INFINITE IMPULSE RESPONSE DIGITAL FILTERS: LECTURE 2

Paolo Favaro

Chapter 5 of Textbook

(original material from John Thompson)

Contents

- Revise Bilinear design example
- Canonical form of IIR Filters
- High pass, bandpass, bandstop filters
- Implementation issues

Bilinear Design Example

Problem: Design a 2nd order Butterworth filter with cutoff ω_c=628 rad/s, sampling freq ω_s=5024 rad/s

- The normalised Butterworth filter is:

$$H(s) = \frac{1}{1 + \sqrt{2}s + s^2}$$

- Calculate pre-warping frequency:

$$\omega_a = \left(\frac{2}{1/800}\right)\tan\left(\frac{2\pi 100}{2\times 800}\right) = 663 \text{ rad/s}$$

- De-normalise using : $s \mapsto s/\omega_a = s/663$

$$H(s) = \frac{1}{1 + (\sqrt{2}s/663) + (s/663)^2}$$

- Apply bilinear transform:

$$s = \frac{2(1 - z^{-1})}{\Delta t(1 + z^{-1})}$$

- Applying bilinear transformation gives:

$$H(z) = \frac{1}{\left(\dfrac{2\times 800(1-z^{-1})}{663(1+z^{-1})}\right)^2 + \sqrt{2}\left(\dfrac{2\times 800(1-z^{-1})}{663(1+z^{-1})}\right) + 1}$$

- Algebraic simplification leads to:

$$H(z) = \frac{0.098 + 0.195z^{-1} + 0.098z^{-2}}{1 - 0.942z^{-1} + 0.333z^{-2}}$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{0.098 + 0.195z^{-1} + 0.098z^{-2}}{1 - 0.942z^{-1} + 0.333z^{-2}}$$

- Multiplying out:

$$Y(z)(1 - 0.942z^{-1} + 0.333z^{-2}) = X(z)(0.098 + 0.195z^{-1} + 0.098z^{-2})$$

- Finally can apply inverse *z*-transform to yield the *difference equation*:

