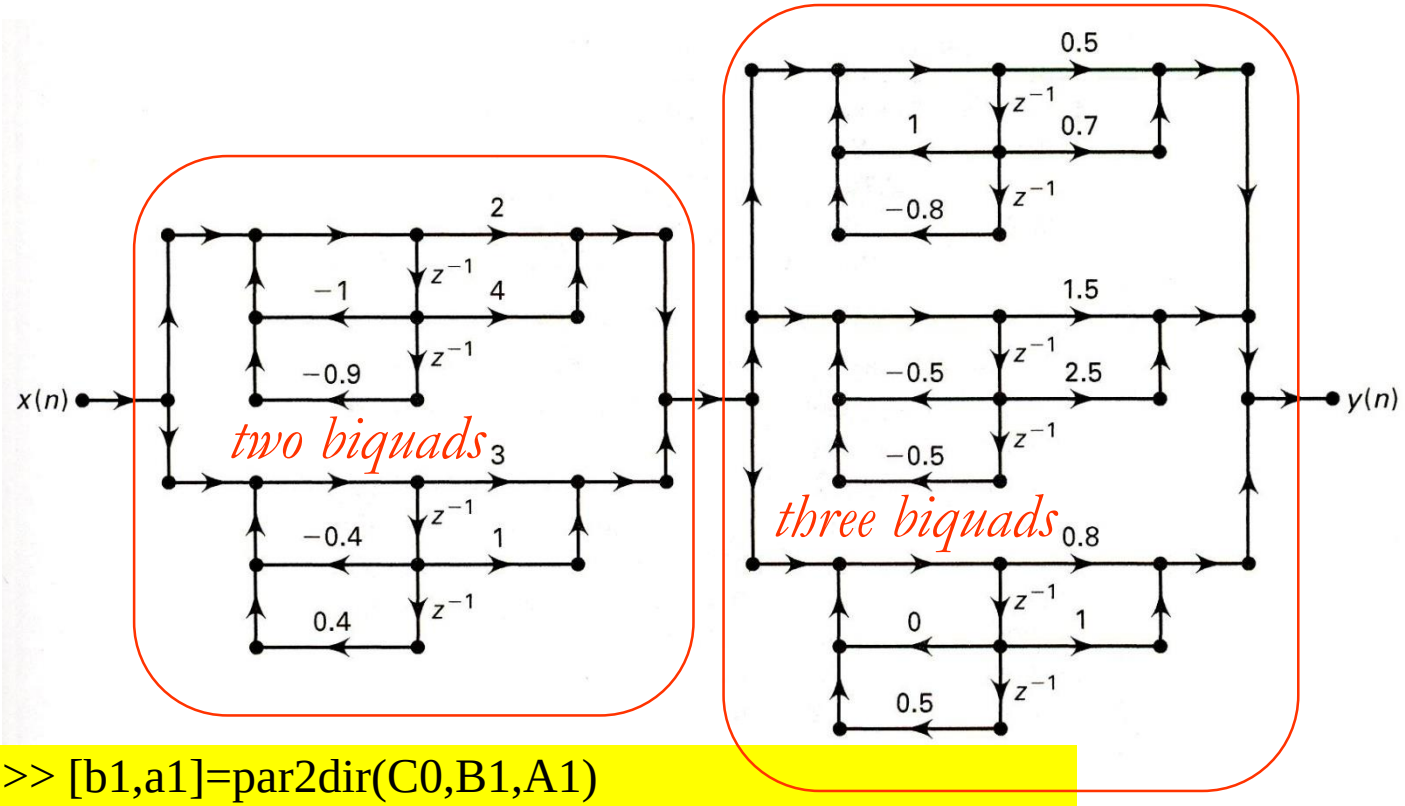
Parallel Form Structure

```
function [b,a] = par2dir(C,B,A);
% PARALLEL-to-DIRECT form conversion
% ------------------------------------------------
%    [b,a] = par2dir(C,B,A)
%    b = numerator polynomial coefficients of DIRECT form
%    a = denominator polynomial coefficients of DIRECT form
%    C = Polynomial part of PARALLEL form
%    B = K by 2 matrix of real coefficients containing bk's
%    A = K by 3 matrix of real coefficients containing ak's
%
[K,L] = size(A); R = []; P = [];

for i = 1:1:K
[r,p,k]=residuez(B(i,:),A(i,:)); R = [R;r], P = [P;p];
end
[b,a] = residuez(R,P,C); b = b(:)'; a = a(:)';
```

Example:

What would be the overall direct, cascade, or parallel form if a structure contains a combination of these forms?




```
>> [b1,a1]=par2dir(C0,B1,A1)
>> [b2,a2]=par2dir(C0,B2,A2)
>> b=conv(b1,b2)  % overall direct form numerator
>> a=conv(a1,a2)   % overall direct form denominator
>> [b0,Bc,Ac]=dir2cas(b,a)  % overall cascade form
>> [C0,Bp,Ap]=dir2par(b,a)  % overall parallel form
```

