ELEN E4810: Digital Signal Processing
Topic 7: Filter types and structures

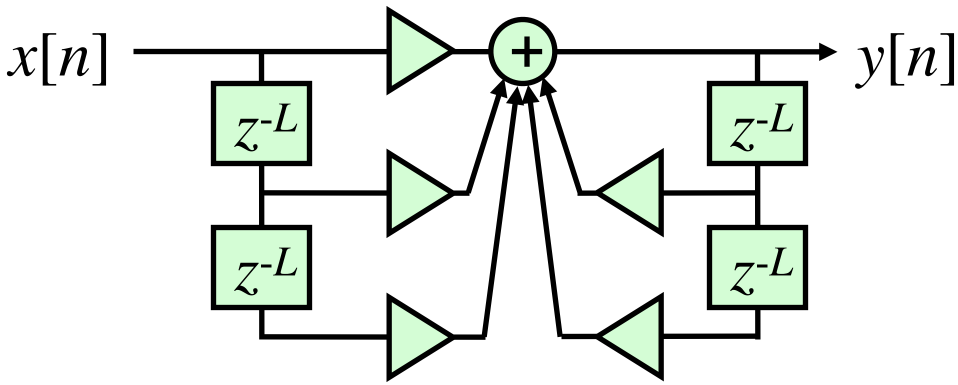
1. More filter types
2. Minimum and maximum phase
3. Filter implementation structures

1. More Filter Types

- We have seen the basics of filters and a range of simple examples
- Now look at a couple of other classes:
  - Comb filters - multiple pass/stop bands
  - Allpass filters - only modify signal phase

Comb Filters

- Replace all system delays z^{-1} with longer delays z^{-L}



→ System that behaves ‘the same’ at a longer timescale

Comb Filters

- 'Parent' filter impulse response $h[n]$ becomes comb filter output as:

$$g[n] = \{h[0]\ \ 0\ \ 0\ \ 0\ \ 0\ \ h[1]\ \ 0\ \ 0\ \ 0\ \ 0\ \ h[2]..\}$$

L-1 *zeros*

- Thus, $G(z) = \sum_n g[n] z^{-n}$

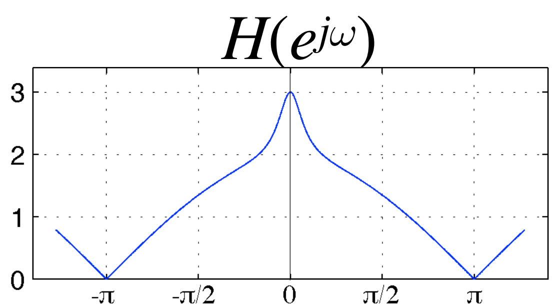
$$= \sum_n h[n] z^{-nL} = H\left(z^L\right)$$

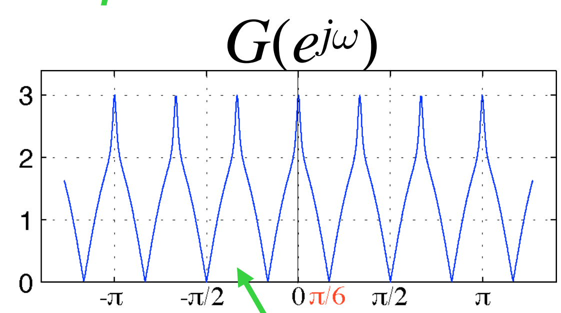
Comb Filters

- Hence frequency response:

$$G(e^{j\omega}) = H(e^{j\omega L})$$

parent frequency response compressed & repeated L times





L copies of $H(e^{j\omega})$

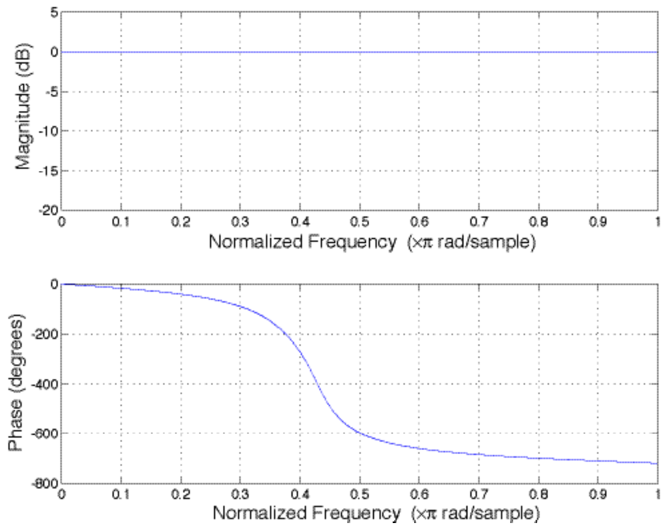
- Low-pass response →
  - pass $\omega = 0, 2\pi/L, 4\pi/L\ldots$
  - cut $\omega = \pi/L, 3\pi/L, 5\pi/L\ldots$

useful to enhance a harmonic series

Allpass Filters

- Allpass filter has $|A(e^{j\omega})|^2 = K$ for all ω

  i.e. spectral energy is not changed
- Phase response is not zero (else trivial)
  - phase correction
  - special effects
- e.g.



Allpass Filters

- Allpass has special form of system fn:

$$A_M(z) = \pm \frac{d_M + d_{M-1}z^{-1} + \ldots + d_1 z^{-(M-1)} + z^{-M}}{1 + d_1 z^{-1} + \ldots + d_{M-1} z^{-(M-1)} + d_M z^{-M}}$$

$$= \pm z^{-M} \frac{D_M\left(z^{-1}\right)}{D_M(z)}$$

=

mirror-image polynomials

- $A_M(z)$ has poles λ where $D_M(\lambda) = 0$

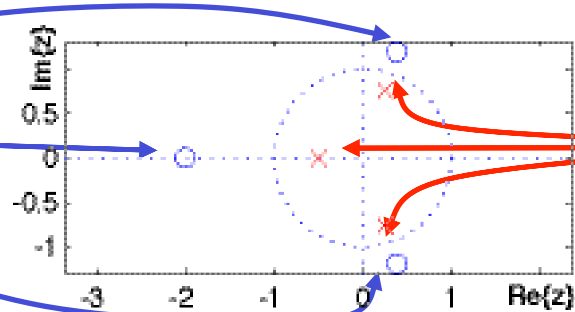
  → $A_M(z)$ has zeros $\zeta = 1/\lambda = \lambda^{-1}$