$$y(n) = 0.098x(n) + 0.195x(n-1) + 0.098x(n-2)$$
$$+ 0.942y(n-1) - 0.333y(n-2)$$

Filter Frequency Response

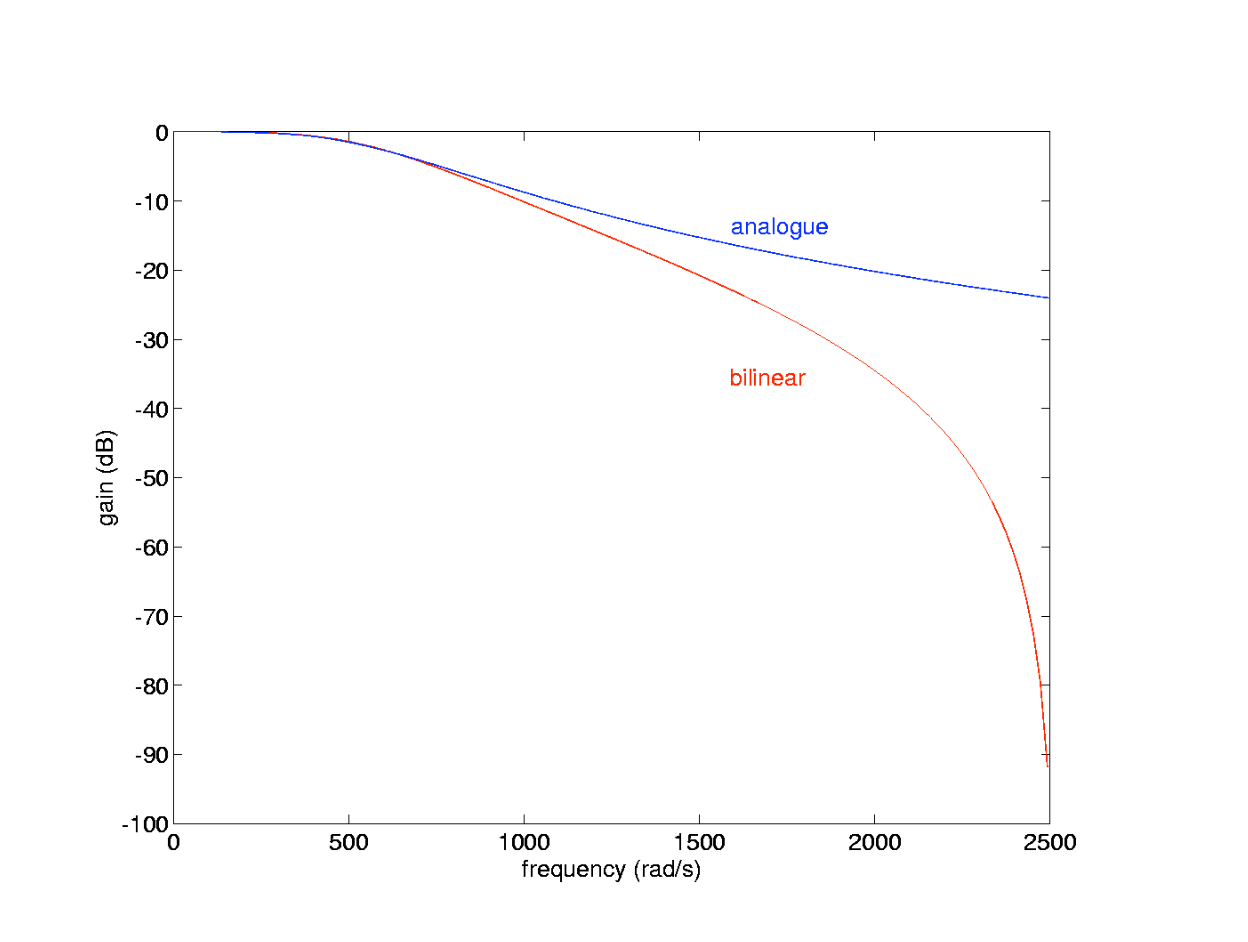