A cascade of two parallel sections

FIR Filter Structures

FIR Filter Structure

- **Four Different Structures**

  → Different ways of representing the system w.r.t. memory storage

  Direct form

  Cascade form

  Linear-phase form

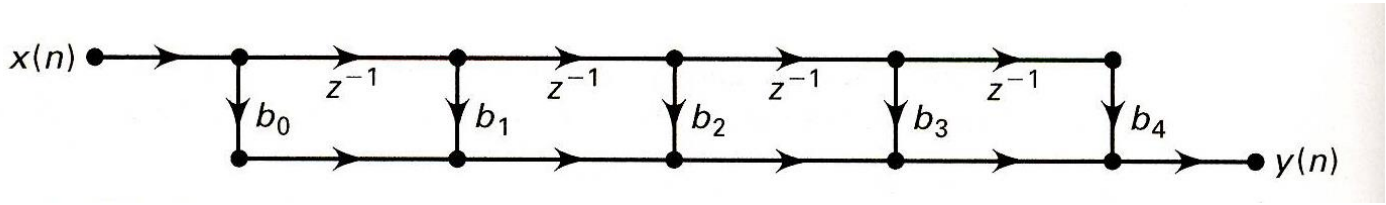
  Frequency sampling form

Direct Form Structure

- **Direct Form**

  → No Feedback

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + b_3 x(n-3) + b_4 x(n-4)$$



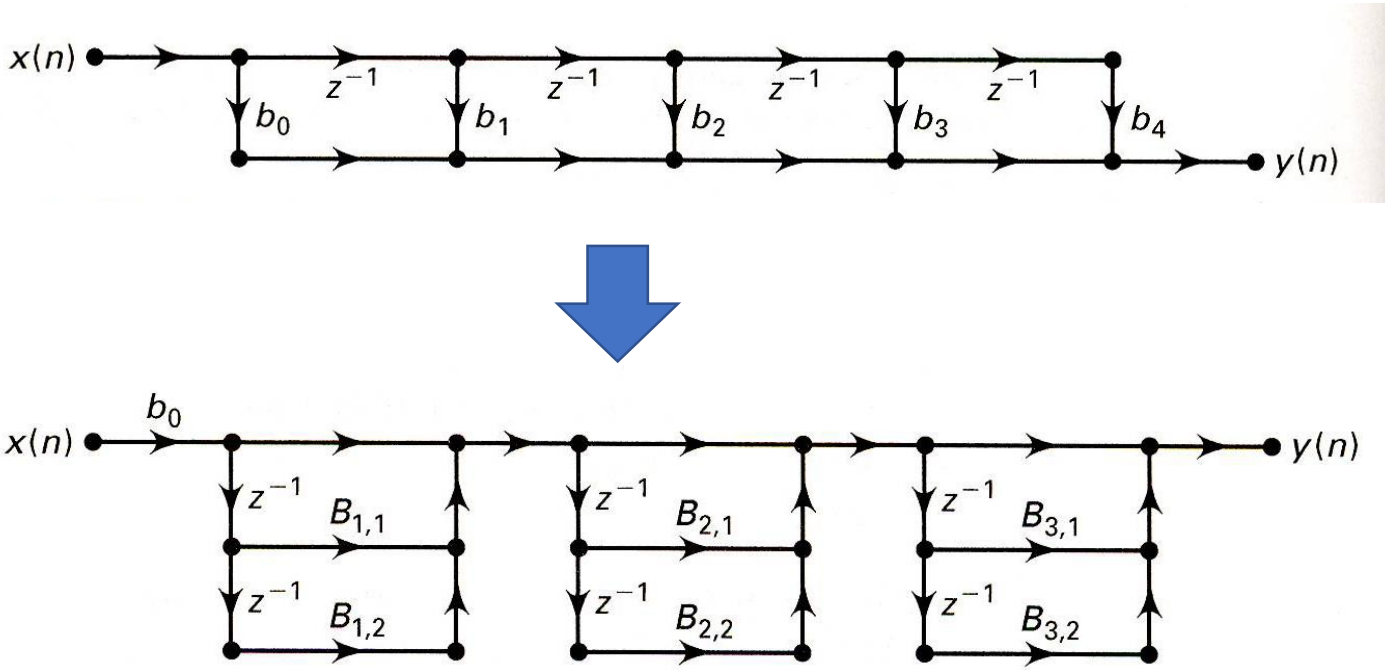
M-1 *memory locations for storing the* *M*-1 *previous inputs,*
M multiplications and *M*-1 *additions per output point*

Cascade Form Structure

- **Cascade Form**

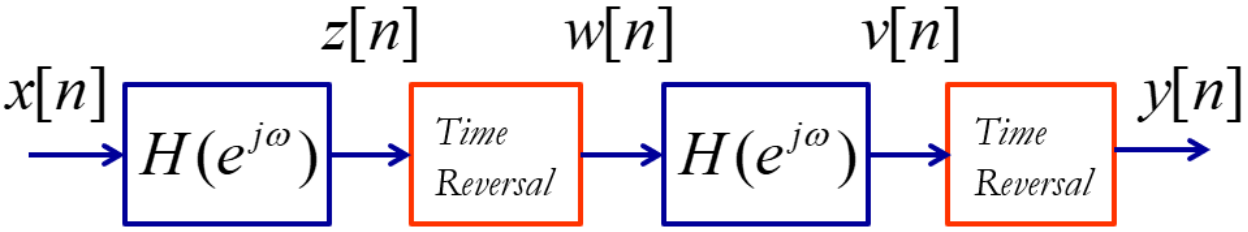
  → Product of second order section

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + b_3 x(n-3) + b_4 x(n-4)$$



Linear-Phase Form Structure

- **Zero-Phase**



$$z[n] = x[n] * h[n] \qquad Z(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega})$$

$$w[n] = z[-n]$$

$$W(e^{j\omega}) = Z^*(e^{j\omega}) = H^*(e^{j\omega})X^*(e^{j\omega})$$

$$v[n] = w[n] * h[n]$$

$$V(e^{j\omega}) = H(e^{j\omega})W(e^{j\omega}) = \left|H(e^{j\omega})\right|^2 X^*(e^{j\omega})$$

$$y[n] = v[-n]$$

$$Y(e^{j\omega}) = V^*(e^{j\omega}) = \left|H(e^{j\omega})\right|^2 X(e^{j\omega})$$

real, positive $\rightarrow$ *zero-phase*

Linear-Phase Form Structure

Linear-Phase

• Consider a signal that consists of several frequency components passing through a filter.

1. The **phase delay** (T_p) of the filter is the amount of time delay each frequency component of the signal suffers in going through the filter.

2. The **group delay** (T_g) is the average time delay the composite signal suffers at each frequency.

3. Mathematically,
$$T_p = -\theta(\omega)/\omega \qquad (3)$$
$$T_g = -d\theta(\omega)/d\omega \qquad (4)$$

where $\theta(\omega)$ is the phase angle.

Linear-Phase Form Structure

- **Linear-Phase**

Given: Phase response is $\theta(\mathrm{w}) = -\alpha \mathrm{w}$, where α is constant.

$\therefore$ Phase Delay $\tau_p = \frac{-\theta(\omega)}{\omega} = \frac{-(-\alpha\omega)}{\omega} = \alpha$, which is a constant.

$\therefore$ Group Delay $\tau_g = \frac{-d}{d\omega}\theta(\omega) = \frac{-d}{d\omega}(-\alpha\omega) = \alpha$, which is a constant.

$\therefore \tau_p = \tau_g$

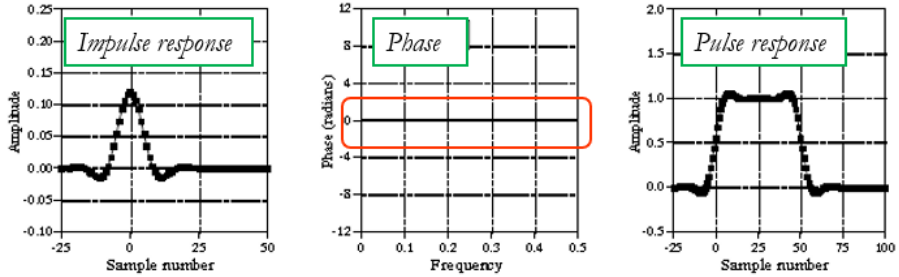
$\therefore$ The given filter is a linear-phase.