Allpass Filters

$$A_M(z) = \pm z^{-M} \frac{D_M(z^{-1})}{D_M(z)}$$

- Any (stable) D_M can be used:

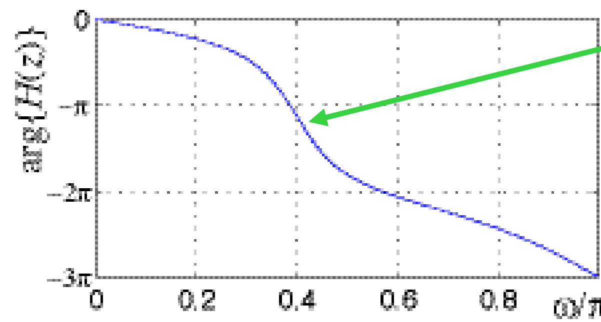
reciprocal zeros from $D_M(z^{-1})$



poles from $1/D_M(z)$

- Phase is always decreasing:

→ $-M\pi$ at $\omega = \pi$



peak group delay

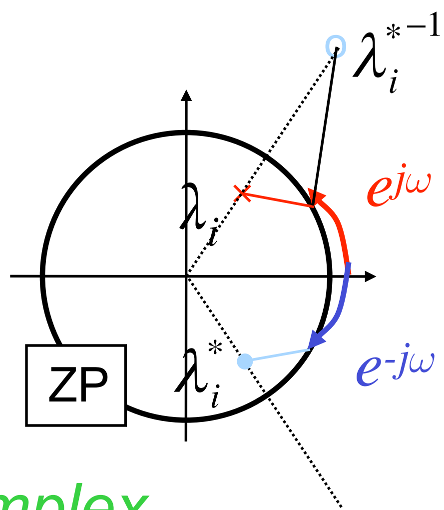
Allpass Filters

Why do mirror-img poly's give const gain?

- Conj-sym system fn can be factored as:

$$A_M(z) = \frac{K\prod_i \left(z - \lambda_i^{*-1}\right)}{\prod_i (z - \lambda_i)}$$

$$= \frac{K\prod_i \lambda_i^{*-1} z\left(\lambda_i^{*} - z^{-1}\right)}{\prod_i (z - \lambda_i)}$$



+ complex conjugate p/z

- $z = e^{j\omega} \rightarrow z^{-1} = e^{-j\omega}$ also on u.circle...

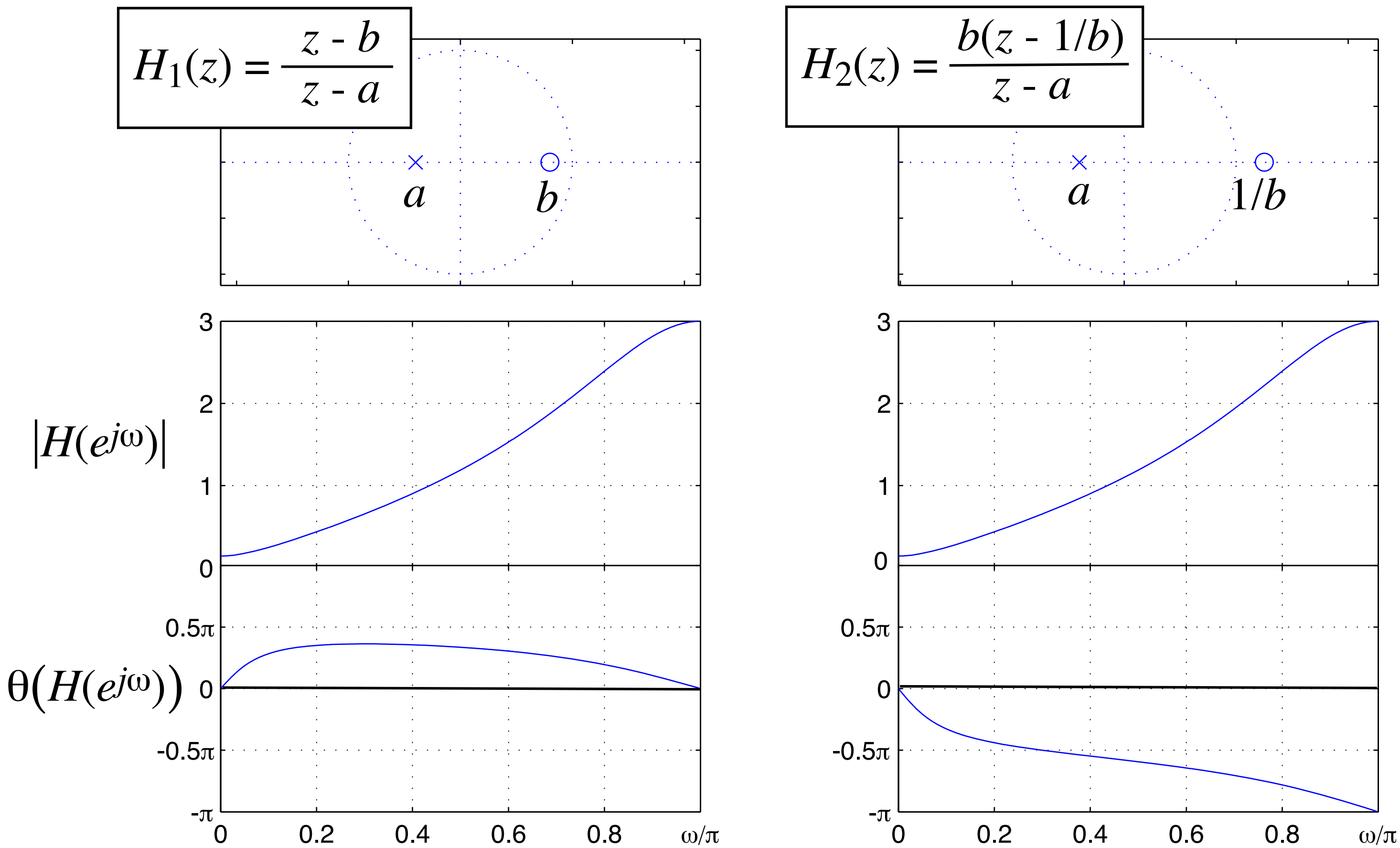
2. Minimum/Maximum Phase

- In AP filters, reciprocal roots have..
  - same effect on magnitude (modulo const.)
  - different effect on phase
- In normal filters, can try substituting reciprocal roots
  - reciprocal of stable pole will be unstable ✕
  - reciprocals of zeros?

→ Variants of filters with same magnitude response, different phase

Minimum/Maximum Phase

- Hence:



reciprocal zero..