Fig5.12

Filter Phase Response

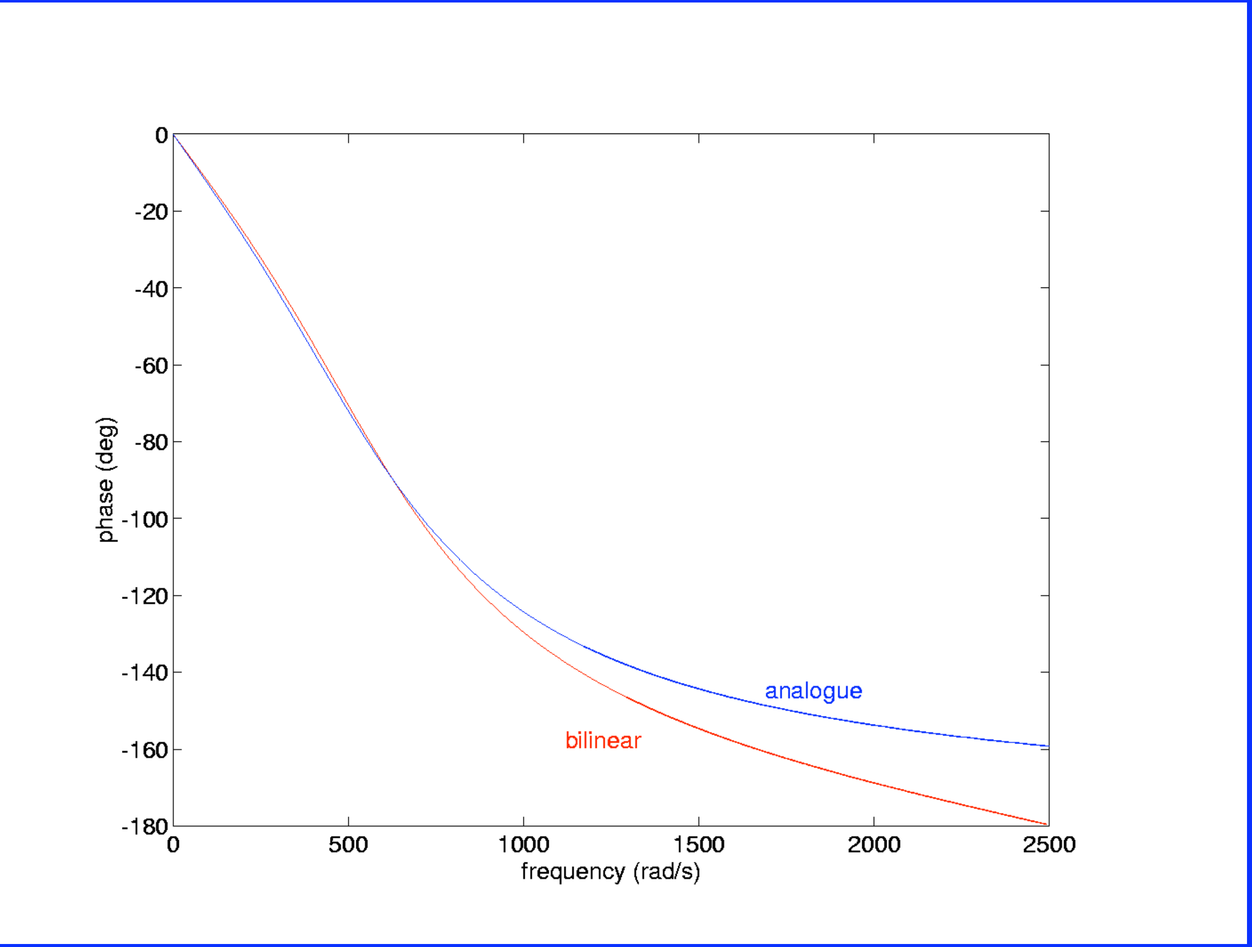
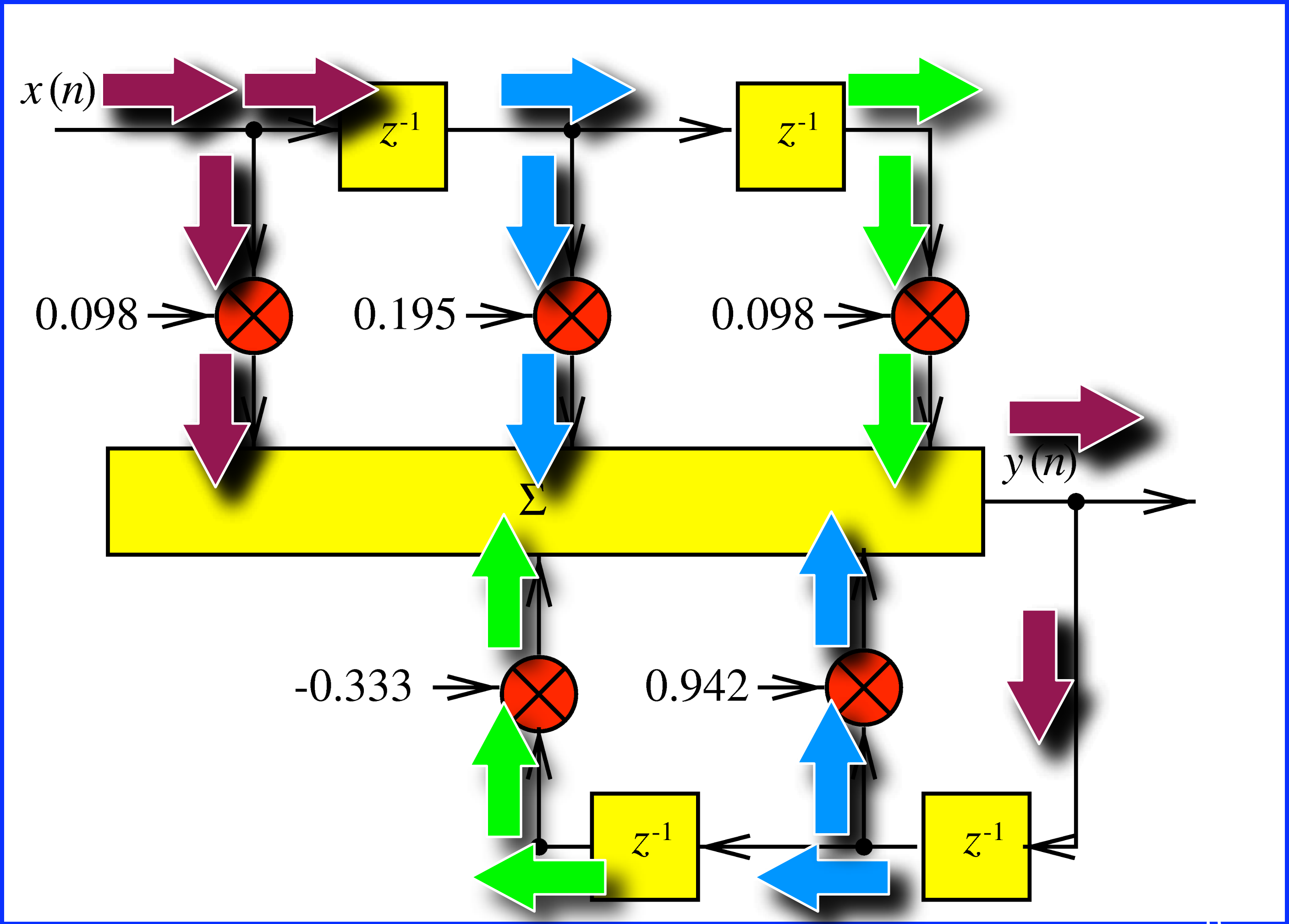


