

# 18-551: DIGITAL COMMUNICATIONS AND SIGNAL PROCESSING SYSTEMS DESIGN

## Spring 2001

---

### Z-Transforms (S & S, Chap 10)

Equation numbers are from Chap 10 in S and S. See also: *DSP using MATLAB* (Chap 4), *Discrete Time Signal Processing* by A. Oppenheim and R. Schaffer (O and S) (Chap 4).

From Chap 3 (S & S page 183), output  $y[n]$  from system with impulse response  $h[n]$  for a complex exponential input  $z^n$  is  $y[n] = H(z)z^n$  where  $H(z) = X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}$  (1)

is the  $z$ -transform (ZT) of a sequence  $x[n]$ ,  $z$  is complex variable

$z$ -transform pairs exist  $x[n] \leftrightarrow X(z)$  (2)

also define a unilateral (not bilateral)  $z$ -transform  $X(z) = \sum_{n=0}^{\infty} x[n]z^{-n}$

Relation to DTFT: If  $z = e^{j\omega}$  (i.e.  $|z| = 1$ ), then  $X(z) = X(e^{j\omega}) = \text{DTFT of } x[n]$ .

\*In 551, we **use** the  $z$ -transform, we don't calculate  $z$ -transform pairs (they already known - Tables) or analyze ROC.

### Overview

For a system with a given or desired  $H(z)$   $z$ -transform, we can

- Find a difference equation that describes the system. Use inverse  $z$ -transform of  $H(z) = Y(z)/X(z)$ .
- Describe a system (structure) to implement difference equation. Use flowgraphs, design filter structures.
- Describe transfer function of system (set  $z = e^{j\omega}$ )  $H(e^{j\omega})$ .
- Describe impulse response of system (use  $ZT^{-1}$ , inverse ZT)  $h[n]$ .
- Determine poles and zeroes and magnitude and phase response of a given system (or filter). Use Matlab.
- Find the solution to a difference equation.
- Our approach will be to use Matlab to design filters (i.e. calculate coefficients) and use ZTs to implement/select the structure of the filters (IIR,FIR), then DSP HW implementation [Great project area]. Then we'll discuss adaptive filters (coefficients change).

**Z-Transform (ZT) Pairs (and insight, their interpretation) see 3-3 and 3-4**

- \*(10.5.1) linearity - this vital, break input into terms you know ZT of

$$5. \quad a^n u[n] \leftrightarrow \frac{1}{1 - az^{-1}} = \frac{z}{z - a} \quad \text{positive time sequence}$$

$$7. \quad na^n u[n] \leftrightarrow \frac{az^{-1}}{(1 - az^{-1})^2}, \quad \text{ZT}^{-1} \text{ terms of form on right hand side}$$

- \*(4),(10.5.2)  $z^{-1}$  corresponds to a unit delay of 1 in  $x[n]$  input

$z^{-n_0}$  corresponds to a delay of  $n_0$  in  $x[n - n_0]$  input

This pair is vital for drawing filter (system) structures with delay lines etc.

- (10.5.7 B) First difference  $x[n] - x[n - 1] \leftrightarrow (1 - z^{-1})X(z)$

This follows from above, obvious use in difference equations.

- (10.5.7 C) Accumulation, useful to describe ZT of a sum of samples.

- \*(10.5.7 A) Convolution  $x_1[n] * x_2[n] \leftrightarrow X_1(z)X_2(z)$

e.g. If  $X_1(z) = 2 + 3z^{-1} + 4z^{-2}$  and  $X_2(z) = 3 + 4z^{-1} + 5z^{-2} + 6z^{-3}$ ,

what is  $X_3(z) = H_1(z)H_2(z)$ ?

Ans: From ZT definition  $H(z) = X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}$

$x_1[n] = \{2, 3, 4\}, x_2[n] = \{3, 4, 5, 6\}$  are 2  $x$  sequences



$X_3(z)$  coefficients found from convolution of 2 sequences

Flip  $x_2$ , \* is  $3 \times 2$ , shift right  $3 \times 3 + 4 \times 2 = 17$ , etc.

or 6 17 34 43 38 24 (use *conv* in Matlab)

$$X_3(z) = 6 + 17z^{-1} + 34z^{-2} + 43z^{-3} + 38z^{-4} + 24z^{-5}$$

**TABLE 10.1 PROPERTIES OF THE z-TRANSFORM**

Section	Property	Signal	z-Transform	ROC
		$x[n]$ $x_1[n]$ $x_2[n]$	$X(z)$ $X_1(z)$ $X_2(z)$	$R$ $R_1$ $R_2$
10.5.1	Linearity	$ax_1[n] + bx_2[n]$	$aX_1(z) + bX_2(z)$	At least the intersection of $R_1$ and $R_2$
10.5.2	Time shifting	$x[n - n_0]$	$z^{-n_0}X(z)$	$R$ , except for the possible addition or deletion of the origin
10.5.3	Scaling in the z-domain	$e^{jn_0}x[n]$ $z_0^n x[n]$ $a^n x[n]$	$X(e^{-jn_0}z)$ $X\left(\frac{z}{z_0}\right)$ $X(a^{-1}z)$	$R$ $z_0R$ Scaled version of $R$ (i.e., $ a R$ = the set of points $\{ a z\}$ for $z$ in $R$ )
10.5.4	Time reversal	$x[-n]$	$X(z^{-1})$	Inverted $R$ (i.e., $R^{-1}$ = the set of points $z^{-1}$ , where $z$ is in $R$ )
10.5.5	Time expansion	$x_{(rk)}[n] = \begin{cases} x[n/r], & n = rk \\ 0, & n \neq rk \end{cases}$	$X(z^k)$	$R^{1/k}$ (i.e., the set of points $z^{1/k}$ , where $z$ is in $R$ )
10.5.6	Conjugation	$x^*[n]$	$X^*(z^*)$	$R$
10.5.7	Convolution	$x_1[n] * x_2[n]$	$X_1(z)X_2(z)$	At least the intersection of $R_1$ and $R_2$
10.5.7	First difference	$x[n] - x[n - 1]$	$(1 - z^{-1})X(z)$	At least the intersection of $R$ and $ z  > 0$
10.5.7	Accumulation	$\sum_{k=-\infty}^n x[k]$	$\frac{1}{1 - z^{-1}}X(z)$	At least the intersection of $R$ and $ z  > 1$
10.5.8	Differentiation in the z-domain	$nx[n]$	$-z \frac{dX(z)}{dz}$	$R$
10.5.9	Initial Value Theorem			
		If $x[n] = 0$ for $n < 0$ , then		
		$x[0] = \lim_{z \rightarrow \infty} zX(z)$		