.. same mag..

.. added phase lag

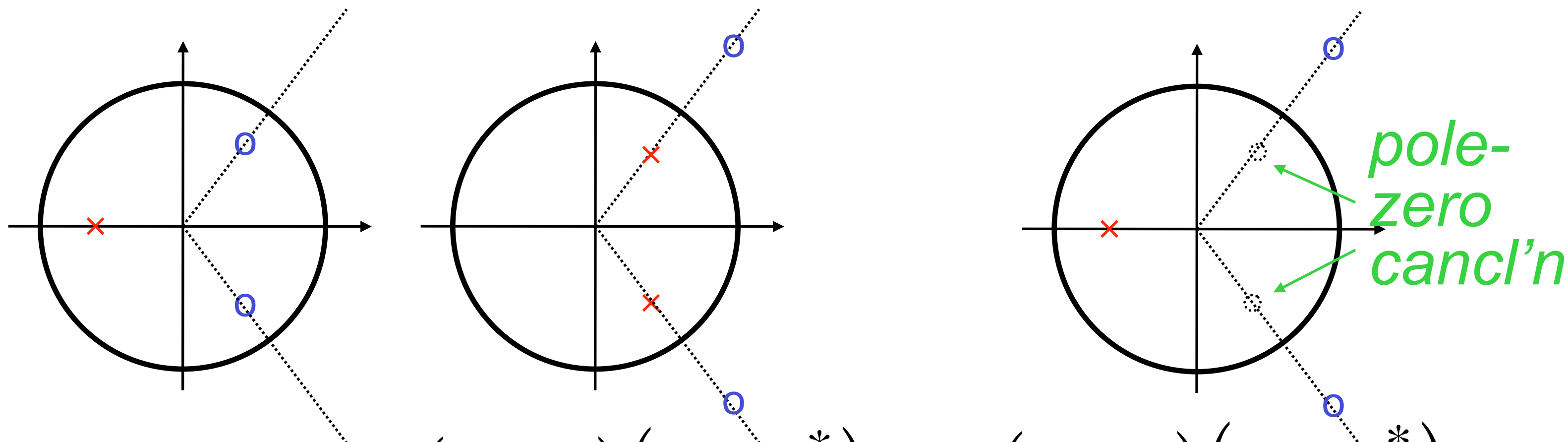
Minimum/Maximum Phase

- For a given magnitude response
  - All zeros *inside* u.circle → minimum phase
  - All zeros *outside* u.c. → maximum phase (greatest phase dispersion for that order)
  - Otherwise, mixed phase
- i.e. for a given magnitude response several filters & phase fns are possible; **minimum phase** is canonical, 'best'

Minimum/Maximum Phase

- Note:

Min. phase + Allpass = Max. phase



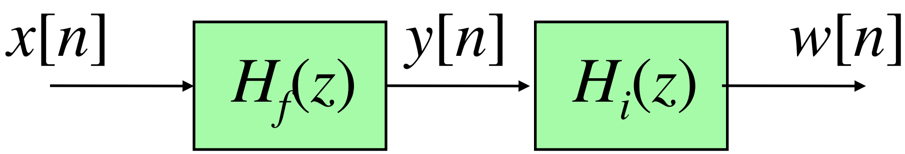
$$\frac{(z-\zeta)(z-\zeta^*)}{z-\lambda} \times \frac{\left(z-\frac{1}{\zeta}\right)\left(z-\frac{1}{\zeta}^*\right)}{(z-\zeta)(z-\zeta^*)} = \frac{\left(z-\frac{1}{\zeta}\right)\left(z-\frac{1}{\zeta}^*\right)}{z-\lambda}$$

Inverse Systems

- $h_i[n]$ is called the inverse of $h_f[n]$ iff

$$h_i[n] \circledast h_f[n] = \delta[n]$$

- Z-transform: $H_f\left(e^{j\omega}\right) \cdot H_i\left(e^{j\omega}\right) = 1$



$$W(z) = H_i(z)Y(z) = H_i(z)H_f(z)X(z) = X(z)$$

$$\Rightarrow w[n] = x[n]$$

- i.e. $H_i(z)$ recovers $x[n]$ from o/p of $H_f(z)$

Inverse Systems

- What is $H_i(z)$?

$$H_i(z)H_f(z)=1$$

$$\Rightarrow H_i(z)=1/H_f(z)$$

- $H_i(z)$ is reciprocal polynomial of $H_f(z)$

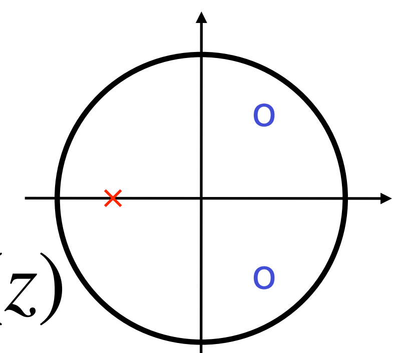
$$H_f(z)=\frac{P(z)}{D(z)} \Rightarrow H_i(z)=\frac{D(z)}{P(z)}$$

poles of fwd → **zeros** of bwd

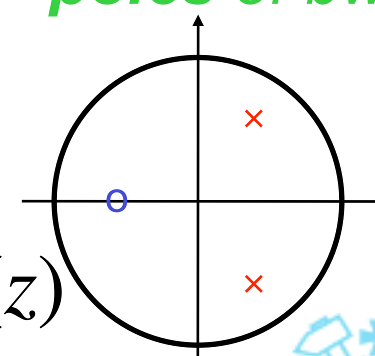
zeros of fwd → **poles** of bwd

- Just swap poles and zeros:

$H_f(z)$



$H_i(z)$

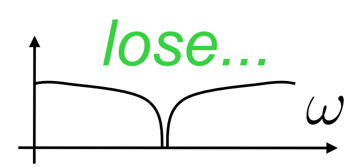


Inverse Systems

When does $H_i(z)$ exist?

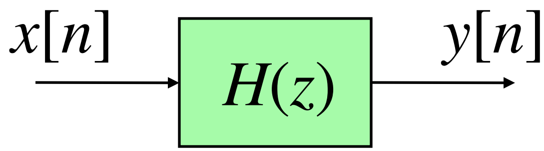
- Causal+stable → all $H_i(z)$ poles inside u.c.
  → all zeros of $H_f(z)$ must be inside u.c.
  → $H_f(z)$ must be minimum phase
- $H_f(z)$ zeros outside u.c. → unstable $H_i(z)$
- $H_f(z)$ zeros **on** u.c. → unstable $H_i(z)$

$$H_i\left(e^{j\omega}\right) = 1/H_f\left(e^{j\omega}\right) = 1/0\big|_{\omega=\zeta}$$



→ only invert if min.phase, ⇒ $H_f(e^{j\omega}) \neq 0$

System Identification



- Inverse filtering = given y and H, find x
- System ID = given y (and ~x), find H
- Just run convolution backwards?