Note: For an IIR filter to satisfy the linear phase condition, both the poles and zeros would need to have mirror images outside the unit circle of the z-plane.

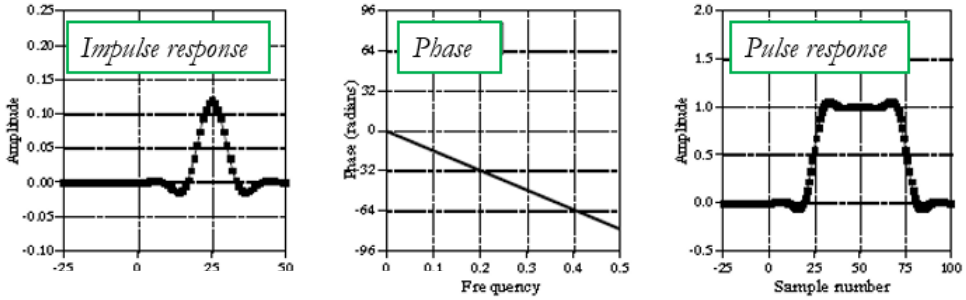
Linear-Phase Form Structure

- **Example**

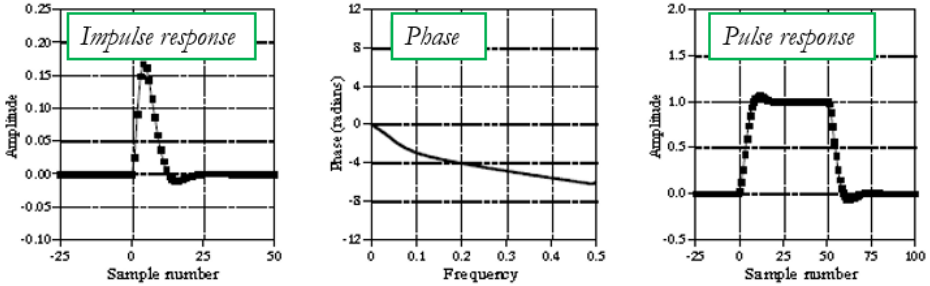
Zero Phase Filter (non-causal); y=filtfilt(b,a,x)



Linear Phase Filter



Nonlinear Phase Filter

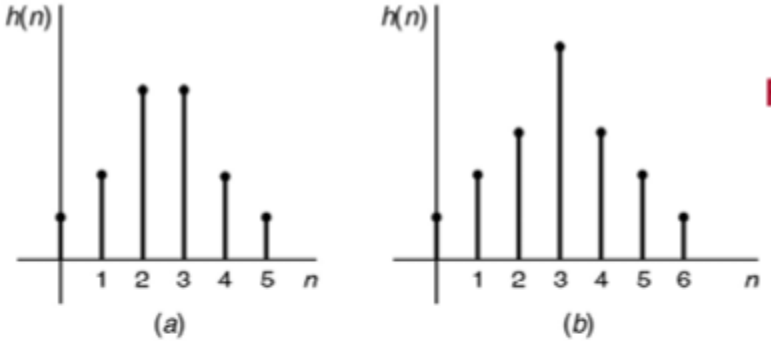


Linear-Phase Form Structure

Linear-Phase Form Structure

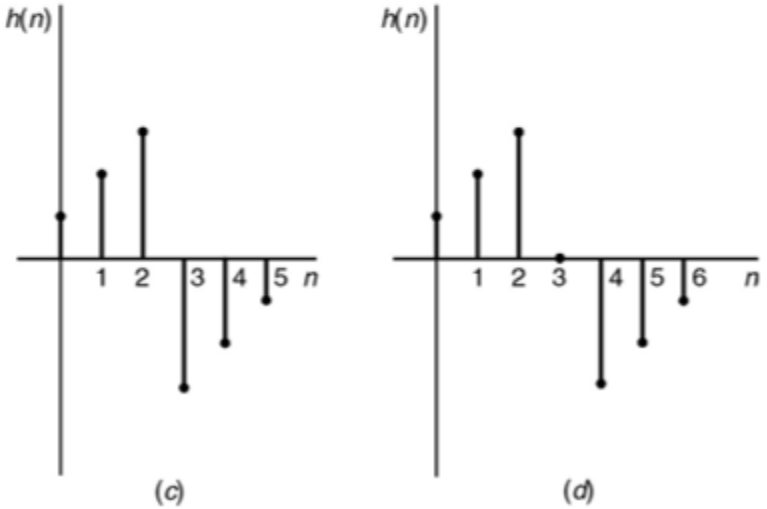
→ Linear phase filter description can be generalised into a formalism for four type of FIR

FIR II: even length, symmetric



FIR I: odd length, symmetric

FIR IV: even length, antisymmetric



FIR III: odd length, antisymmetric

Note for this case that h[M/2]=0

Linear-Phase Form Structure

Linear-Phase Form

desirable to have a phase response that is a linear function of frequency

$$\angle H(e^{j\omega}) = \beta - \alpha\omega, \quad -\pi < \omega \le \pi$$

$$h(n) = h(M-1-n); \quad \beta = 0, \alpha = \frac{M-1}{2}, 0 \le n \le M-1$$

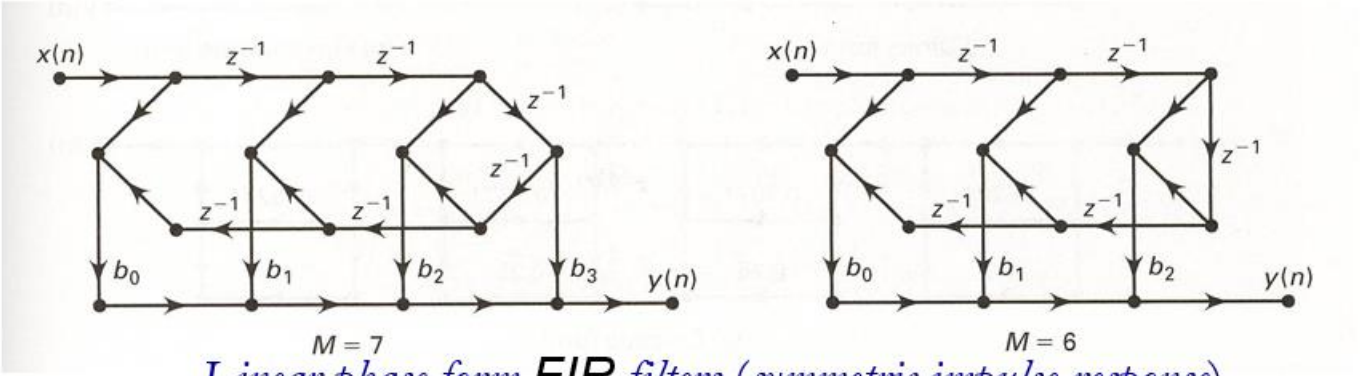
symmetric impulse response

$$h(n) = -h(M-1-n); \quad \beta = \pm\pi/2, \alpha = \frac{M-1}{2}, 0 \le n \le M-1$$