**TABLE 10.2** SOME COMMON  $z$ -TRANSFORM PAIRS

Signal	Transform	ROC
1. $\delta[n]$	1	All $z$
2. $u[n]$	$\frac{1}{1 - z^{-1}}$	$ z  > 1$
3. $-u[-n - 1]$	$\frac{1}{1 - z^{-1}}$	$ z  < 1$
4. $\delta[n - m]$	$z^{-m}$	All $z$ , except 0 (if $m > 0$ ) or $\infty$ (if $m < 0$ )
5. $\alpha^n u[n]$	$\frac{1}{1 - \alpha z^{-1}}$	$ z  >  \alpha $
6. $-\alpha^n u[-n - 1]$	$\frac{1}{1 - \alpha z^{-1}}$	$ z  <  \alpha $
7. $n\alpha^n u[n]$	$\frac{\alpha z^{-1}}{(1 - \alpha z^{-1})^2}$	$ z  >  \alpha $
8. $-n\alpha^n u[-n - 1]$	$\frac{\alpha z^{-1}}{(1 - \alpha z^{-1})^2}$	$ z  <  \alpha $
9. $[\cos \omega_0 n]u[n]$	$\frac{1 - [\cos \omega_0]z^{-1}}{1 - [2 \cos \omega_0]z^{-1} + z^{-2}}$	$ z  > 1$
10. $[\sin \omega_0 n]u[n]$	$\frac{[\sin \omega_0]z^{-1}}{1 - [2 \cos \omega_0]z^{-1} + z^{-2}}$	$ z  > 1$
11. $[r^n \cos \omega_0 n]u[n]$	$\frac{1 - [r \cos \omega_0]z^{-1}}{1 - [2r \cos \omega_0]z^{-1} + r^2 z^{-2}}$	$ z  > r$
12. $[r^n \sin \omega_0 n]u[n]$	$\frac{[r \sin \omega_0]z^{-1}}{1 - [2r \cos \omega_0]z^{-1} + r^2 z^{-2}}$	$ z  > r$

## How to Calculate The Inverse Z-transform (ZT<sup>-1</sup>)

Motivation - Filters will be described by an  $H(z)$  that is the quotient of polynomials in  $z^{-1}$ , e.g.

$$H(z) = \frac{b_0 + b_1 z^{-1} + \dots + b_M z^{-M}}{1 + a_1 z^{-1} + \dots + a_N z^{-N}} \quad \text{with } N \geq M \quad (3)$$

To implement such a filter (determine its structure/architecture/flowgraph), we need to express  $H(z)$  as the sum of terms whose ZT<sup>-1</sup> we know. Obvious solution method is *partial fraction expansion* (PFE).

For causal systems, to get (3), rewrite  $H(z)$  as polynomials in  $z$  (not  $z^{-1}$ ) (multiply top and bottom by  $z^N$  or  $z^M$ ). If the order of the new numerator polynomial is  $\leq$  order of new denominator polynomial, then system is causal (and  $H(z)$  is rational). Thus, **you need  $N \geq M$  in (3), then  $H(z)$  is rational function of  $z^{-1}$  and you can do PFE.**

Aside: if  $H(z)$  is not in  $z^{-1}$  form in (3) with 1 in denominator, divide through by  $z$  to a power etc to get the form in (3).

**PFE of (3)** yields

$$H(z) = \sum_{k=1}^N \frac{R_k}{1 - p_k z^{-1}} \quad (4)$$

where  $p_k$  is  $k^{\text{th}}$  pole of  $H(z)$  and  $R_k$  is the residue of  $H(z)$  at  $z = p_k$  i.e.

$$R_k = H(z)(1 - p_k z^{-1}) \Big|_{z=p_k} \quad (5)$$

Now  $H(z)$  is the sum of terms of the form  $\frac{a}{1 - b z^{-1}}$  (6)

Thus, you can find  $h[n]$ , poles, etc.  **$N$  is the order of the filter.**

### How To Do PFE?

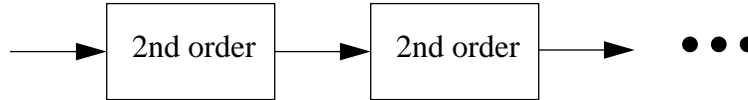
Matlab function `residuez` `residuez(b, a) = [R, p, c]`

where  $R, p$  etc are vectors and  $c$  (remainder coeffs of  $z^{-k}$ ) will be []. See special rule for multiple

poles (you get repeated  $p$  values and expansion terms are  $\frac{R_1}{1 - p_k z^{-1}} + \frac{R_2}{(1 - p_k z^{-1})^2}$  etc.)

## How To Describe An $N$ -th order $H(z)$ As The Product of A Number of 2nd Order $H_m(z)$ Functions?

Motivation - now  $N$ -th order  $H(z)$  can be represented as a cascade of 2nd order  $H_k^2(z)$ .



$$\begin{aligned}
 \text{i.e. } H(z) &= \frac{b_0 + b_1 z^{-1} + \dots + b_M z^{-M}}{1 + a_1 z^{-1} + \dots + a_N z^{-N}} \quad (7a) \text{ factor out } b_0 \text{ to get new } b' \\
 &= b_0 \frac{1 + b'_1 z^{-1} + \dots + b'_M z^{-M}}{1 + a_1 z^{-1} + \dots + a_N z^{-N}} \\
 &= b_0 \prod \frac{1 + b_{1,k} z^{-1} + b_{2,k} z^{-2}}{1 + a_{1,k} z^{-1} + a_{2,k} z^{-2}} = (7b) \text{ product of quotient of 2nd order polynomials}
 \end{aligned}$$

**Solution** - factor the numerator and denominator polynomials in (7a)

**How?**

(A) Just factor the numerator and denominator in (7a) into factors of the form  $(1 \pm a_m z^{-1})$ . This implementation yields a cascade of first order sections.

$$\text{(B) Factor (7a) into the form } H(z) = b_0 z^{N-M} \frac{\prod_{l=1}^M (z - z_l)}{\prod_{k=1}^N (z - z_k)} \quad (8)$$

Use Matlab function `roots` on numerator/denominator. This is useful as it gives poles and zeroes of  $H(z)$ . You can plot these with Matlab `zplane(b, a)` etc. Use on HW2.

(C) Factor (7a) into first-order factors with real and complex conjugate roots

$$H(z) = b_0 \frac{\prod_k (1 - c_k z^{-1}) \prod_k (1 - d_z z^{-1})(1 - d_z^* z^{-1})}{\prod_k (1 - e_k z^{-1}) \prod_k (1 - f_z z^{-1})(1 - f_z^* z^{-1})} \quad (9)$$

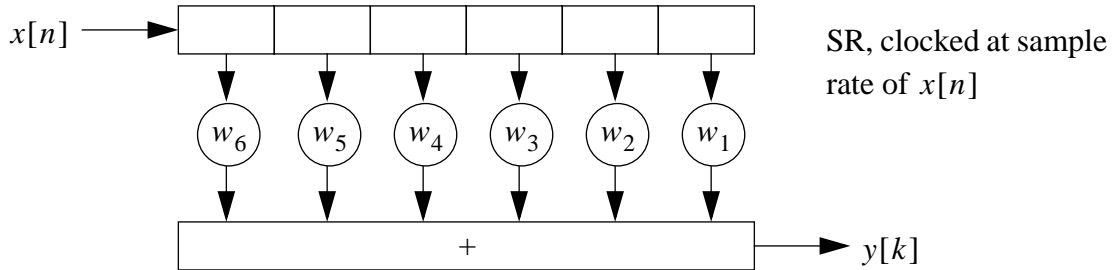
where  $c_k$  and  $e_k$  are real (zeroes and poles) and  $d_z$  and  $f_z$  are complex conjugate zero and pole pairs. Use `z = roots()` in Matlab, gives complex roots, you group into complex conjugate pairs. `cplxpairs(roots(norm a))` gives complex conjugate roots of the normalized  $a$  or  $b$

polynomial (normalized as  $b = \frac{b_1}{b_0}$ ).