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

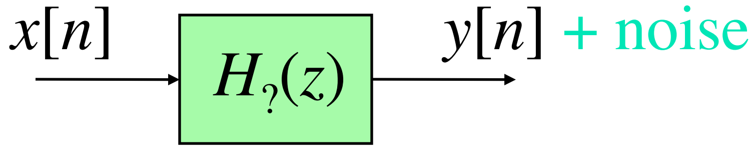
$$\Rightarrow \quad y[0] = h[0]x[0] \quad \rightarrow \quad h[0]$$

$$y[1] = h[0]x[1] + h[1]x[0] \quad \rightarrow \quad h[1] \ldots$$

deconvolution

but: errors accumulate

System Identification



- Better approach uses correlations; Cross-correlate input and output:

$$r_{xy}[\ell] = y[\ell] \circledast x[-\ell] = h_?[\ell] \circledast x[\ell] \circledast x[-\ell]$$

$$= h_?[\ell] \circledast r_{xx}[\ell]$$

- If r_{xx} is ‘simple’, can recover $h_?[n]$...
- e.g. (pseudo-) white noise:

$$r_{xx}[\ell] \approx \delta[\ell] \quad \Rightarrow \quad h_?[n] \approx r_{xy}[\ell]$$

System Identification

- Can also work in frequency domain:

$$S_{xy}(z) = H_?(z) \cdot S_{xx}(z) \longleftarrow \textit{make a const.}$$

- $x[n]$ is not observable → S_{xy} unavailable, but $S_{xx}(e^{j\omega})$ may still be known, so:

$$S_{yy}(e^{j\omega}) = Y(e^{j\omega})Y^*(e^{j\omega})$$
$$= H(e^{j\omega})X(e^{j\omega})H^*(e^{j\omega})X^*(e^{j\omega})$$
$$= \left|H(e^{j\omega})\right|^2 \cdot S_{xx}(e^{j\omega})$$

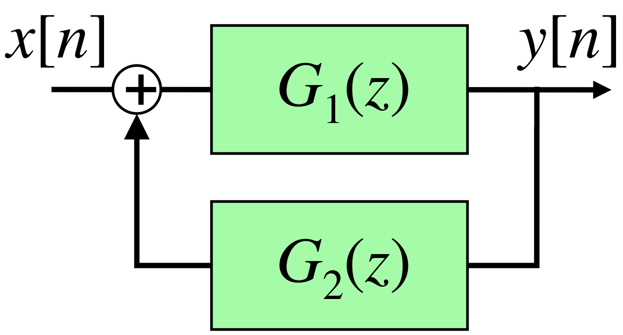
- Use e.g. min.phase to rebuild $H(e^{j\omega})$...

3. Filter Structures

- Many different implementations, representations of same filter
- Different costs, speeds, layouts, noise performance, ...

Block Diagrams

- Useful way to illustrate implementations
- Z-transform helps analysis:



$$Y(z) = G_1(z)\left[X(z) + G_2(z)Y(z)\right]$$

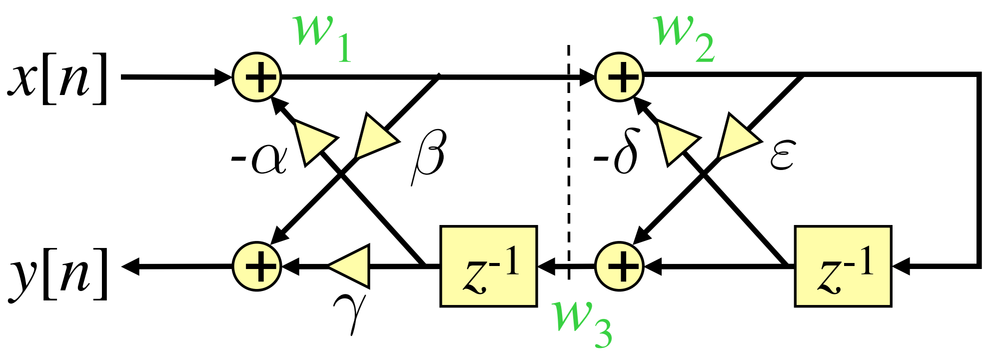
$$\Rightarrow Y(z)\left[1 - G_1(z)G_2(z)\right] = G_1(z)X(z)$$

$$\Rightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{G_1(z)}{1 - G_1(z)G_2(z)}$$

- Approach
  - Output of summers as dummy variables
  - Everything else is just multiplicative

Block Diagrams

- More complex example:



$$W_1 = X - \alpha z^{-1} W_3$$

$$W_2 = W_1 - \delta z^{-1} W_2$$

$$W_3 = z^{-1} W_2 + \varepsilon W_2$$

$$Y = \gamma z^{-1} W_3 + \beta W_1$$

$$\Rightarrow \frac{Y}{X} = \frac{\beta + z^{-1}(\beta\delta + \gamma\varepsilon) + z^{-2}(\gamma)}{1 + z^{-1}(\delta + \alpha\varepsilon) + z^{-2}(\alpha)}$$

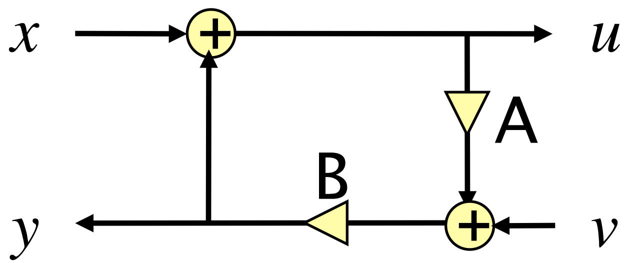
stackable
2nd order section

$$W_2 = \frac{W_1}{1 + \delta z^{-1}}$$

$$W_3 = \frac{(z^{-1} + \varepsilon) W_1}{1 + \delta z^{-1}}$$

Delay-Free Loops

- Can't have them!