antisymmetric impulse response

$$y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_1 x(n-M+2) + b_0 x(n-M+1)$$
$$= b_0\left[x(n) + x(n-M+1)\right] + b_1\left[x(n-1) + x(n-M+2)\right] + \cdots$$

50% *fewer multiplication* $y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_{M-1} x(n-M+1)$



Linear phase form FIR *filters (symmetric impulse response)*

Example

An FIR filter is given by the system function

$$H(z) = 1 + 16\frac{1}{16}z^{-4} + z^{-8}$$

Direct form

$$y(n) = x(n) + 16.0625\,x(n-4) + x(n-8)$$

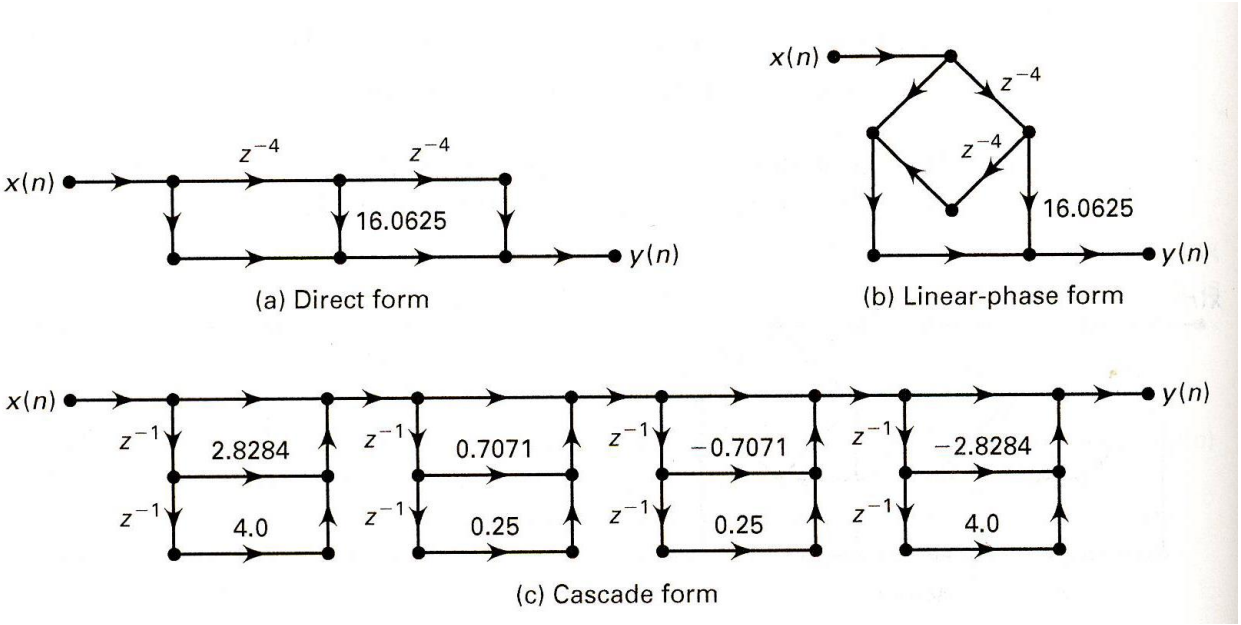
Linear-phase form

$$y(n) = \left[x(n) + x(n-8)\right] + 16.0625\,x(n-4)$$

Example

Cascade form

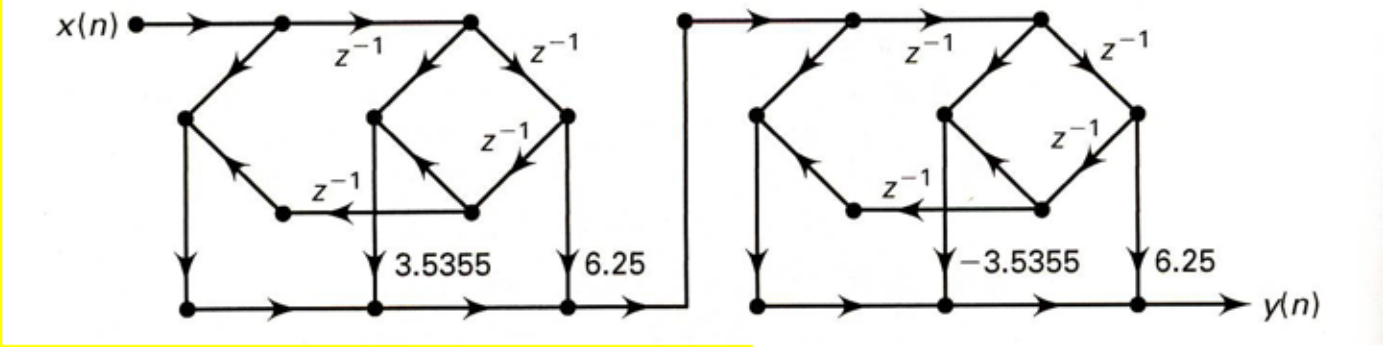
```
>> b=[1,0,0,0,16+1/16,0,0,0,1]; [b0,B,A] = dir2cas(b,1)
b0 = 1
B =
    1.0000   2.8284   4.0000
    1.0000   0.7071   0.2500
    1.0000  -0.7071   0.2500
    1.0000  -2.8284   4.0000
A =
     1     0     0
     1     0     0
     1     0     0
     1     0     0
```