**Now weights in (9) are real (you don't need complex multiplies)**

## Obtaining Difference Equations Describing A System With A Given $H(z)$ ZT (And Obtaining A Flowgraph From A Difference Equation)

**Background** - Tapped delay line (shift register SR, weights Taps  $W_N$ )



$$y[1] = x[1]w_6,$$

$$y[2] = x[1]w_5 + x[2]w_6$$

$$y[3] = x[1]w_4 + x[2]w_5 + x[3]w_6$$

etc

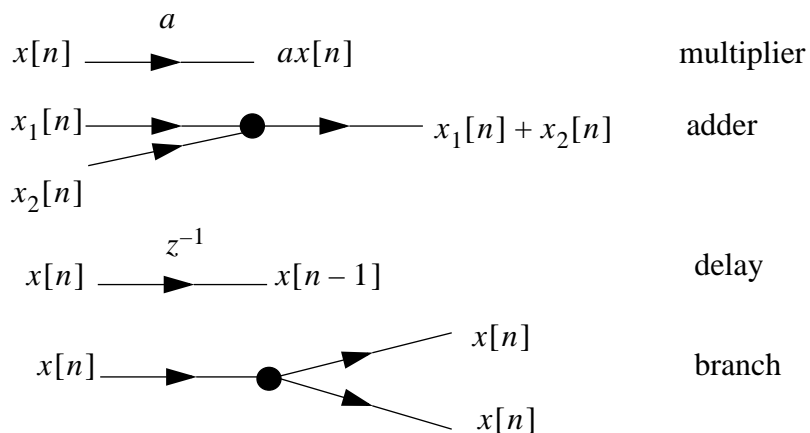
$$y[6] = x[1]w_1 + x[2]w_2 + x[3]w_3 + \dots + x[6]w_6$$

etc

$$y[k] = \sum w[n]x[n-k] = w \otimes x$$

Delays are SRs, weights are multipliers

### Flowgraphs



### Examples

Write difference equation (and draw flowgraph) for systems described by the system function  $H(z)$  given  $x[n] = \text{input}$ ,  $y[n] = \text{output}$ . Do for a number of examples.

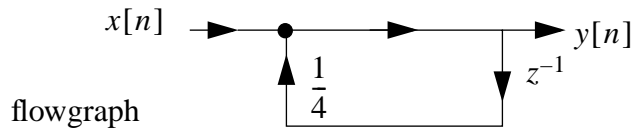
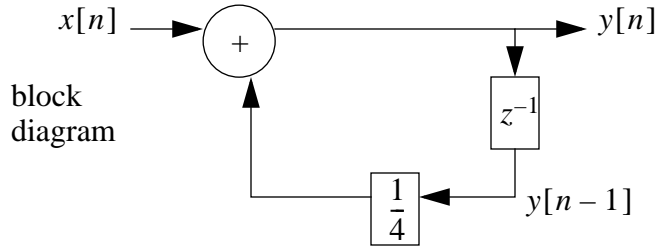
$$(1) H(z) = \frac{1}{1 - \frac{1}{4}z^{-1}}$$

$$\frac{Y(z)}{X(z)} = \frac{1}{1 - \frac{1}{4}z^{-1}}, \text{ cross multiply}$$

$$Y(z) - \frac{1}{4}Y(z)z^{-1} = X(z), \text{ do ZT}^{-1}$$

$$y[n] - \frac{1}{4}y[n-1] = x[n]$$

$$\text{or } y[n] = x[n] + \frac{1}{4}y[n-1]$$



**Fig A**

$$(2) H(z) = \frac{1}{\left(1 + \frac{1}{2}z^{-1}\right)\left(1 - \frac{1}{4}z^{-1}\right)}, \text{ mult out denom}$$

$$\frac{Y(z)}{X(z)} = \frac{1}{1 + \frac{1}{4}z^{-1} - \frac{1}{8}z^{-2}}, \text{ cross multiply}$$

$$Y(z) + \frac{1}{4}Y(z)z^{-1} - \frac{1}{8}Y(z)z^{-2} = X(z), \text{ do ZT}^{-1}$$

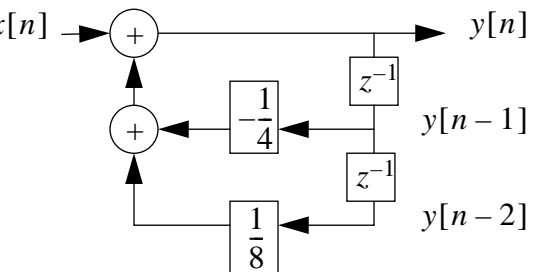
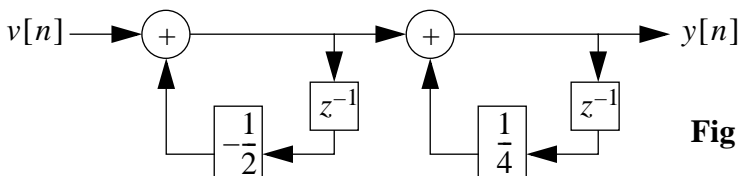
$$y[n] + \frac{1}{4}y[n-1] - \frac{1}{8}y[n-2] = x[n]$$

$$\text{or } y[n] = x[n] - \frac{1}{4}y[n-1] + \frac{1}{8}y[n-2]$$

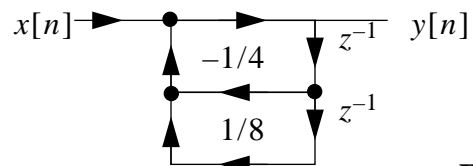
(2B) or using (2) as

$$H(z) = \left(\frac{1}{1 + \frac{1}{2}z^{-1}}\right)\left(\frac{1}{1 - \frac{1}{4}z^{-1}}\right) = H_1H_2$$

Implement  $H_1(z)$ , implement  $H_2(z)$ , hook two together



\*start drawing at  $y[n]$ , produce its delays, then feedback, produce  $y[n]$ , connect to form assumed  $y[n]$



**Fig B**

each portion is like Fig A

**Fig C**

Version in Fig. B is *direct form* (used multiplied out  $H(z)$  eqn)