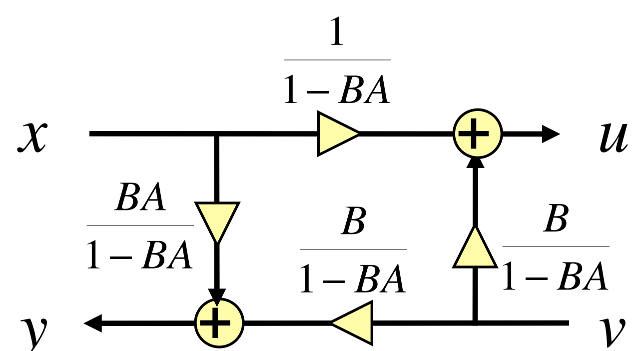
$$y = B(v + Au) \quad u = x + y$$

$$\Rightarrow y = B(v + A(x + y))$$

- At time $n = 0$, setup inputs x and v ; need u for y, also y for u → can't calculate
- Algebra:

$$y(1 - BA) = Bv + BAx$$

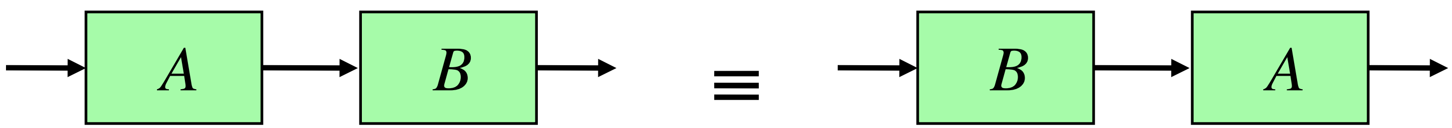
$$\Rightarrow y = \frac{Bv + BAx}{1 - BA}$$



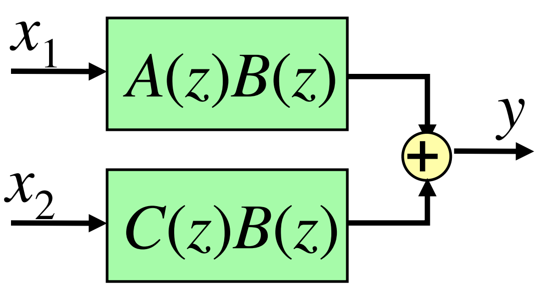
can simplify...

Equivalent Structures

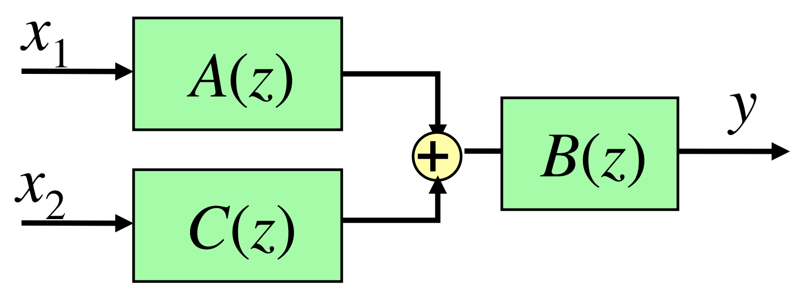
- Modifications to block diagrams that do not change the filter
- e.g. Commutation $H = AB = BA$



- Factoring $AB+CB = (A+C)\cdot B$



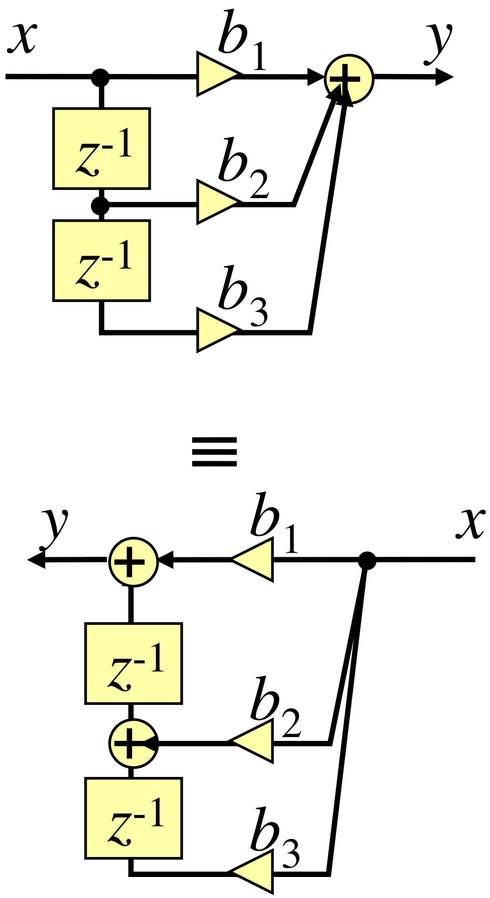
fewer blocks



less computation

Equivalent Structures

- Transpose
  - reverse paths
  - adders↔nodes
  - input↔output

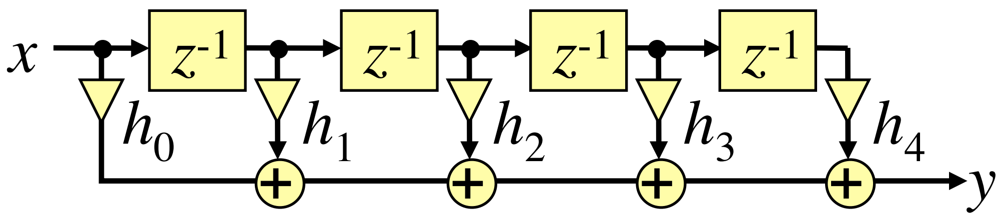


$$Y = b_1 X + b_2 z^{-1} X + b_3 z^{-2} X$$
$$= b_1 X + z^{-1}\left(b_2 X + z^{-1} b_3 X\right)$$