Fig5.13

IIR Filter Structure



IIR Filtering

- In example, 2nd order IIR filter design had 3 feedforward/2 feedback taps
- In general, *N*-th order IIR filter design has *N+1* feedforward/*N* feedback taps
- Must trade off:
  - →Roll off rate of filter design
  - →N° of taps in filter implementation

Direct to Canonical Form

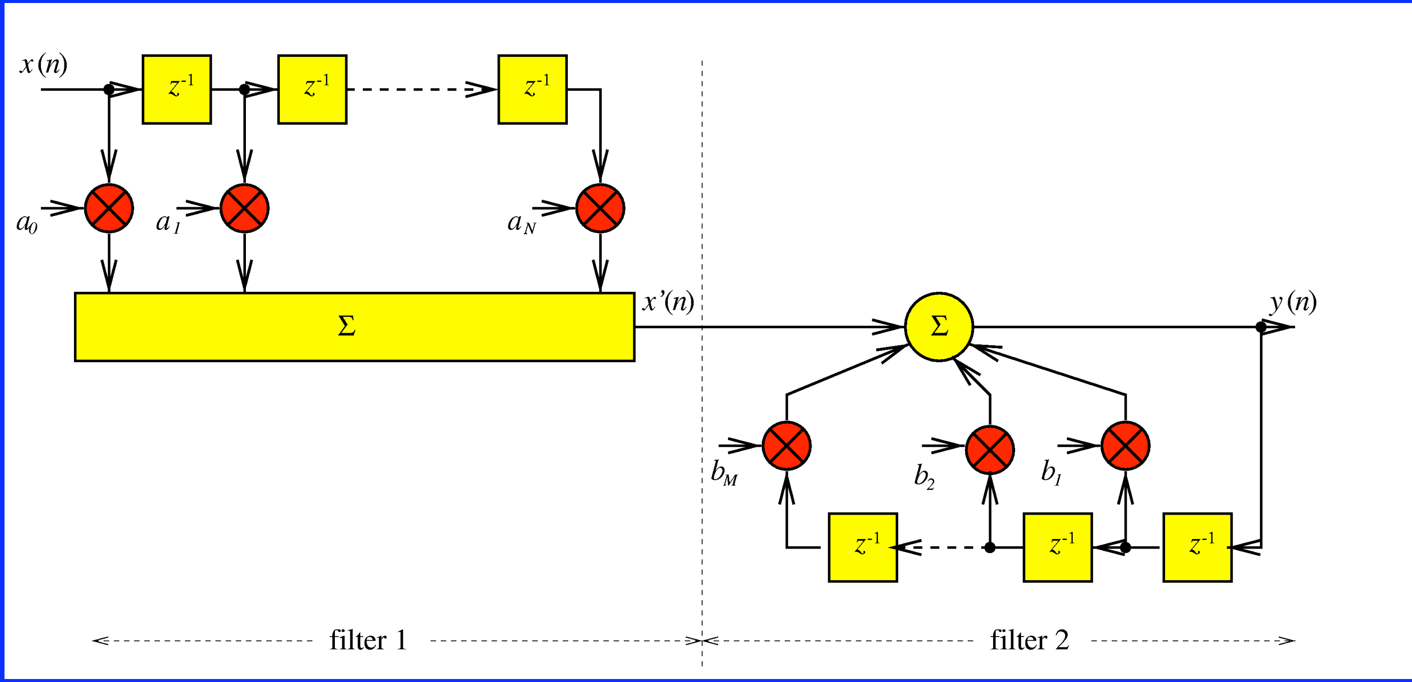


Fig5.5a

$$H(z) = Y(z) / X(z) = \sum_{i=0}^{N} a_i z^{-i} \Big/ 1 - \sum_{i=1}^{M} b_i z^{-i}$$