Version in Fig. C is *cascade form* (used factored form of  $H(z)$  eqn)

(3)  $H(z) = \frac{1 - 2z^{-1}}{1 - \frac{1}{4}z^{-1}}$  use *cascade design*

$$\frac{Y(z)}{X(z)} = \left( \frac{1}{1 - \frac{1}{4}z^{-1}} \right) (1 - 2z^{-1}) = H_1(z)H_2(z) = \frac{Y_1(z)}{X(z)} \cdot \frac{Y(z)}{Y_1(z)}$$

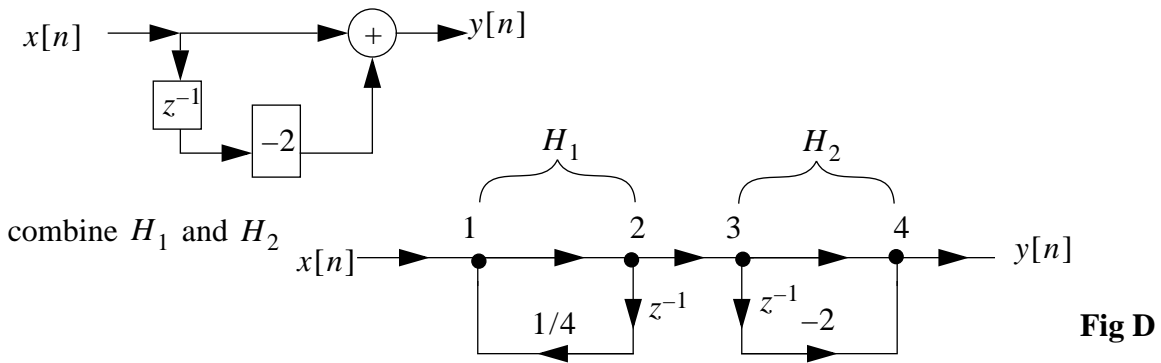
$H_1(z)$  looks like (1) and Fig A for  $H_1(z) = \frac{Y_1(z)}{X(z)}$ , output is  $Y_1(z)$ , see  $H_1$  in Fig. D below

$H_2(z) = \frac{Y(z)}{Y_1(z)}$ , input is  $Y_1(z)$  output from stage 1

$= 1 - 2z^{-1}$ , cross multiply

$Y_1(z) - 2Y_1(z)z^{-1} = Y(z)$ , do ZT<sup>-1</sup>

$y_1[n] - 2y_1[n-1] = y[n]$ , assume  $y_1[n]$ , form its delays, form LHS, label it  $y[n]$



If you implement (3) as  $H(z) = (1 - 2z^{-1}) \left( \frac{1}{1 - (1/4)z^{-1}} \right)$ , then

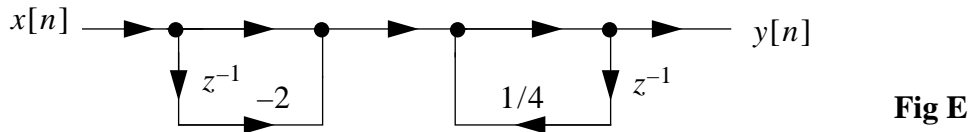
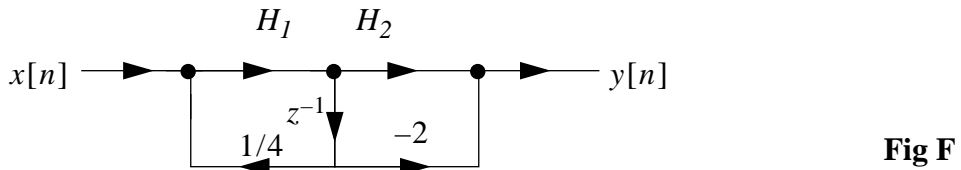


Fig D and Fig E are equivalent, can always flip two cascade sections. Form in Fig D preferred ( $H_1 =$  denominator,  $H_2 =$  numerator) since from it realize node 2 = node 3, can omit one  $z^{-1}$  delay.



Note eqns are recursive (recurrent). Numerator term is moving average ( $H_2$  in Fig F). Denominator term is recursive part ( $H_1$  in Fig F).

(4) Let's do general case now

$$H(z) = \frac{b_0 + b_1z^{-1} + \dots + b_Mz^{-M}}{1 - a_1z^{-1} - \dots - a_Nz^{-N}} = \frac{\sum_{k=0}^M b_kz^{-k}}{1 - \sum_{k=1}^N a_kz^{-k}} = H_1H_2 \quad (10)$$

↓ numerator  
↑ denominator

$$\text{Solution is } y[n] = \sum_{k=1}^N a_k y[n-k] + \underbrace{\sum_{k=0}^M b_k x[n-k]}_{v[n]} \quad (11)$$

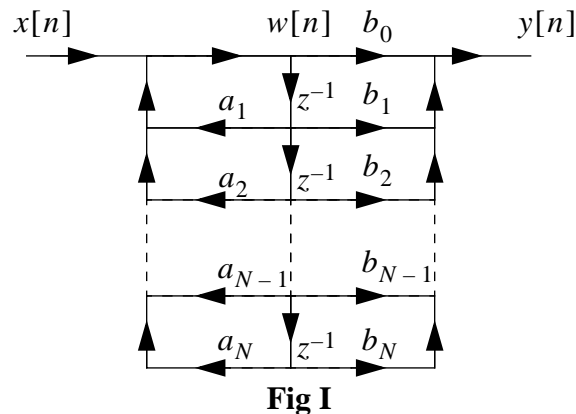
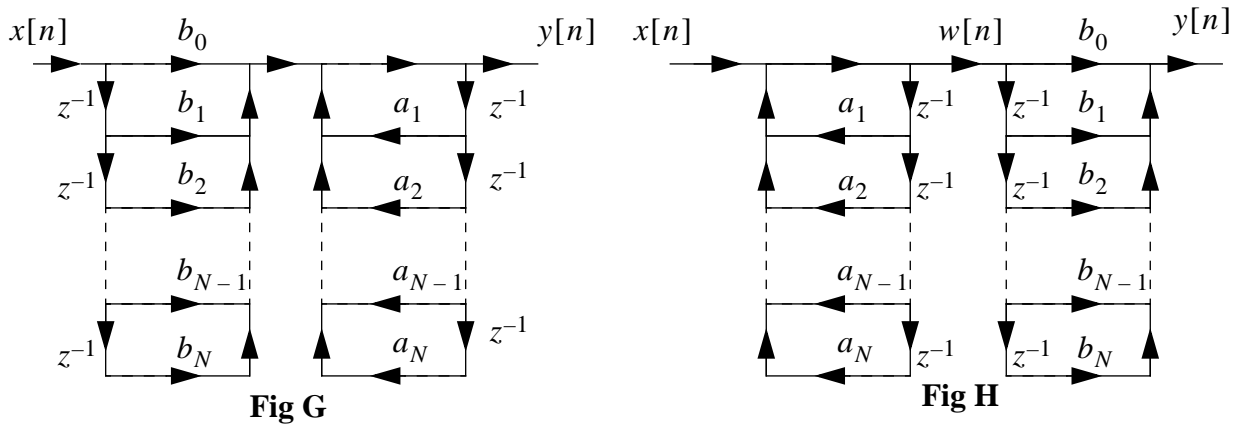
Implement by calculating  $v[n]$  and then RHS of Eq (11) using  $v[n]$ . This results in Fig G (assume  $N = M$ ).