✓

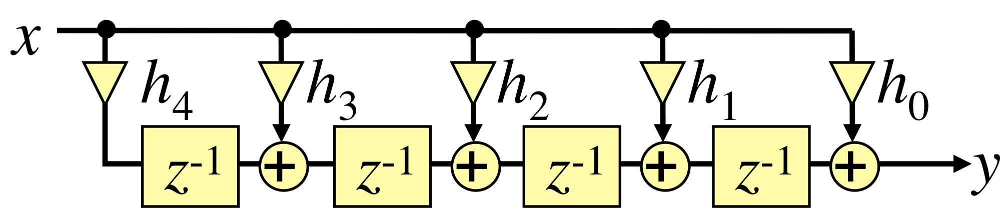
FIR Filter Structures

- Direct form "Tapped Delay Line"



$$y[n] = h_0 x[n] + h_1 x[n-1] + \ldots$$
$$= \sum_{k=0}^{4} h_k x[n-k]$$

- Transpose

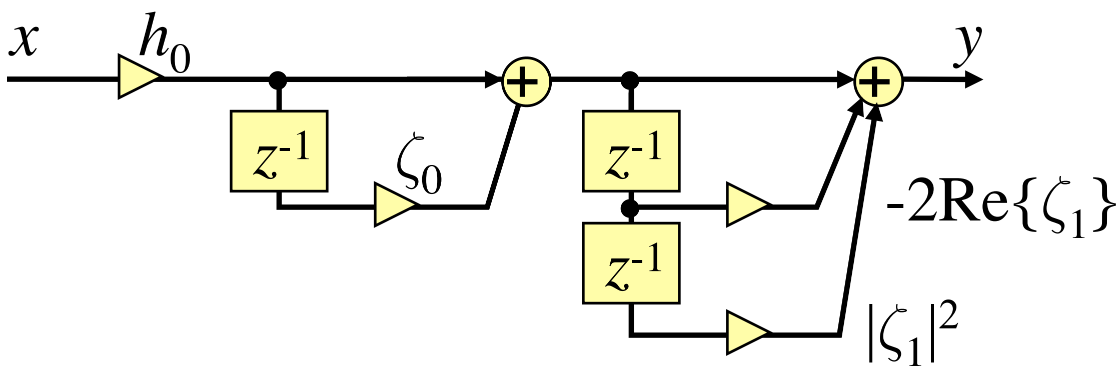


- Re-use delay line if several inputs x_i for single output y ?

FIR Filter Structures

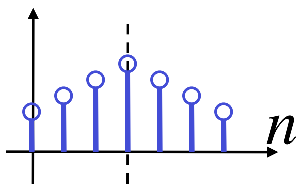
- Cascade
  - factored into e.g. 2nd order sections

$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3}$$
$$= h_0\left(1-\zeta_0 z^{-1}\right)\left(1-\zeta_1 z^{-1}\right)\left(1-\zeta_1^* z^{-1}\right)$$
$$= h_0\left(1-\zeta_0 z^{-1}\right)\left(1-2\,\mathrm{Re}\{\zeta_1\} z^{-1} + |\zeta_1|^2 z^{-2}\right)$$



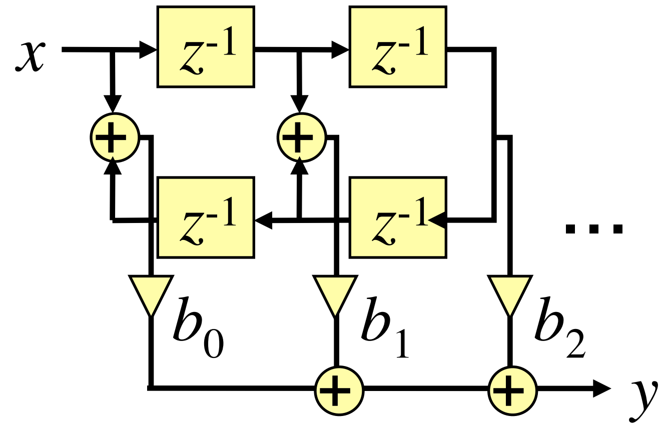
FIR Filter Structures

- Linear Phase:



Symmetric filters with $h[n] = (-)h[N - n]$

$$y[n] = b_0\left(x[n] + x[n-4]\right) + b_1\left(x[n-1] + x[n-3]\right) + b_2 x[n-2]$$



half as many multiplies

- Also Transpose form:
  gains first, feeding folded delay/sum line

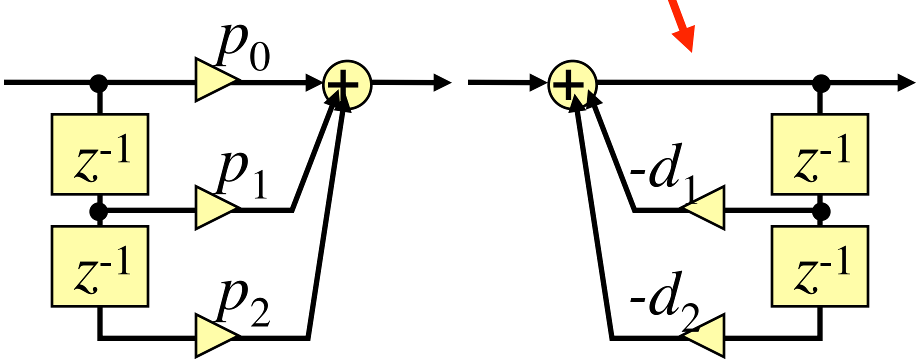
IIR Filter Structures

- IIR: numerator + denominator

$$H(z) = \frac{p_0 + p_1 z^{-1} + p_2 z^{-2} + \ldots}{1 + d_1 z^{-1} + d_2 z^{-2} + \ldots}$$

$$= P(z) \cdot \frac{1}{D(z)}$$

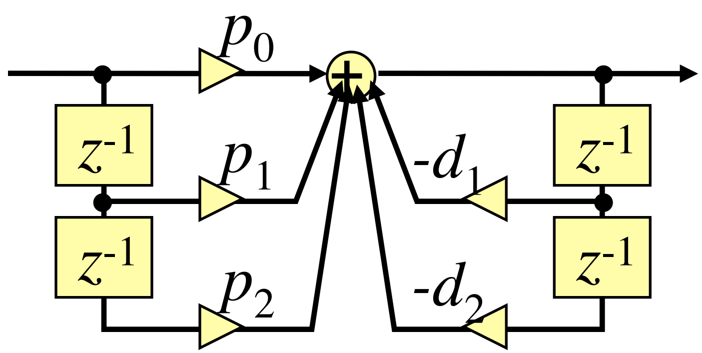
FIR



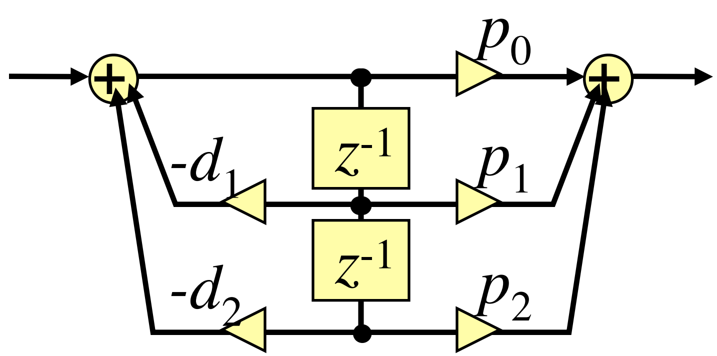
all-pole IIR

IIR Filter Structures

- Hence, Direct form I



- Commutation → Direct form II (DF2)