Example

What would be the structure if we desire a cascade form containing linear-phase components with real coefficients?

```
>> b=[1,0,0,0,16+1/16,0,0,0,1]; broots=roots(b)
broots =
      -1.4142 + 1.4142i
      -1.4142 – 1.4142i
       1.4142 + 1.4142i
       1.4142 – 1.4142i
      -0.3536 + 0.3536i
      -0.3536 – 0.3536i
       0.3536 + 0.3536i
       0.3536 – 0.3536i
>> B1=real(poly([broots(1),broots(2),broots(5),broots(6)]))
B1 =
      1.0000      3.5355      6.2500      3.5355      1.0000
>> B2=real(poly([broots(3),broots(4),broots(7),broots(8)]))
B2 =
      1.0000     -3.5355      6.2500     -3.5355      1.0000
```



M-1 *zeros possess certain symmetries:*

$$z = r\angle\theta,\ (1/r)\angle\theta,$$
$$r\angle-\theta,\ (1/r)\angle-\theta$$

Frequency Sampling Form Structure

The system function $H(z)$ *of an FIR filter can be reconstructed from its samples on the unit circle. These samples are the* M*-point DFT values* $\{H(k),\ \ 0 \le k \le M-1\}$ *of the* M*-point impulse response.*

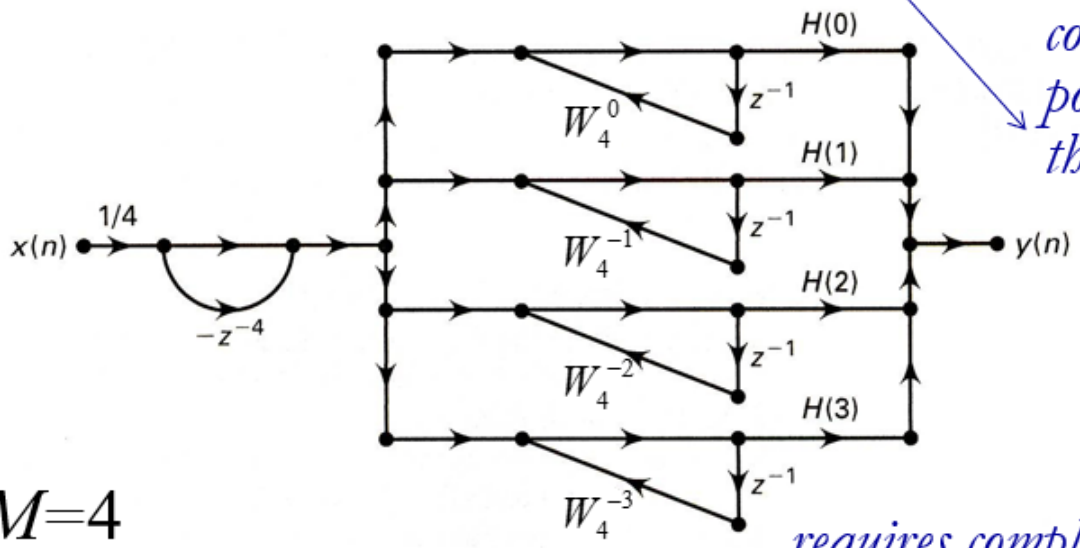
$$H(z) = Z[h(n)] = Z[IDFT\{H(k)\}]$$

$$H(z) = \left(\frac{1-z^{-M}}{M}\right)\sum_{k=0}^{M-1}\frac{H(k)}{1-W_M^{-k}z^{-1}}$$

z-domain reconstruction formula

contains both poles and zeros, poles at W_M^{-k} *are canceled by the roots of* $1-z^{-M}=0$

FIR filter



$M=4$

requires complex arithmetic implementation

Frequency Sampling Form Structure

$$h(n) = \frac{1}{M}\sum_{k=0}^{M-1} H(k) W_M^{-nk}$$

$$H(z) = \sum_{n=0}^{M-1} h(n) z^{-n}$$

$$= \sum_{n=0}^{M-1}\left[\frac{1}{M}\sum_{k=0}^{M-1} H(k) W_M^{-nk}\right] z^{-n}$$

$$= \sum_{k=0}^{M-1} H(k)\left[\frac{1}{M}\sum_{n=0}^{M-1}\left(W_M^{-nk} z^{-n}\right)\right]$$

$$= \sum_{k=0}^{M-1} H(k)\left[\frac{1}{M}\sum_{n=0}^{M-1}\left(W_M^{-k} z^{-1}\right)^n\right]$$

$$W_M^{-kM} = 1$$

$$= \sum_{k=0}^{M-1} H(k)\left[\frac{1}{M}\frac{1 - W_M^{-kM} z^{-M}}{1 - W_M^{-k} z^{-1}}\right]$$

$$= \frac{1 - z^{-M}}{M}\sum_{k=0}^{M-1}\frac{H(k)}{1 - W_M^{-k} z^{-1}}$$

Frequency Sampling Form Structure

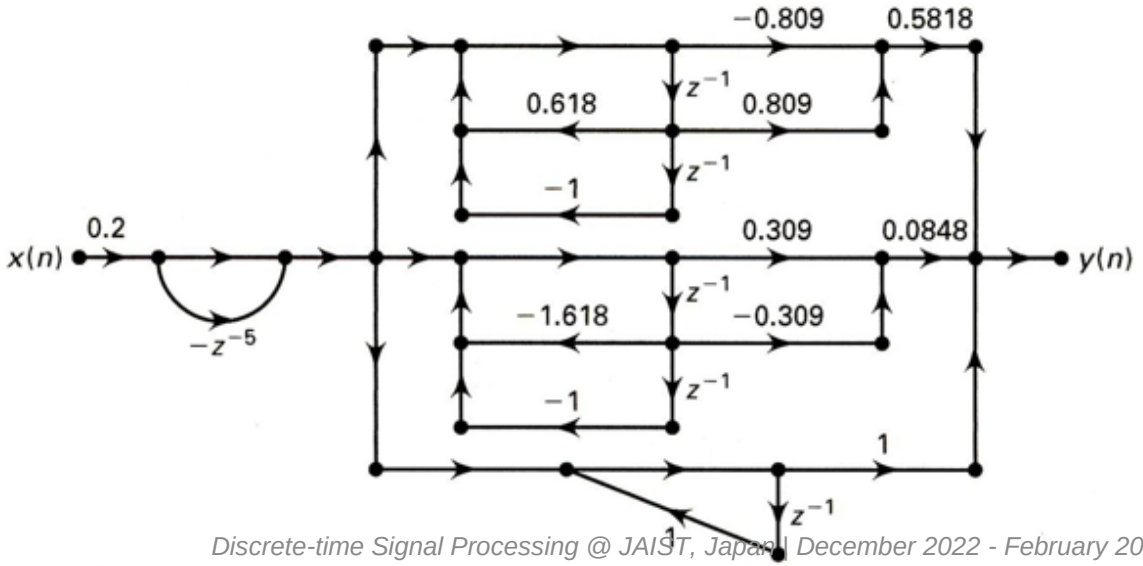
```
function [C,B,A] = dir2fs(h)
% Direct form to Frequency Sampling form conversion
% ------------------------------------------------------------
% [C,B,A] = dir2fs(h)
%  C = Row vector containing gains for parallel sections
%  B = Matrix containing numerator coefficients arranged in rows
%  A = Matrix containing denominator coefficients arranged in rows
%  h = impulse response vector of an FIR filter
%
M = length(h);     H = fft(h,M);
magH =abs(H);     phaH = angle(H)';
% check even or odd M
if (M == 2*floor(M/2))
        L = M/2-1;     % M is even
      A1 = [1,-1,0;1,1,0];     C1 = [real(H(1)),real(H(L+2))];
else
        L = (M-1)/2;     % M is odd
      A1 = [1,-1,0];     C1 = [real(H(1))];
end
k = [1:L]';
% initialize B and A arrays
B = zeros(L,2);     A = ones(L,3);
% compute denominator coefficients
A(1:L,2) = -2*cos(2*pi*k/M);      A = [A;A1];
% compute numerator coefficients
B(1:L,1) = cos(phaH(2:L+1));
B(1:L,2) =-cos(phaH(2:L+1)-(2*pi*k/M));
% compute gain coefficients
C = [2*magH(2:L+1),C1]';
```