$$\text{Alternatively, implement } w[n] \text{ where } W(z) = H_2(z)X(z) = \frac{1}{1 - \sum a_k z^{-k}} X(z) \quad (12)$$

$$\text{Then implement } y[n] \text{ where } Y(z) = H_1(z)W(z) = (\sum b_k z^{-k})W(z) \quad (13)$$

Result is Fig H is better, since it reduces to Fig I ( $N = M$  assumed).

Fig G is *direct form I*, Fig I is *direct form II*.

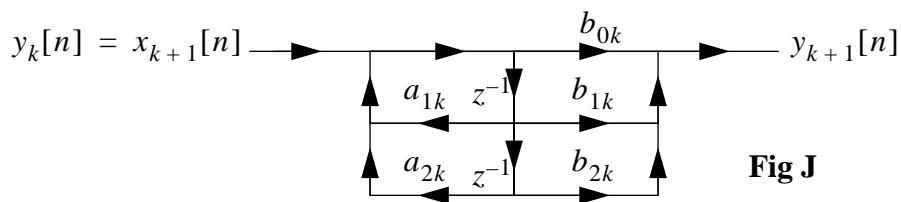


(5) Consider general case in (10) when numerator and denominator are now products of separate second order sections

$$H(z) = \prod_{k=1}^{N_s} \frac{b_{0k} + b_{1k}z^{-1} + b_{2k}z^{-2}}{1 - a_{1k}z^{-1} - a_{2k}z^{-2}} \quad (14)$$

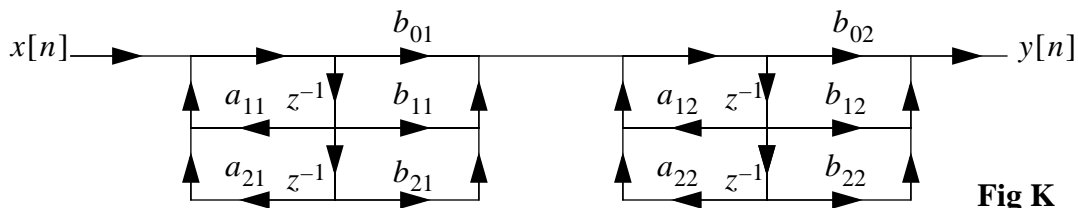
where  $N_s$  is the integer part of  $\frac{(N+1)}{2}$ .

If there are an odd number of real poles (or zeros), one of the coefficients  $b_{2k}$  (or  $a_{2k}$ ) will be 0. We obviously implement the system as a cascade of  $N_s$  2nd-order sections. Note that - signs are present in the assumed denominator in (14). (MATLAB CALCULATED  $a_{1k}$  AND  $a_{2k}$  VALUES IN `roots()` MUST THUS BE NEGATED). Each stage looks like a  $N = 2$  order version of Fig. I. The  $k$ -th stage looks like



**Fig J**

The design for a  $N = 4$  th order filter is shown in Fig K



**Fig K**

The notation  $a_{1k}, a_{2k}$  is standard,  $k =$  stage

Fig J is called a *biquad*.

Fig K is called a *cascade design using biquad sections*.

All 5 designs are IIR (recursive, feedback), have poles 3.9 - 3.11

$$(6)H(z) = b_0 + b_1z^{-1} + \dots + b_{N-1}z^{-(N-1)} = \sum_{n=0}^{N-1} b_nz^{-n} \quad (15)$$

The impulse response for this system is (definition on VG 1)

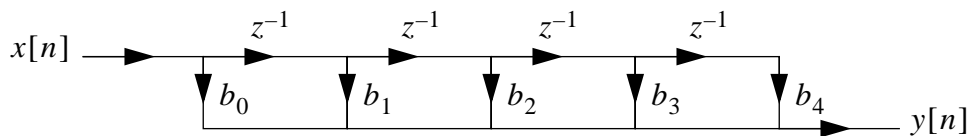
$$h[n] = b_n, \text{ for } 0 \leq n \leq N-1 \quad (16)$$

The associated difference equation for the system is

$$y[n] = b_0x[n] + b_1x[n-1] + \dots + b_{N-1}x[n-N+1] \quad (17)$$

The filter is of order  $N-1$  (largest power of  $z$ );  $N$  coefficients; NOTE DEFINITION

The design for a fourth order ( $N=5$ ) filter is just



**Fig L**

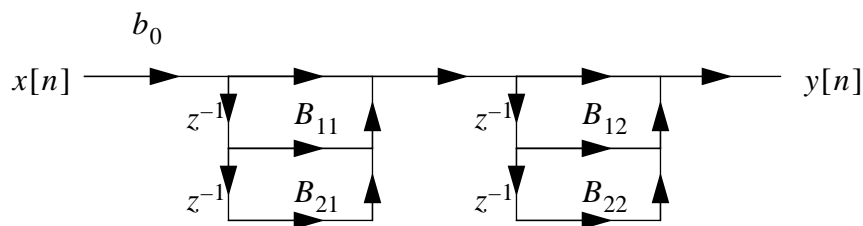
Note

- There is no feedback, Fig L is the *direct solution*.
- It is implemented as a tapped delay line (VG 3-7).

If you write (15) as the product of 2nd order polynomials

$$H(z) = b_0 \prod_{k=1}^{N/2} (1 + B_{1,k}z^{-1} + B_{2,k}z^{-2}) \quad (18)$$

The 4th order ( $N=5$ ) *cascade biquad design* is



**Fig M**

*This is an FIR filter (non-recursive)*

NOTATION

Form in (15) is standard one

order = highest power of  $z$  is  $N-1$

There are  $N = \text{order} + 1$  terms or coefficients

**NOTE THESE DEFINITIONS (AND WHY)**

Case (7) same as (15), Case 6, except *impulse response  $h[n]$  is even symmetric* i.e.

$$h[n] = h[N - 1 - n] \quad (19)$$

e.g.  $h[0] = h[N - 1], h[1] = h[N - 2], 0 \leq n \leq N - 1$ .