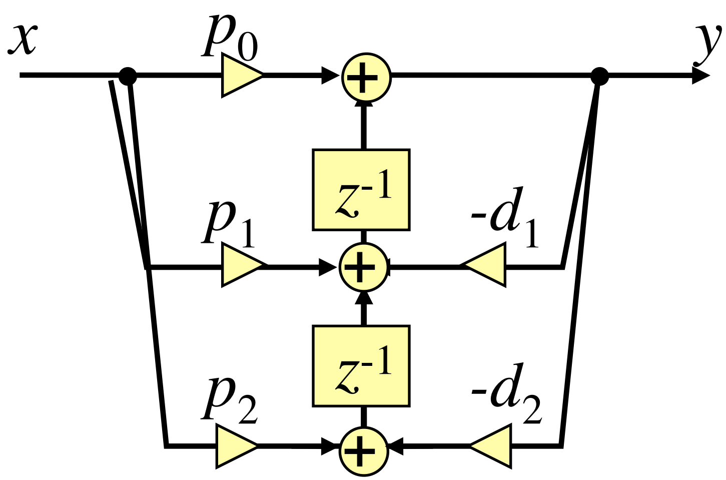


• *same signal*
∴ *delay lines merge*

• *"canonical"*
= min. memory usage

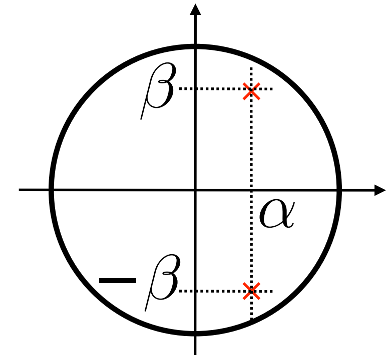
IIR Filter Structures

- Use Transpose on FIR/IIR/DF2



- "Direct Form II Transpose"

Factored IIR Structures



- Real-output filters have conjugate-symm roots:

$$H(z)=\frac{1}{\left(1-(\alpha+j\beta)z^{-1}\right)\left(1-(\alpha-j\beta)z^{-1}\right)}$$

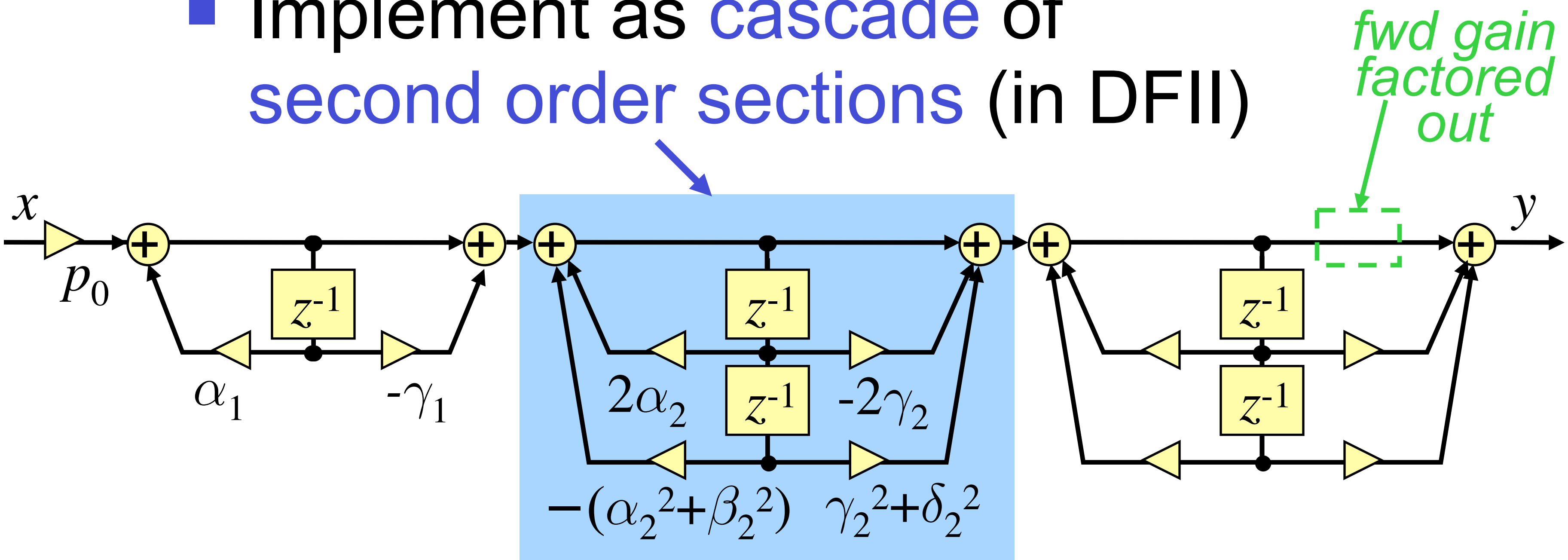
- Can always group into 2nd order terms with real coefficients:

$$H(z)=\frac{p_0\left(1-\gamma_1 z^{-1}\right)\left(1-2\gamma_2 z^{-1}+(\gamma_2^2+\delta_2^2)z^{-2}\right)\ldots}{\left(1-\alpha_1 z^{-1}\right)\left(1-2\alpha_2 z^{-1}+(\alpha_2^2+\beta_2^2)z^{-2}\right)\ldots}$$

real root

Cascade IIR Structure

- Implement as cascade of second order sections (in DFII)



- Second order sections (SOS):
  - modular - any order from optimized block
  - well-behaved, real coefficients (sensitive?)

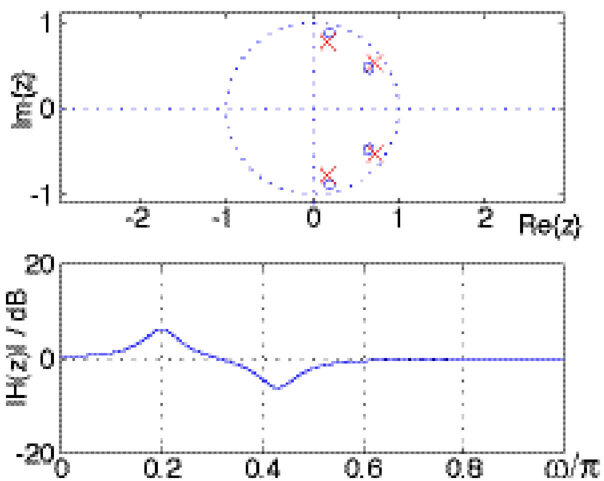
Second-Order Sections

- 'Free' choices:
  - grouping of pole pairs with zero pairs
  - order of sections
- Optimize numerical properties:
  - avoid very large values (overflow)
  - avoid very small values (quantization)
- e.g. Matlab's `zp2sos`
  - attempt to put 'close' roots in same section
  - intersperse gain & attenuation?