Example:

$$h(n) = \frac{1}{9}\left\{1, \underset{\uparrow}{} \quad 2, \quad 3, \quad 2, \quad 1\right\}$$

```
>> h = [1,2,3,2,1]/9;   [C,B,A] = dir2fs(h)
C =
    0.5818
    0.0849
    1.0000
B =
   -0.8090    0.8090
    0.3090   -0.3090
A =
    1.0000   -0.6180    1.0000
    1.0000    1.6180    1.0000
    1.0000   -1.0000         0
```

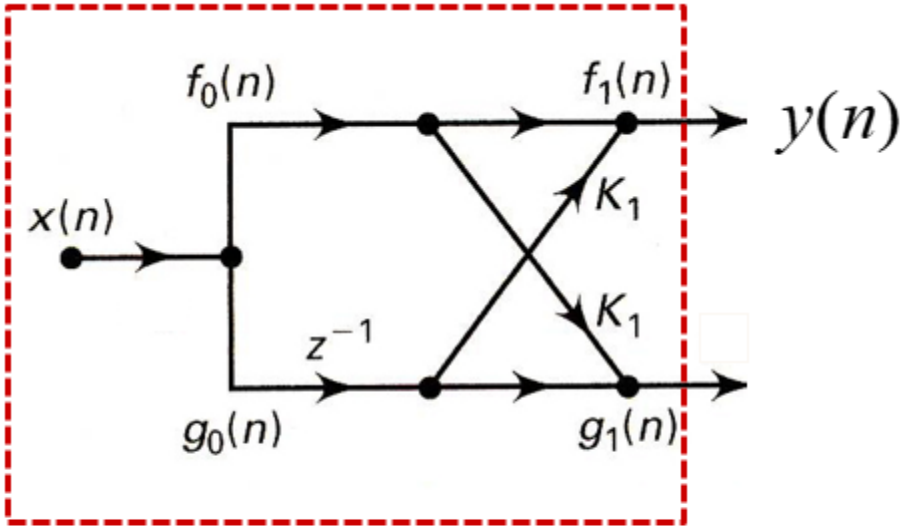
$$H(z) = \frac{1-z^{-5}}{5}\left[0.5818\,\frac{-0.809+0.809z^{-1}}{1-0.618z^{-1}+z^{-2}} + 0.0848\,\frac{0.309-0.309z^{-1}}{1+1.618z^{-1}+z^{-2}} + \frac{1}{1-z^{-1}}\right]$$



Lattice Filter Structures

FIR Lattice Filter Structure (All-Zero)

- **Single-stage Lattice**

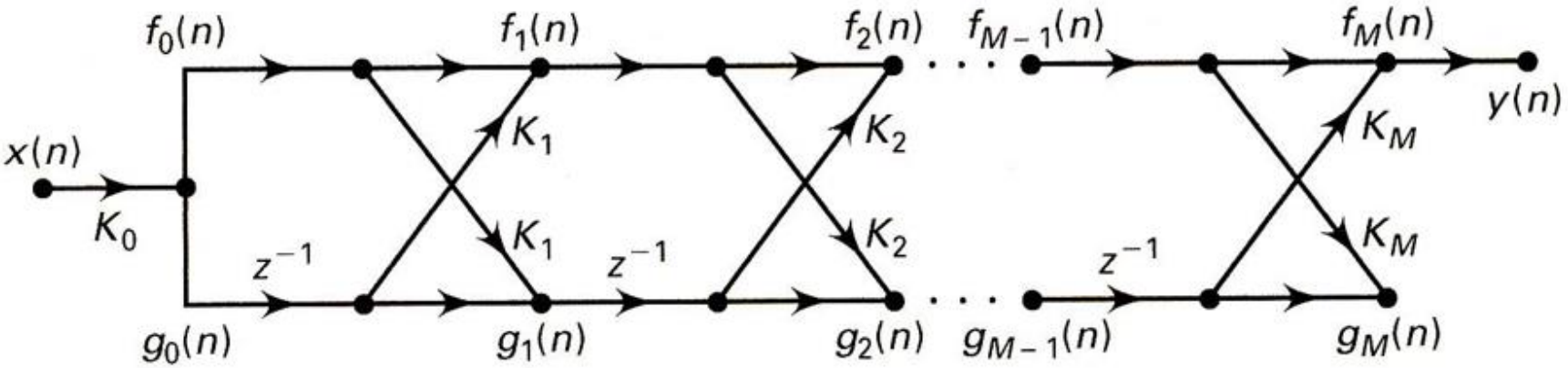


$$f_1(n) = f_0(n) + K_1 g_0(n-1) = x(n) + K_1 x(n-1)$$

$$g_1(n) = K_1 f_0(n) + g_0(n-1) = K_1 x(n) + x(n-1)$$

FIR Lattice Filter Structure

- **General Lattice Structure**



$$f_0(n) = g_0(n) = K_0 x(n)$$
$$y(n) = f_{M-1}(n)$$

The output of the $(M\text{-}1)$ *stage lattice filter corresponds to the output of an* $(M\text{-}1)$ *order FIR filter*

$$f_m(n) = f_{m-1}(n) + K_m g_{m-1}(n-1), \quad m = 1,2,\ldots,M-1$$
$$g_m(n) = K_m f_{m-1}(n) + g_{m-1}(n-1), \quad m = 1,2,\ldots,M-1$$