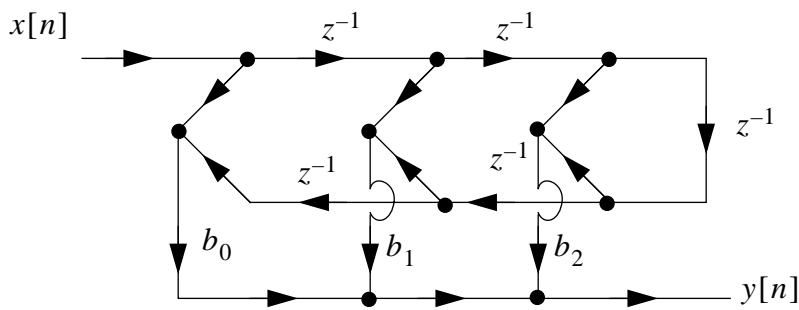
The difference equation in (17) is repeated

$$y[n] = b_0x[n] + b_1x[n - 1] + \dots + b_{N-2}x[n - N + 2] + b_{N-1}x[n - N + 1] \quad (17)$$

This becomes (recall  $h[n] = b_n$ ) using (19)

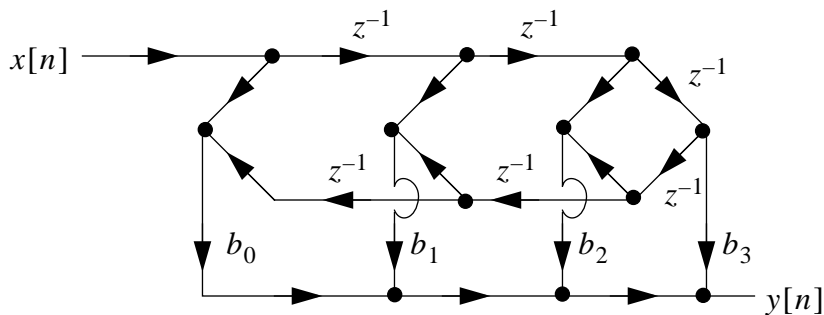
$$\begin{aligned} y[n] &= b_0x[n] + b_1x[n - 1] + \dots + b_1x[n - N + 2] + b_0x[n - N + 1] \\ &= b_0(x[n] + x[n - N + 1]) + b_1(x[n - 1] + x[n - N + 2]) + \dots \quad (20) \end{aligned}$$

The flowgraph for this as a 5th order ( $N = 6$ ) filter is



**Fig N**

The flowgraph for this as a 6th order ( $N = 7$ ) filter is



**Fig O**

Aside - In any  $H(z)$ , if all powers of  $z^{-1}$  are not present, you can use longer delays  $z^{-3}$  etc in flowgraphs (when intermediate delay taps are not needed)

Cascade versions of these direct designs are also possible.

*This is FIR filter with linear phase (symmetric  $h$ ) type 1,  $h$  is even.*

## IIR And FIR Filters (pick $a_n$ and $b_n$ )

Filter types (IIR and FIR)

Filter design (select the  $a_n$  and  $b_n$  coefficients in polynomials)

Filter implementation or structure (which flowgraph)

Only useful refs (Matlab and Openheim and Schafer)

Then, filters in DSP

**IIR** (inifinite-duration impulse response) filters have an impulse response  $h[n]$  that is infinite in extent (They have at least 1 pole).

They have  $H(z)$  of the form

$$H(z) = \frac{\sum b_n z^{-n}}{\sum a_n z^{-n}}, \quad \text{order } N \text{ (largest } z^{-n} \text{ power in denom)}$$

They are described by recursive difference equations and  $(a, b)$  and implemented efficiently by recursive methods ARMA.

**FIR** (finite-duration impulse response) filters have an impulse response  $h[n]$  of finite extent (no non-zero poles).

They have  $H(z)$  of the form  $H(z) = \sum b_n z^{-n}$ , order  $N$  is highest power of  $z^{-n}$ .

Describe by  $h[n]$  values (these =  $b_n$ ), thus finite extent.

Nonrecursive, moving average, MA, filter preferable as length of output is length of signal (obviously want filtered signal equal to length of signal, or signals overlap). Tapped delay line.

### Filter Specs LPF example

$\delta$  = Ripple in passband, ripple in stopband

$N$  = order, Rate of rolloff (higher order filters roll off faster)

$A_s$  = Attenuation (how far down is stopband response)

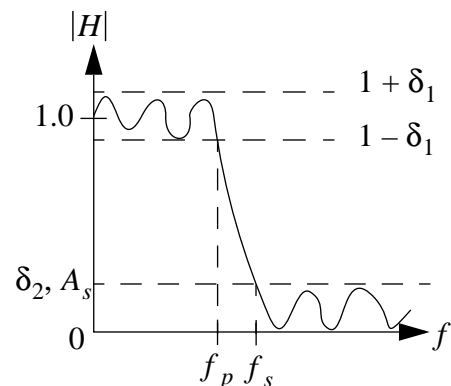
How linear is phase response of filter

$f_p$  = end of passband (where  $\delta_1$  and  $H$  cross)

$f_s$  = start of stopband (where  $\delta_2$  and  $H$  cross)

$f_m$  = maximum freq of interest (need for normalization)

$f_s - f_p$  = transition width;  $f_c = f_r = \text{cutoff freq} = 0.5 |H_m| = 3 \text{ dB}$



$f_s$  ( $s = \text{sampling}$ ) = sampling frequency

**IIR Filters** ( $a_n, b_n$  determine which IIR filter types you have)

Can give flattest response in pass and stop bands (Butterworth)

Can give equal ripple in pass *or* stop bands (Chebyshev)

Can give equal ripple in pass *and* stop bands (Elliptic)

Elliptic gives fastest rolloff (for given  $N$ ) but very nonlinear phase response.

**Butterworth pretty linear phase, converts to digital better.**

**All IIR require lower order  $N$  (fewer taps, hardware) than FIR (often  $\times 100$ ) especially if want low transition width and low  $\delta_1$ .**

**FIR Filters**

**Larger order, more calculations per tap.**

**Can achieve precise linear phase (Butterworth IIR close)**

**Filter types (within FIR and IIR)**

LPF, HPF, BPF, Notch

Once you have ( $a_n, b_n$ ) for one type, easy to get another type (just do substitution)

**We'll do only BW (Butterworth) LP (low pass)**

**MISC**

**Cascade designs much less sensitive (better) than direct for coeff quantization**

**Direct designs OK and most used FIR ones (zeros are widely spaced)**

Can cascade filter and all pass inverse and correct phase errors. But lots of hardware