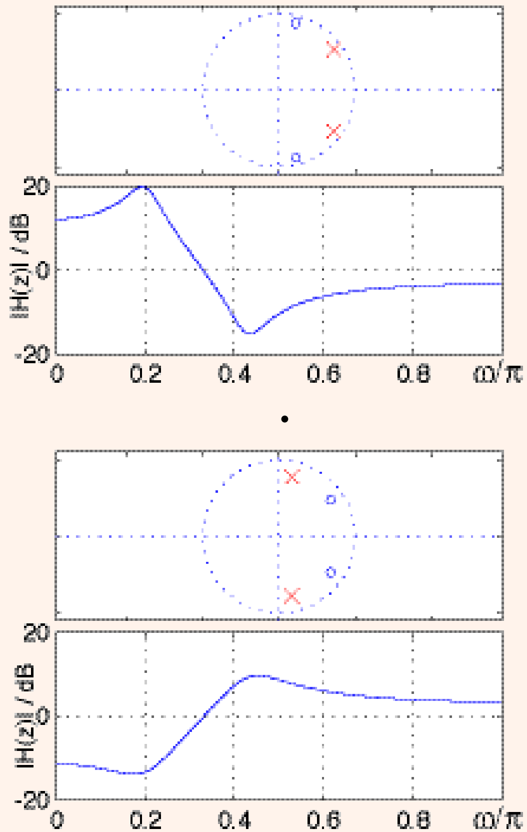
Second Order Sections

- Factorization affects intermediate values

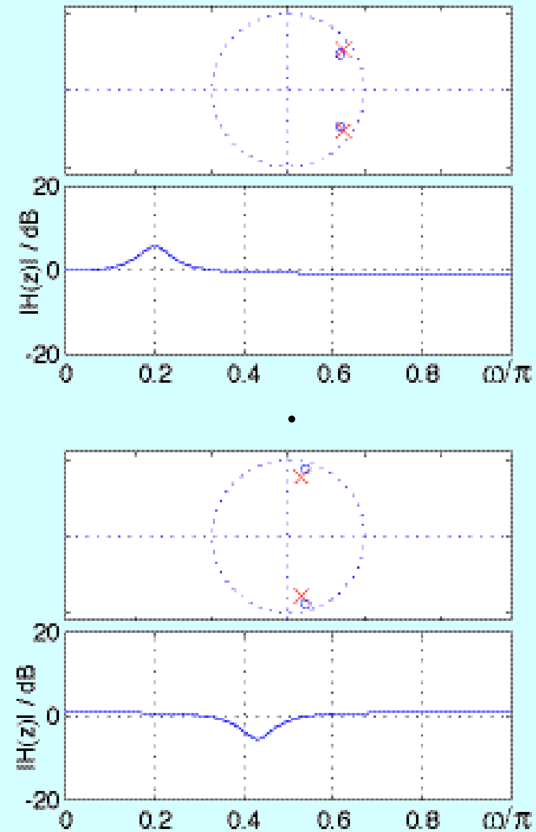
Original System
(2 pair poles, zeros)



Factorization 1



Factorization 2



Parallel IIR Structures

- Can express $H(z)$ as sum of terms (IZT)

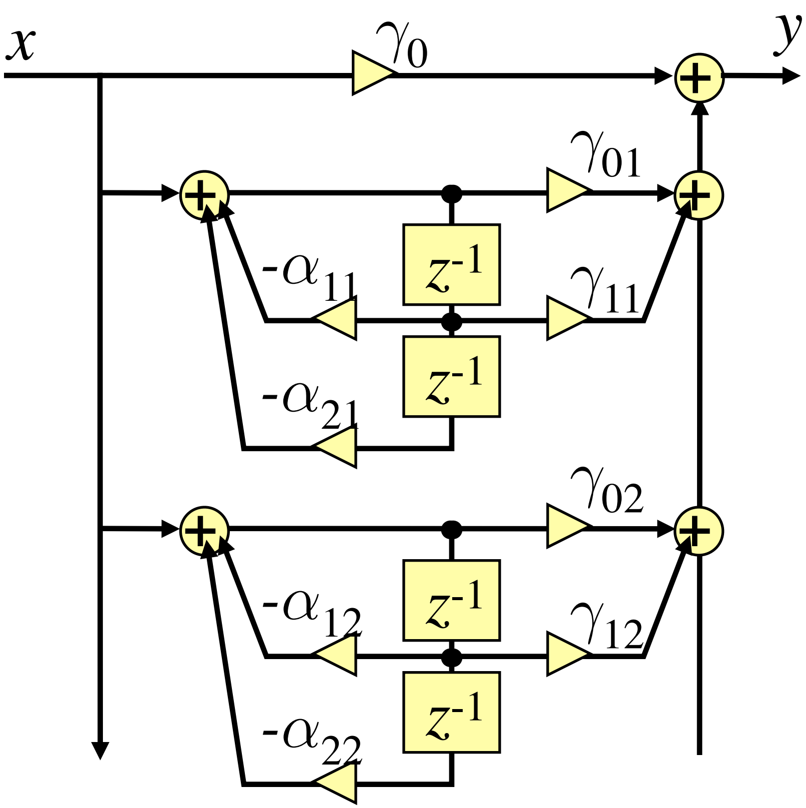
$$H(z) = \text{consts} + \sum_{\ell=1}^{N} \frac{\rho_\ell}{1 - \lambda_\ell z^{-1}} \qquad \rho_\ell = \left(1 - \lambda_\ell z^{-1}\right) F(z)\Big|_{z=\lambda_\ell}$$

- Or, second-order terms:

$$H(z) = \gamma_0 + \sum_k \frac{\gamma_{0k} + \gamma_{1k} z^{-1}}{1 + \alpha_{1k} z^{-1} + \alpha_{2k} z^{-2}}$$

- Suggests parallel realization...

Parallel IIR Structures



- Sum terms become parallel paths
- Poles of each SOS are from full TF
- System zeros arise from output sum
- Why do this?
  - stability/sensitivity
  - reuse common terms