- Redefine filter 1: $Y'(z) = X(z) \Big/ 1 - \sum_{i=1}^{M} b_i z^{-i}$

- So filter 2 is now: $Y(z) = \sum_{i=0}^{N} a_i z^{-i} Y'(z)$

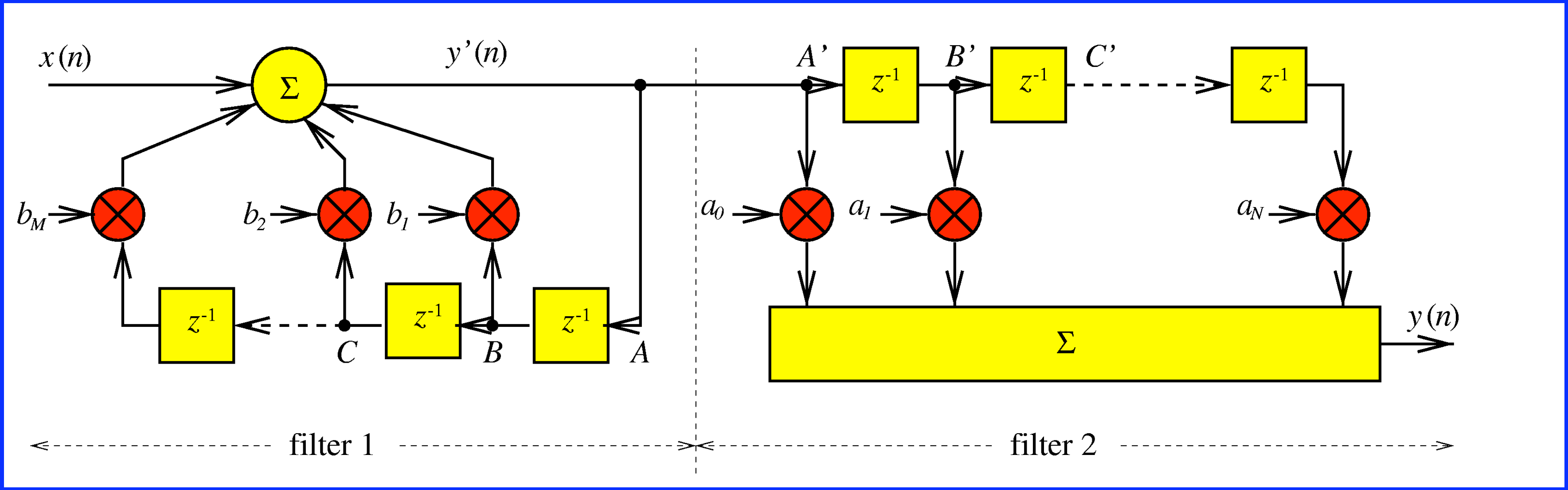


Fig5.5b

The Canonical Form

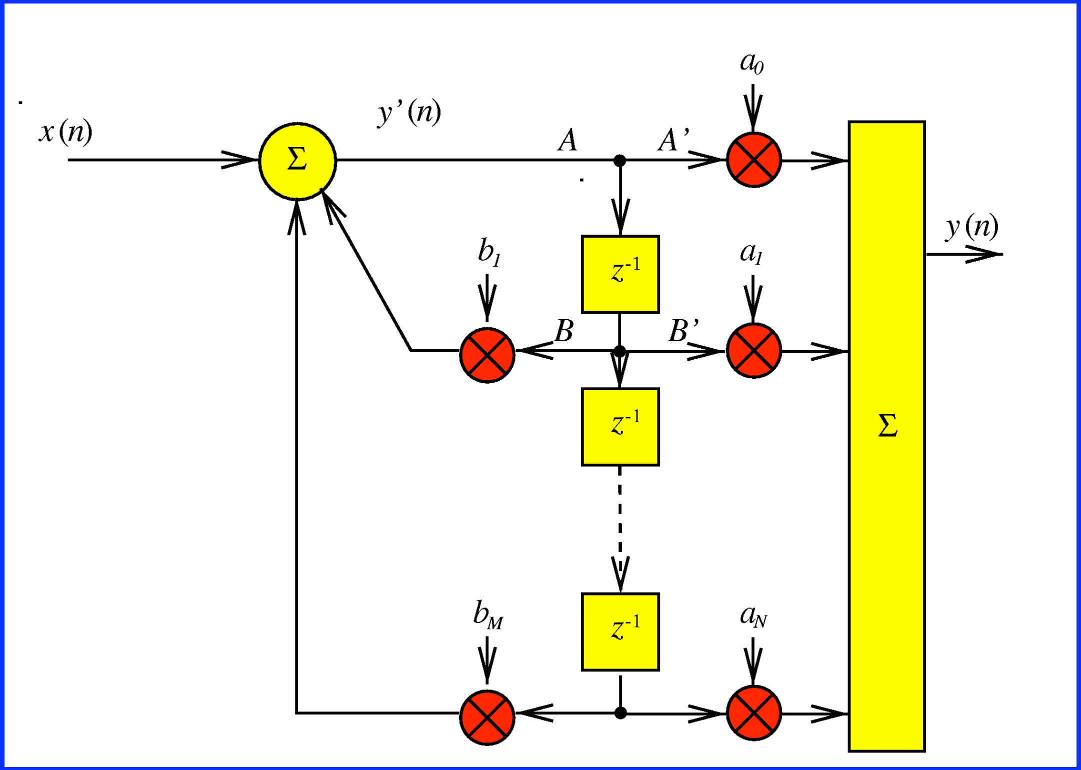


Fig5.5c

- Minimum Storage Filter Implementation

Worked Example:Canonical Form

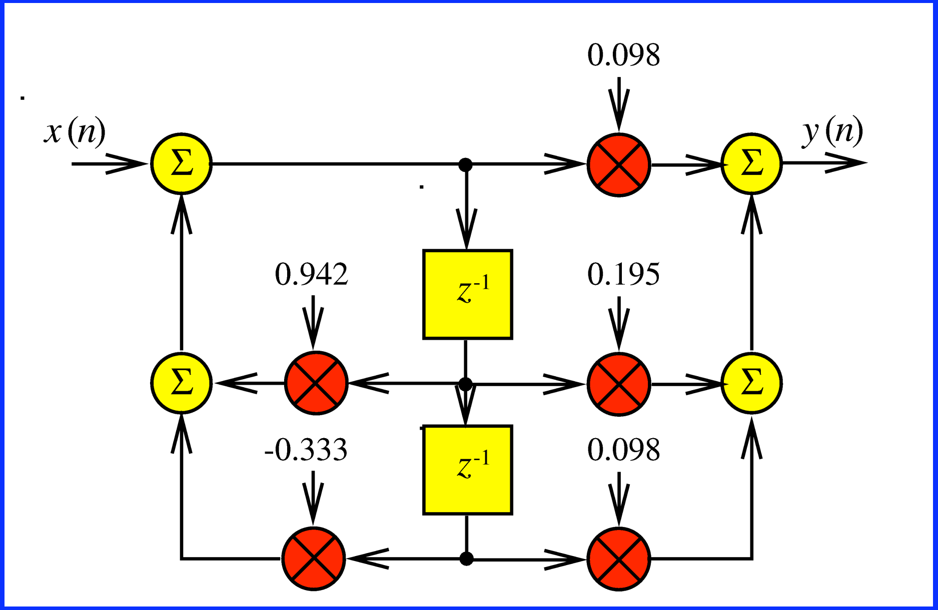
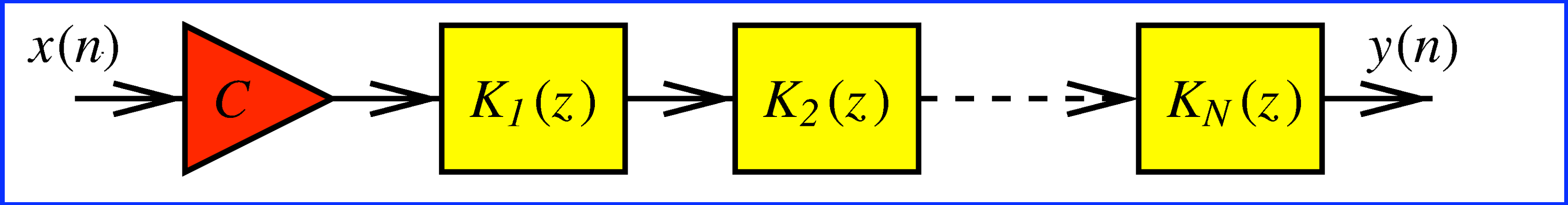