**IIR** = Cases (1) - (5) pp 4-8 to 4-11, Direct, Cascade

**FIR** = Cases (6) - (7) pp 4-12 to 4-13, case (7) = linear phase

**Adaptive Filters** ( $a_n, b_n$  change) use FIR due to nature of algorithm and adaptive weight calculations.

**Nice Project Analysis topic** - sensitivity of filter coefficients

# Analog IIR Filter Design - 1

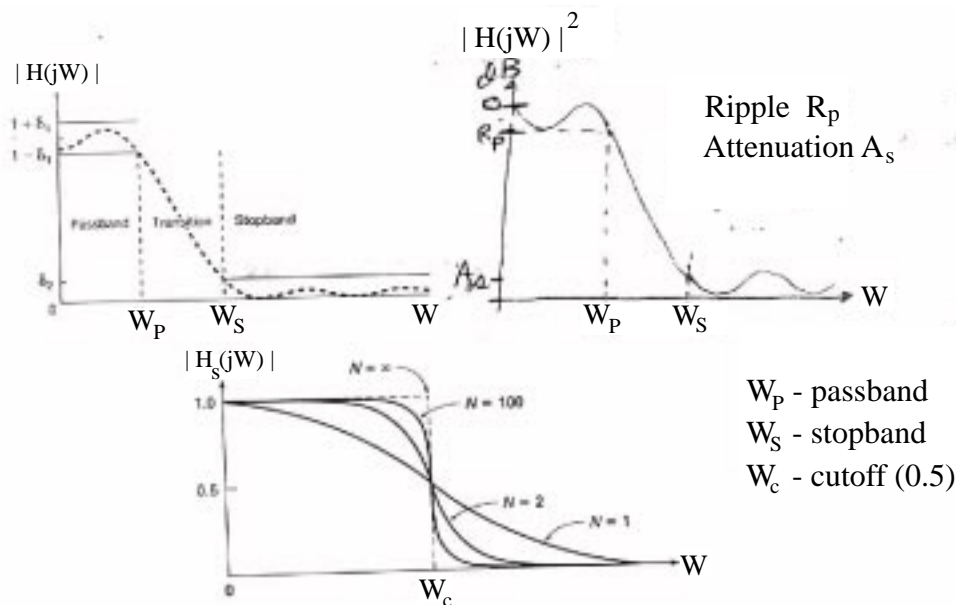
Refs: S & S (446-7, 703-6), Matlab (Chap 8), O & S

**LPF Example**, Gain = 1, cutoff freq =  $W_c$  = rad/sec, order =  $N$  ( $W$  denotes analog,  $\omega$  digital)

Butterworth filter (analog  $H_a$ ), all pole filter

$$\text{magnitude squared response is } |H_a(jW)|^2 = \frac{1}{1 + \left(\frac{W}{W_c}\right)^{2N}} \quad (1)$$

As  $N$  increases, filter gets more ideal, narrower transition region, better attenuation  $A_s$  (stop band ripple  $\delta_2$ ). IIR filters don't control phase ( $\angle H(e^{jW})$ ), so consider *magnitude* or *dB* vs frequency response



Butterworth filter is *maximally flat* in passband (at  $W = 0$ ); all derivatives are zero. LPF cutoff freq  $W_c$  is where  $|H_a(jW)|$  is 0.5 of peak value in (1)

Butterworth solved the above problem and derived Butterworth polynomials in  $s$  for denominator in (1), both factored and unfactored ( $p = s/W_c$ , VG 3-18)

Thus, once filter order  $N$  is chosen, just read off polynomials (denominator in (1))

**MISC**  $R_p$  and  $A_s$  in dB and absolute value (Thus, use  $|H|^2$  not  $H$ )

$$A_s \text{ (dB)} = -20 \log_{10} [\delta_2 / (1 + \delta_1)] = \text{min/max}$$

$$R_p \text{ (dB)} = -20 \log_{10} [(1 - \delta_1) / (1 + \delta_1)]$$

$$\delta_1 = \frac{1 - 10^{-R_p/20}}{1 + 10^{-R_p/20}}, \quad \delta_2 = (1 + \delta_1) 10^{-A_s/20}$$

## Analog IIR Design - 2

$p + 1$	$p + 1$
$p^2 + 1.4121p + 1$	$p^2 + 1.4121p + 1$
$p^3 + 2p^2 + 2p + 1$	$(p + 1)(p^2 + p + 1)$
$p^4 + 2.61313p^3 + 3.41421p^2 + 2.61313p + 1$	$(p^2 + 0.7653p + 1)(p^2 + 1.848p + 1)$
$p^5 + 3.23607p^4 + 5.23607p^3 + 5.23607p^2 + 3.23607p + 1$	$(p + 1)(p^2 + 0.6180p + 1)(p^2 + 1.618p + 1)$
$p^6 + 3.8637p^5 + 7.4641p^4 + 9.14162p^3 + 7.4641p^2 + 3.8637p + 1$	$(p^2 + 0.5176p + 1)(p^2 + 1.414p + 1)(p^2 + 1.932p + 1)$

### How to Determine Order $N$ (and $W_c$ )? Needed in IIR (Butterworth)

$W_c$  is cutoff freq where  $|H_a| = 0.5$  of peak response.

Specs given:  $W_p$  (passband cutoff),  $W_s$  (stopband cutoff),  $R_p$  (ripple in passband),

$A_s$  (stopband attenuation)

In practice: all  $W$  in rad/sec or  $f$  in Hz,  $|R_p|$  and  $|A_s|$  in dB.

### Theory

To select  $N$ , we are given  $W_p$  (passband cutoff),  $W_s$  (stopband cutoff),  $R_p$  and  $A_s$  (in dB)

We need to find  $N$  and  $W_c$  (cutoff freq,  $|H_a| = 0.5$ ). We want

$$\text{at } W = W_p, \text{ want } -10\log_{10}|H_a(jW)|^2 = R_p \text{ or } -10\log_{10}\left[\frac{1}{1 + (W_p/W_c)^{2N}}\right] = R_p$$

$$\text{at } W = W_s, \text{ want } -10\log_{10}|H_a(jW)|^2 = A_s \text{ or } -10\log_{10}\left[\frac{1}{1 + (W_s/W_c)^{2N}}\right] = A_s$$

$$\text{Solutions are } N = \frac{\log_{10}[(10^{R_p/10} - 1)/(10^{A_s/10} - 1)]}{2\log_{10}(W_p/W_s)} \quad (4)$$

Pick  $N$  to be smallest integer  $\geq$  Eq (4)

Pick  $W_c$  to meet or exceed specs at  $W_p$  or  $W_s$

$$\text{To satisfy spec at } W_p \text{ exactly } W_c = \frac{W_p}{\sqrt[2N]{(10^{R_p/10} - 1)}} \quad (5)$$

$$\text{To satisfy spec at } W_s \text{ exactly } W_c = \frac{W_s}{\sqrt[2N]{(10^{A_s/10} - 1)}} \quad (6)$$

This yields range of  $W_c$  (Matlab uses (6) to calculate  $W_c$ ).

# MATLAB SW (IIR, BUTTERWORTH)

## Notation

$w$  = digital ( $0-\pi$ ),  $W$  = analog;  $w' = \tilde{w} = (0-1)$ ,  $W' = \tilde{W} = (0-1)$  normalized.

