Discrete Time FIR Filtering

Maxx Seminario

University of Nebraska-Lincoln

Fall 2025

Introduction: FIR Filters in Digital Signal Processing

What are FIR Filters?

- **Finite Impulse Response (FIR):** Filter output depends only on current and past inputs
- Impulse response has *finite* duration (eventually becomes zero)
- Fundamental building block in discrete-time signal processing

General Form:

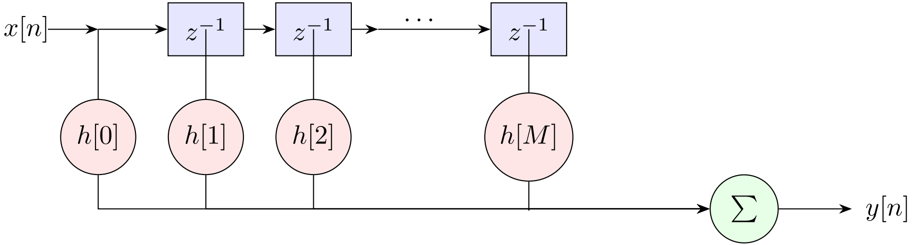
$$y[n] = \sum_{k=0}^{M} b_k x[n-k] = b_0 x[n] + b_1 x[n-1] + \cdots + b_M x[n-M]$$

where $h[n] = b_n$ for $0 \leq n \leq M$ (impulse response coefficients)

Key Advantages:

- **Always stable** (finite sum cannot diverge)
- **Exact linear phase possible** (symmetric delay, no distortion)
- **Simple structure** (no feedback, easy to implement)

Discrete-Time Implementation of FIR Filters



Difference Equation:

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

Computational Cost (per output sample):

- **Multiplications:** $M + 1$
- **Additions:** M
- **Memory:** M delay elements + $(M + 1)$ coefficients

Four Types of Linear-Phase FIR Filters

Classification by Symmetry and Length:

Type	Length	Symmetry	Applications
I	$(M+1)$ odd M even	Symmetric $h[n] = h[M-n]$	Lowpass, highpass, bandpass Most versatile
II	$(M+1)$ even M odd	Symmetric $h[n] = h[M-n]$	Lowpass, bandpass NOT for highpass
III	$(M+1)$ odd M even	Antisymmetric $h[n] = -h[M-n]$	Differentiators Hilbert transformers
IV	$(M+1)$ even M odd	Antisymmetric $h[n] = -h[M-n]$	Differentiators Hilbert transformers

Key Distinction:

- **Symmetric** (Types I & II): Real-valued frequency response
- **Antisymmetric** (Types III & IV): Imaginary frequency response

Type I: Symmetric, Odd Length

DSP

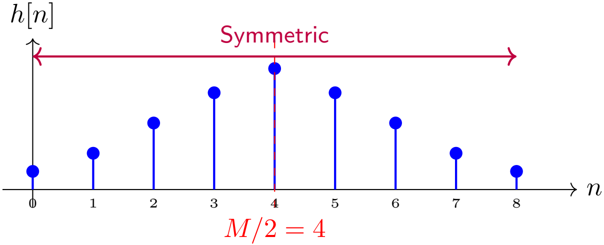
Maxx Seminario

Introduction

Implementation

Linear Phase FIR Filter Types

Impulse Response Example ($M = 8$, length = 9):

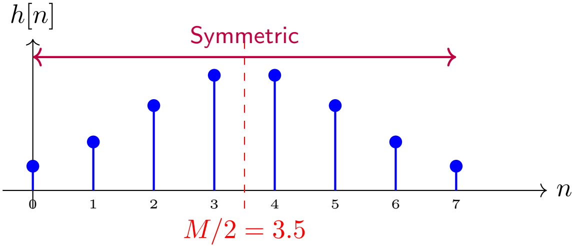


Frequency Response:

$$H(e^{j\omega}) = e^{-j\omega M/2} \underbrace{\left[h[M/2] + 2\sum_{n=1}^{M/2} h[M/2-n]\cos(\omega n) \right]}_{\text{Real amplitude } A_e(e^{j\omega})}$$

Type II: Symmetric, Even Length

Impulse Response Example ($M = 7$, length = 8):



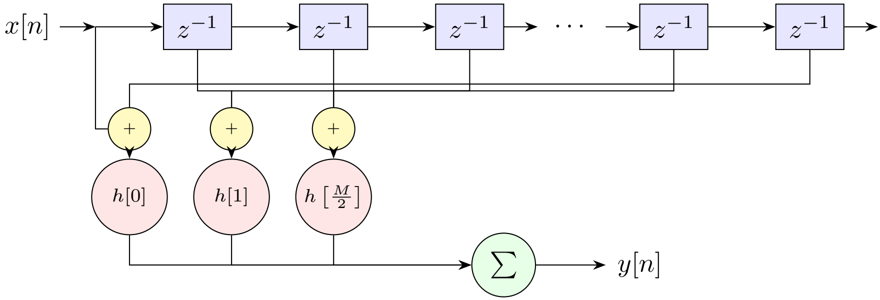
Frequency Response:

$$H(e^{j\omega}) = e^{-j\omega M/2}\cos(\omega/2)\underbrace{P(\cos\omega)}_{\text{Polynomial}}$$

Limitation: $\cos(\pi/2) = 0$ forces $H(e^{j\pi}) = 0$ (cannot realize highpass)

Efficient Implementation: Exploiting Symmetry

For Symmetric FIR Filters (Types I & II):
Since $h[n] = h[M-n]$, we can reduce computations by 50%