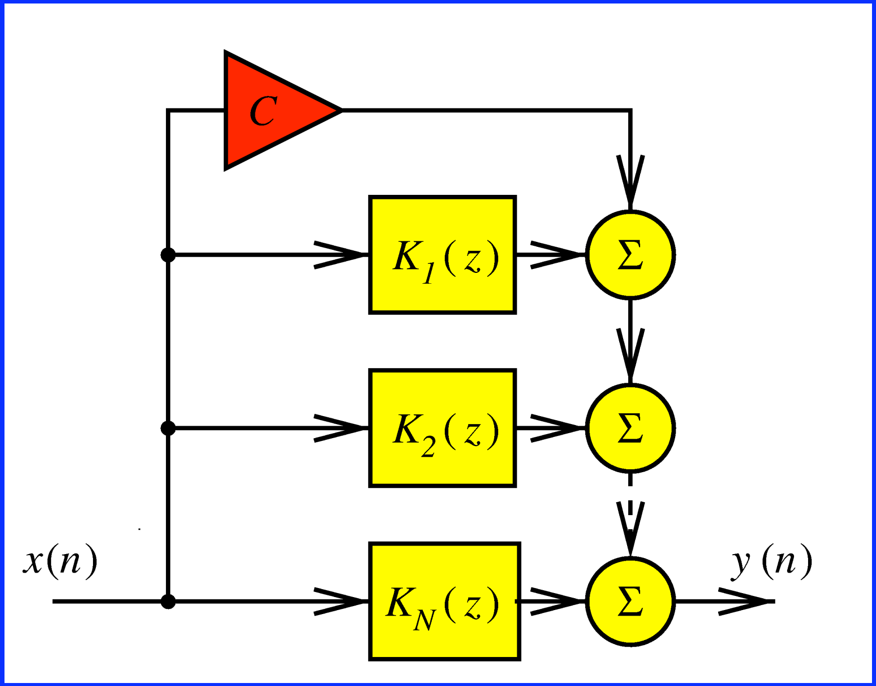
Fig5.11

$$y(n) = 0.098y'(n) + 0.195y'(n-1) + 0.098y'(n-2)$$

$$y'(n) = x(n) + 0.942y'(n-1) - 0.333y'(n-2)$$

Serial and Parallel Cascades





Figs 5.6 & 5.7

Serial or Parallel?

- Build up filter from 1st/2nd order blocks
- Parallel cascades are less sensitive to filter coefficient errors
- Serial cascade usually preferred:
  - →2nd order blocks have integer coefficients
  - →Many filter design packages assume this configuration

High Pass Filters

- To design a LP filter, we used the analogue substitution $s \rightarrow s/\omega_a$
- Given a digital LP filter with cutoff ω_{cLP}, design HP filter with cutoff $(\omega_{cs}/2 - \omega_{cLP})$:
  - $\rightarrow$Swap signs of poles and zeros
  - $\rightarrow$So $H_{HP}(z) = H_{LP}(-z)$

$$s_L = \frac{2}{\Delta t}\frac{1 - z^{-1}}{1 + z^{-1}} \qquad s_H = \frac{2}{\Delta t}\frac{1 + z^{-1}}{1 - z^{-1}} = \frac{4}{\Delta t^2}\frac{1}{s_L}$$

Bandpass and Bandstop Filters

- Can create a bandpass filter from a lowpass digital filter prototype:
  - →Low pass cutoff = $\omega_{cu}-\omega_{cl}$
  - →Substitution for H(z) is:

$$z^{-1} \rightarrow \frac{-z^{-1}(z^{-1}-\alpha)}{1-\alpha z^{-1}} \qquad \alpha = \frac{\cos(\pi(\omega_{cu}+\omega_{cl})/\omega_s)}{\cos(\pi(\omega_{cu}-\omega_{cl})/\omega_s)}$$

$$s_B = \frac{1}{4s_L}(\alpha_0^* + \beta_0^* s_L)(\alpha_1^* + \beta_1^* s_L)$$

- Transform doubles number of poles/ zeros and hence the filter order
- In an identical way, bandstop filters can be created as follows:

$$z^{-1} \rightarrow \frac{+z^{-1}(z^{-1} - \alpha)}{1 - \alpha z^{-1}} \qquad \alpha = \frac{\cos(\pi(\omega_{cu} + \omega_{cl})/\omega_s)}{\cos(\pi(\omega_{cu} - \omega_{cl})/\omega_s)}$$

- Note the change in the sign!