FIR Lattice Filter Structure

```
function [K] = dir2latc(b)
% FIR Direct form to All-Zero Lattice form conversion
% ------------------------------------------------------------
% [K] = dir2latc(b)
%  K = Lattice filter coefficients (reflection coefficients)
%  b = FIR direct form coefficients (impulse response)
%
M = length(b); K = zeros(1,M); b1 = b(1);
if b1 == 0
error('b(1) is equal to zero')
end
K(1) = b1; A = b/b1;
for m=M:-1:2
K(m) = A(m); J= fliplr(A);
A = (A-K(m)*J)/(1-K(m)*K(m)); A = A(1:m-1);
end
```

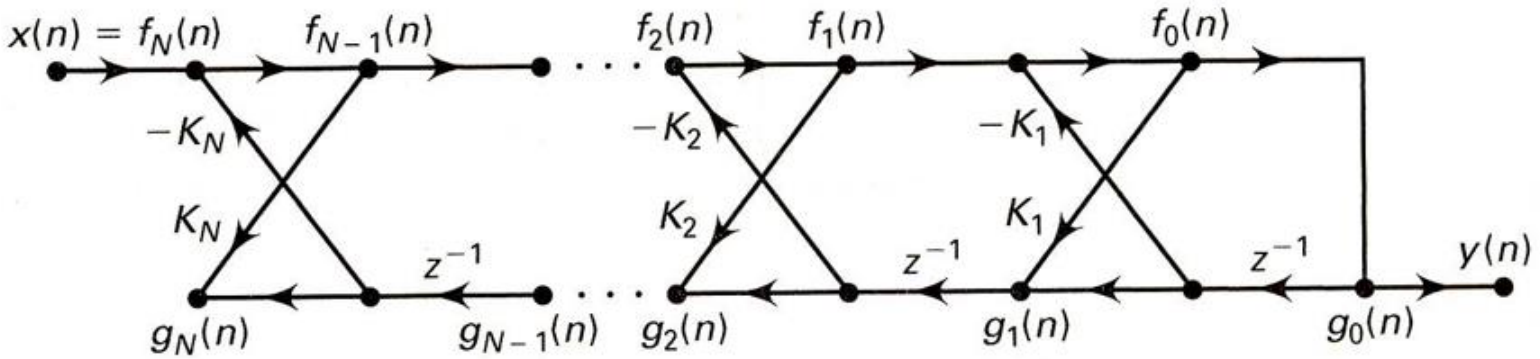
Given the coefficients $\{b_m\}$ *of the direct form, obtain the lattice filter coefficients* $\{K_m\}$.

```
function [y] = latcfilt(K,x)
% Lattice form realization of FIR filters
% ----------------------------------------
% y = latcfilt(K,x)
%    y = output sequence
%    K = Lattice filter (reflection) coefficient array
%    x = input sequence
%
Nx = length(x)-1; x = K(1)*x;
M = length(K)-1; K = K(2:M+1); fg = [x; [0 x(1:Nx)]];
for m = 1:M
      fg = [1,K(m);K(m),1]*fg;
      fg(2,:) = [0 fg(2,1:Nx)];
end
y = fg(1,:)
```

IIR Lattice Filter Structure (All-Pole)

$$H(z) = \frac{1}{1 + \sum_{m=1}^{N} a_N(m) z^{-m}}$$

An inverse to the FIR lattice



$$f_N(n) = x(n)$$

$$f_{m-1}(n) = f_m(n) - K_m g_{m-1}(n-1), \quad m = N, N-1, \ldots, 1$$

$$g_m(n) = K_m f_{m-1}(n) + g_{m-1}(n-1), \quad m = N, N-1, \ldots, 1$$

$$y(n) = f_0(n) = g_0(n)$$

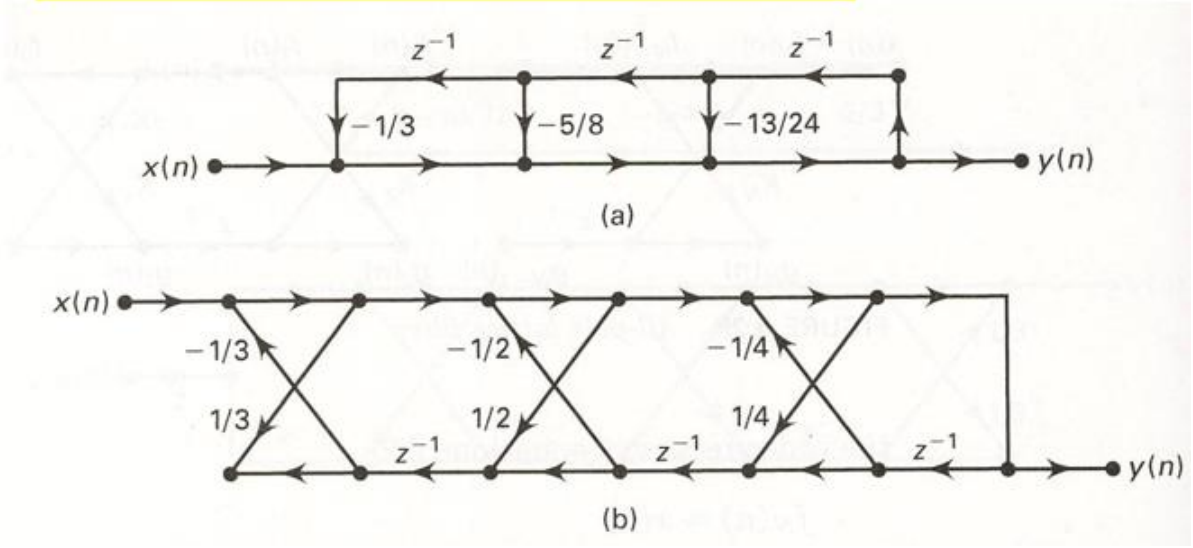
IIR Lattice Filter Structure (All-Pole)

Consider an all-pole **IIR** *filter given by*

$$H(z) = \frac{1}{1 + \frac{13}{24}z^{-1} + \frac{5}{8}z^{-2} + \frac{1}{3}z^{-3}}$$

Determine its lattice structure.