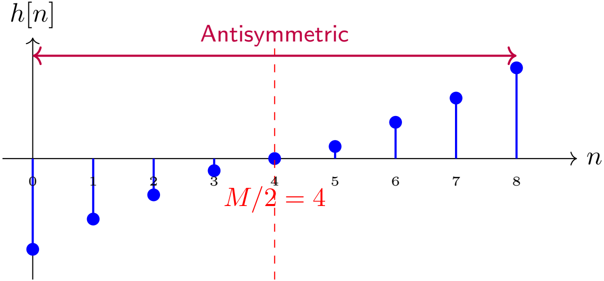


Optimized Cost: Trading Multiplies for Additions

- **Multiplications:** $\lceil (M+1)/2 \rceil$ (50% reduction!)
- **Additions:** M (pre-additions) + $\lceil (M+1)/2 \rceil - 1$ (post-additions)
- **Memory:** Still M delays + $(M+1)$ coefficients

Type III: Antisymmetric, Odd Length

Impulse Response Example ($M = 8$, length = 9):



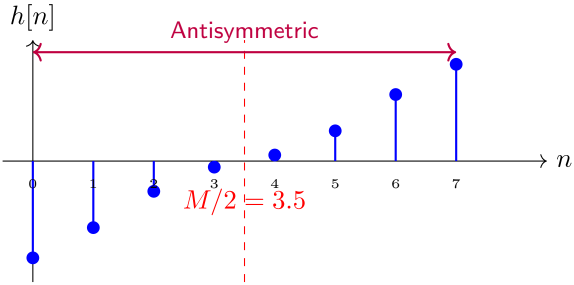
Frequency Response:

$$H(e^{j\omega}) = je^{-j\omega M/2}\sin(\omega)P(\cos\omega)$$

Limitation: $\sin(0) = 0$ forces $H(e^{j\pi 0}) = 0$ (cannot realize lowpass)

Type IV: Antisymmetric, Even Length

Impulse Response Example ($M = 7$, length = 8):



Frequency Response:

$$H(e^{j\omega}) = je^{-j\omega M/2}\sin(\omega/2)P(\cos\omega)$$

Limitation: $\sin(0) = 0$ forces $H(e^{j\pi(0)}) = 0$ (cannot realize lowpass)

Comparison: All Four Types

Property	Type I	Type II	Type III	Type IV
M	Even	Odd	Even	Odd
Length $(M+1)$	Odd	Even	Odd	Even
Symmetry	Sym	Sym	Antisym	Antisym
$h[M/2]$	Non-zero	N/A	Zero	N/A
$H(0)$	Any	Any	Zero	Non-zero
$H(\pi)$	Any	Zero	Zero	Non-zero

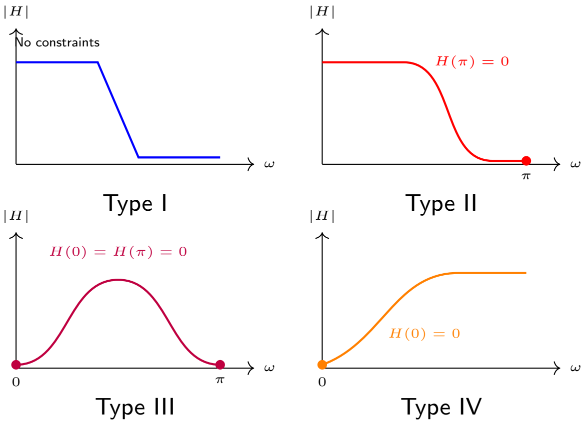
Lowpass	$\checkmark$	$\checkmark$	$\times$	$\times$
Highpass	$\checkmark$	$\times$	$\times$	$\checkmark$
Bandpass	$\checkmark$	$\checkmark$	$\times$	$\checkmark$
Differentiator	$\times$	$\times$	$\checkmark$	$\checkmark$
Hilbert	$\times$	$\times$	$\checkmark$	$\checkmark$

Design Choice:

- Type I: Most versatile, use for general frequency-selective filters
- Type II: Lowpass only (response forced to zero at $\omega = \pi$)
- Types III & IV: Specialized applications (differentiators, Hilbert)

Frequency Response Constraints

Why certain applications are prohibited:



FIR vs IIR Filters: Implementation Comparison

Property	FIR Filters	IIR Filters
Stability	**Always stable** (no feedback)	Can be unstable (pole locations critical)
Phase Response	**Exact linear phase** possible (Types I-IV)	Nonlinear phase (except Bessel approximation)
Filter Order	Higher order needed (typically 10-100+ taps)	**Lower order** (typically 2-10 poles)
Computation	More operations per sample ($M + 1$ multiplies)	**Fewer operations** (typically < 10 multiplies)
Memory	More memory (M delays)	**Less memory** (order of filter)
Finite Wordlength	**Robust** (no limit cycles, low sensitivity)	Sensitive (limit cycles, pole location errors)
Design	**Systematic** (windowing, Parks-McClellan)	More complex (bilinear transform, pole placement)

When to Use FIR vs IIR Filters

Choose FIR When:

- **Linear phase required**
  - Audio processing
  - Image processing
  - Data transmission
- **Stability critical**
  - Safety systems
  - Embedded systems
- **Fixed-point implementation**
  - Low wordlength DSPs
  - Quantization robustness needed

Choose IIR When:

- **Computational efficiency critical**
  - Real-time constraints
  - Low-power devices
  - High sample rates
- **Sharp transitions needed**
  - Narrow stopband
  - Steep rolloff
  - Lower order achievable
- **Memory limited**
  - Small coefficient storage
  - Few delay elements

General rule of thumb:

- Use **FIR** for audio/video (linear phase) and when stability/robustness are critical
- Use **IIR** for control systems and when computational resources are constrained