Bandpass/Highpass Example

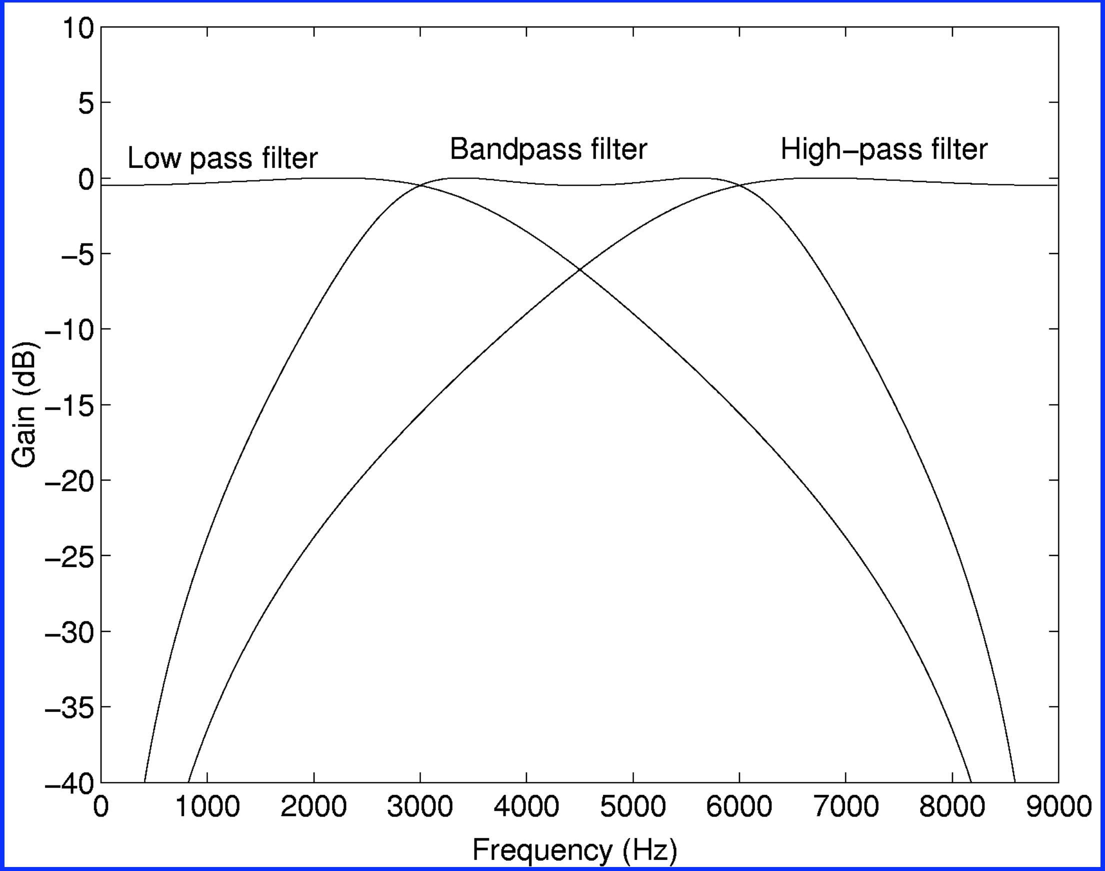


Fig 5.16: Low pass prototype with the bandpass & high-pass designs

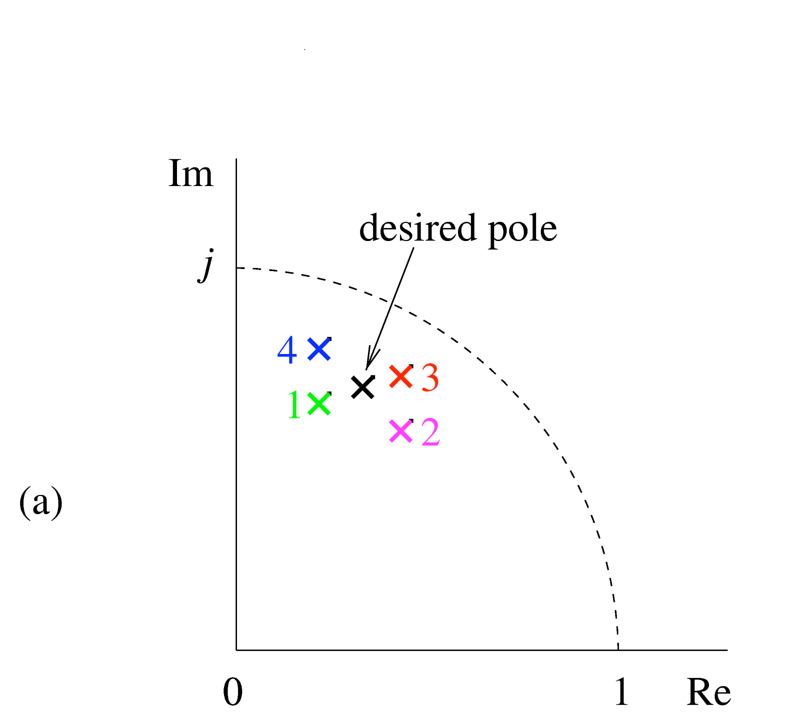
Coefficient Quantisation Errors

- Quantising filter coefficients will move locations of poles and zeros

  → This will change the frequency response of the filter

  → Must ensure all poles stay INSIDE unit circle to ensure stability

Effect of Pole Quantisation



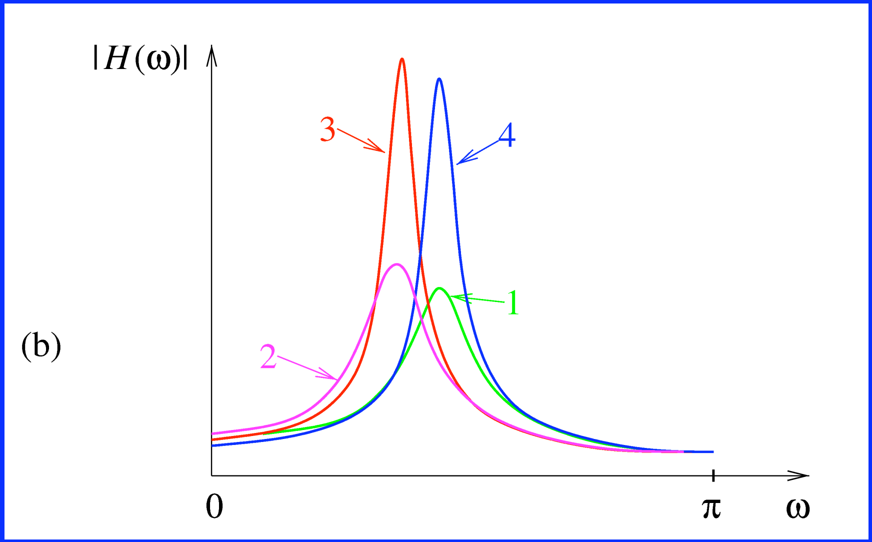


Fig5.17: (a) z-plane, (b) Frequency response

Stability Problems

- Poles near unit circle can move outside causing instability
- Consider:

$$H(z) = \frac{1}{(1-0.901z^{-1})(1-0.943z^{-1})}$$

→Poles at 0.901 and 0.943

→Quantising to nearest 0.05, poles at 0.9 and 0.95

- Multiplying out the denominator of *H(z)* gives:

$$H(z) = \frac{1}{1 - 1.844z^{-1} + 0.8496z^{-2}}$$

- Quantise *H(z)* coeffs to nearest 0.05:
  - →Obtain poles at 0.85 and 1
  - →Hence the filter is now unstable!
  - →Problem gets worse for high order filters

Limit Cycles

- Limit cycles give an oscillatory output with zero input
- Consider the first order filter:

$$y(n) = -0.9\,y(n-1) + x(n)$$

- With infinite precision, impulse response decays to zero:

$$y(n) = 1, -0.9, 0.8, -0.7, 0.6, -0.5, 0.45 \ldots$$