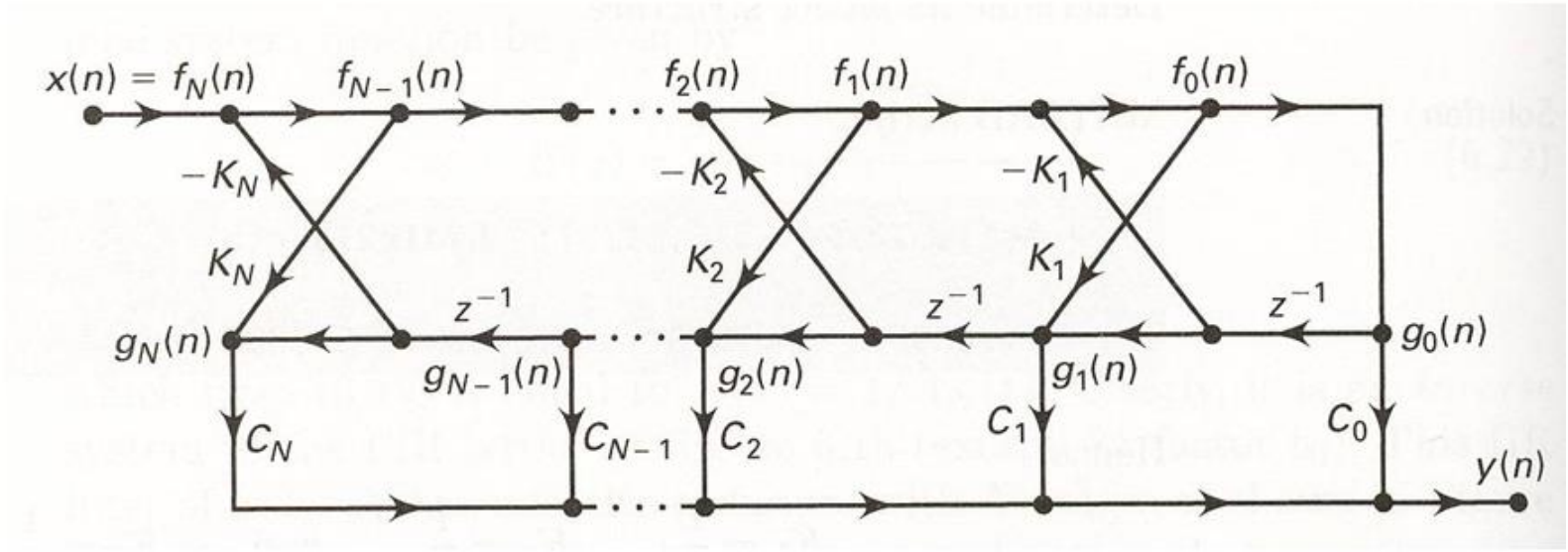
```
>> a=[1, 13/24, 5/8, 1/3]; K=dir2latc(a)
K =
    1.0000    0.2500    0.5000    0.3333
```



IIR Lattice Filter Structure

$$H(z) = \frac{\sum_{k=0}^{M} b_M(k) z^{-k}}{1 + \sum_{k=1}^{N} a_N(k) z^{-k}} = \frac{B_M(z)}{A_N(z)}$$

Use an all-pole lattice as the basic building block



Add a ladder part by taking the output as a weighted linear combination of $\{g_m(n)\}$.

IIR Lattice Filter Structure

```
function [K,C] = dir2ladr(b,a)
% IIR Direct form to pole-zero Lattice/Ladder form conversion
% ------------------------------------------------------------------------
% [K,C] = dir2ladr(b,a)
%  K = Lattice coefficients (reflection coefficients), [K1,…,KN]
%  C = Ladder coefficients, [C0,…,CN]
%  b = Numerator polynomial coefficients (deg <= Num deg)
%  a = Denominator polynomial coefficients
%
a1 = a(1); a = a/a1; b = b/a1;
M = length(b); N = length(a);
if M > N
       error('  *** length of b must be <= length of a  ***')
end
b = [b, zeros(1,N-M)]; K = zeros(1,N-1);
A = zeros(N-1,N-1); C = b;
for m = N-1:-1:1
       A(m,1:m) = -a(2:m+1)*C(m+1);
       K(m) = a(m+1); J = fliplr(a);
       a = (a-K(m)*J)/(1-K(m)*K(m)); a = a(1:m);
       C(m) = b(m) + sum(diag(A(m:N-1,1:N-m)));
end
```

IIR Lattice Filter Structure

```
function [b,a] = ladr2dir(K,C)
% Lattice/Ladder form to IIR Direct form conversion
% ------------------------------------------------------------
% [b,a] = ladr2dir(K,C)
%  b = Numerator polynomial coefficients
%  a = Denominator polynomial coefficients
%  K = Lattice coefficients (reflection coefficients)
%  C = Ladder coefficients
%
N = length(K); M = length(C);
C = [C, zeros(1,N-M+1)];
J = 1; a = 1; A = zeros(N,N);
for m = 1:1:N
      a = [a,0]+conv([0,K(m)],J);
      A(m,1:m) = -a(2:m+1); J = fliplr(a);
end
b(N+1) = C(N+1);
for m = N:-1:1
      A(m,1:m) = A(m,1:m)*C(m+1);
      b(m) = C(m) – sum(diag(A(m:N,1:N-m+1)));
end
```

IIR Lattice Filter Structure

```
function [y] = ladrfilt(K,C,x)
% Lattice/Ladder form realization of IIR filters
% ------------------------------------------------
% [y] = ladrfilt(K,C,x)
%  y = output sequence
%  K = Lattice (reflection) coefficients
%  C = Ladder coefficient array
%  x = input sequence
%
Nx = length(x); y = zeros(1,Nx);
N = length(C); f = zeros(N,Nx); g = zeros(N,Nx+1);
f(N,:) = x;
for n = 2:1:Nx+1
      for m = N:-1:2
            f (m-1,n-1) = f(m,n-1) – K(m-1)*g(m-1,n-1);
            g(m,n) = K(m-1)*f(m-1,n-1) + g(m-1,n-1);
      end
      g(1,n) = f(1,n-1);
end
y = C*g(:,2:Nx+1);
```

Note

SP Toolbox tf2latc, latc2tf

compute all-pole lattice, all-zero lattice, and lattice-ladder structure coefficients, and vice versa.

SP Toolbox latcfilt

implements the all-zero lattice structure.

Does not provide a function to implement the lattice-ladder structure.

Thank you


JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY
1990