- Now assume rounding to nearest 0.1
  - →The value 0.45 rounds up to 0.5
  - →The output cycles between ±0.5

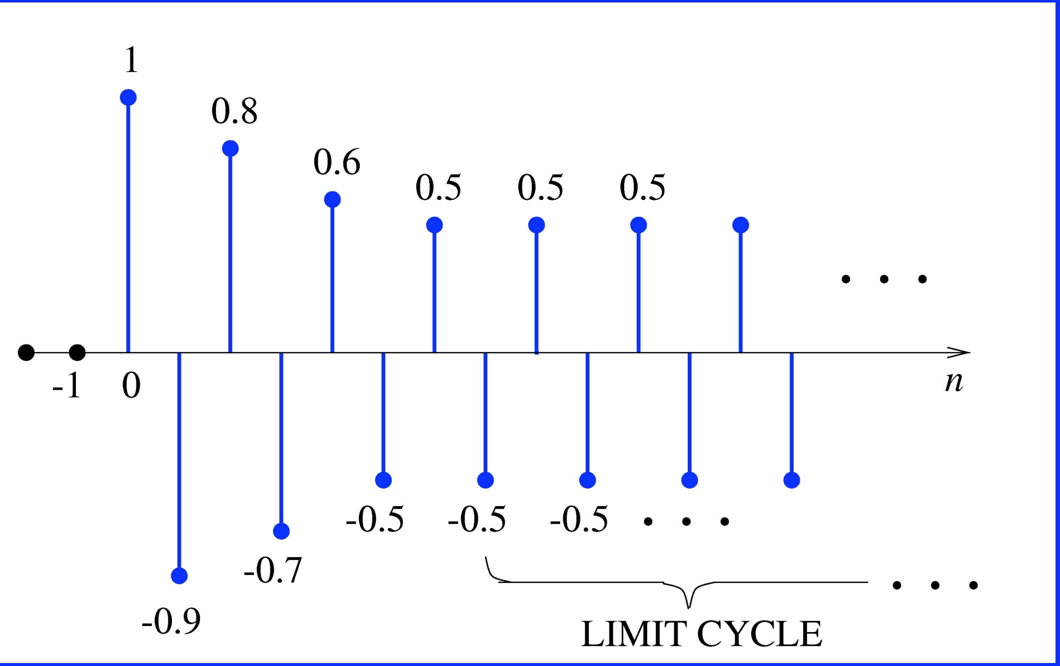


Fig 5.18

- Minimise effect with increased precision

Minimising Finite Precision Effects

- A key result for cascade designs:

> *If each section is stable (poles inside unit circle) and free from limit cycles, then the cascaded higher order design retains these properties*

- A major motivation for cascade filters built up from1st/2nd order sections

How Many Bits?

- Formula for fixed precision processors:

$$b_c = \log_2(1/[\delta_1 - \delta_{10}]) + \log_2(\omega_s /[\Delta\omega\, 2\sin(2\pi\omega_{cl} / \omega_{sl})])$$

- Where:

  δ_1, δ_{10} = passband ripple before/after quantisation

  ω_{cl} , ω_{sl} , ω_s = cutoff, bandstop and sampling frequency

  $\Delta\omega = \omega_{sl} - \omega_{cl}$

IIR Filter Implementation

- Four key steps:
  - →Define storage of inputs, outputs & filter taps
  - →Function to get current input
  - →Evaluate filter and return the output
  - →Apply unit delay to input and output memories

Summary of Chapter 5

- Prototype analogue filters
- Bilinear transform filter design
- Direct vs canonical form for IIR filter
- Implementation issues