To normalize, divide by largest freq of interest  $W_{SAMP}/2$ , usually  $2W_s$  (twice stop band cutoff)

$W_c$  here is  $W_n$  in Matlab

## Conversions

$\tilde{w}$  or  $\tilde{W}$  can be rad/sec or Hz

$$w \text{ is linear with } W; \quad \frac{W}{W_{SAMP}/2} = \tilde{W}; \quad \frac{2W}{W_{SAMP}} = \frac{2f}{f_{SAMP}} = \tilde{w} = \tilde{W}, \quad w = \tilde{W}\pi$$

Applies whether  $w$  or  $W$  is rad/sec or Hz

**MATLAB SW** (see help on each)

`buttord(  $\tilde{W}_P, \tilde{W}_s, R_p, A_s, s$  ) = [  $N, \tilde{W}_n$  ]`

Given specs, it calculates order  $N$  and  $\tilde{W}_n = \tilde{W}_C$  for *analog* filter, all  $\tilde{W}$  norm 0-1,  
 $R$  and  $A$  in |dB|.

`buttord` (no  $s$ ) tells order  $N$  and  $\tilde{w}_n = \tilde{w}_c$  needed for *digital* filter 0 – 1 .

`buttap(  $N$  ) = [  $\tilde{z}, \tilde{p}, \tilde{k}$  ]` gives poles  $\tilde{p}$  and zeros  $\tilde{z}$  for analog order  $N$  filter (it factors polynomial) This is for  $W_n = 1$ , scale as  $\tilde{p}_i W_n = p_i$ , Gain  $\tilde{k} W_n^N = k$ . Cascade design.

`real and[ poly(  $\tilde{z}$  ) ] =  $c_n$`  = polynomial coeffs for product of a number of zeroes.

These are *analog* filter coefficients. Direct design.

`butter(  $N, \tilde{w}_n$  ) = [  $b, a$  ]` gives *digital* coeffs ( $b_n, a_n$ ) for a filter of order  $N$  and  $\tilde{w}_n$ .

$$a = a_1, \dots, a_N, a_0 = 1, b = b_0, \dots, b_N.$$

`bilinear(  $c, d, W_{SAMP}$  ) = [  $b, a$  ]` gives *digital* coeffs ( $b_n, a_n$ ) for a given analog design with coefficients ( $c_n, d_n$ ),  $c_n$  is num,  $d_n$  is denom. It uses BL (bilinear) transform

Note  $b_n$  is num,  $a_n$  is denom for a *digital filter*. See conventions below.

`roots( coeffs of polynomial ) = roots`, useful in obtaining cascade design.

General convention: Digital filters =  $\frac{\sum b_n z^{-n}}{\sum a_n z^{-n}} \rightarrow \frac{B}{A}$

Analog filters =  $\frac{\sum c_n s^n}{\sum d_n s^n} \rightarrow \frac{C}{D}$

`cplxpair ( )` gives complex conjugate root pairs from polynomial coefficients

**Recall: negate Matlab  $a_n$  coefficients**

## HOW TO DECIDE IF GOOD DESIGN

Plot magnitude and phase response.

In 551, do it vs  $f = \text{Hz}$  (not normalized)

Linearity of phase response, flatness of magn, width of transition band, stopband attenuation

### Plot results (Matlab) - Useful in Lab 2 and HW

`freqz(b, a, M)` = [  $H, \tilde{w}$  ] numerator then denominator ( $b, a$ ). Given  $b_n, a_n$  for digital filter; output is  $H$  vs  $\tilde{w}$  (array for  $H$  vs  $\tilde{w}$ ),  $M$  = number of points in  $\tilde{w}$  in plot.  $H$  is complex (real, imaginary) numbers.

`freqs(c, d, M)` = [  $H, \tilde{W}$  ] is same for analog filter.

`plot(\tilde{W}, abs(H))` = magn response, `plot(\tilde{W}, phase(H))` = phase response

Use `plot(\tilde{W}, unwrap(angle(H)))`, avoids  $2\pi$  discontinuities, only OK up to  $w_s$ , only care about linearity up to  $w_s$ .

Plots in 551 (HW, labs, project) are vs  $f$  (Hz). Thus convert  $\tilde{w}$  or  $\tilde{W}$  to Hz.

# MATLAB SW Details

(GIVEN  $H_a(s)$  (ANALOG DESIGN), HOW TO PRODUCE DIGITAL IIR DESIGN)

Problem is to convert analog  $H_a(s)$  into digital  $H(z)$ , 2 major techniques

**Impulse Response** (poor results; aliasing etc.)

**Bilinear Transform** - most popular/best method

$$s = \left(\frac{2}{T}\right) \frac{1 - z^{-1}}{1 + z^{-1}} \quad (7)$$

This produces  $H(z)$ , get it into standard quotient of polynomials  $z^{-1}$  form with new digital filter coefficients  $a_n, b_n$ . Then do implementations (earlier flowgraphs).

Since digital filter, their specs use digital freqs  $\tilde{w}_p, \tilde{w}_s$  ( $0 - 1$ ). Thus, substitute for rad/sec freqs

using 
$$W_p = \frac{2}{T} \tan\left(\frac{\tilde{w}_p}{2}\right), W_s = \frac{2}{T} \tan\left(\frac{\tilde{w}_s}{2}\right) \quad (8)$$

`buttord` in Matlab does this and outputs order  $N$  and  $\tilde{w}_n$  needed. It does this from specified  $\tilde{w}_p, \tilde{w}_s, R_p$  and  $A_s$ . (Digital Filter)

`bilinear` in Matlab does this and outputs  $(a_n, b_n)$  digital coeffs from analog  $(c_n, d_n)$  ones.

`butter` in Matlab does this and outputs  $(a_n, b_n)$  digital coeffs from order  $N$  and  $\tilde{w}_n$  (from `buttord`).

## Non LPF IIR filters

Given a LPF  $H(z)$  design, you can produce  $H(z)$  for a different  $\omega_c$  or for HPF, BPF, etc by substituting for  $z^{-1}$  as below

Type of Transformation	Transformation	Parameters
Lowpass	$z^{-1} \rightarrow \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}}$	$\omega_c =$ cutoff frequency of new filter $\alpha = \frac{\sin [(\omega'_c - \omega_c) / 2]}{\sin [(\omega'_c + \omega_c) / 2]}$
Highpass	$z^{-1} \rightarrow -\frac{z^{-1} + \alpha}{1 + \alpha z^{-1}}$	$\omega_c =$ cutoff frequency of new filter $\alpha = -\frac{\cos [(\omega'_c + \omega_c) / 2]}{\cos [(\omega'_c - \omega_c) / 2]}$
Bandpass	$z^{-1} \rightarrow -\frac{z^{-2} - \alpha_1 z^{-1} + \alpha_2}{\alpha_2 z^{-2} - \alpha_1 z^{-1} + 1}$	$\omega_\ell =$ lower cutoff frequency $\omega_u =$ upper cutoff frequency $\alpha_1 = -2\beta K / (K + 1)$ $\alpha_2 = (K - 1) / (K + 1)$ $\beta = \frac{\cos [(\omega_u + \omega_\ell) / 2]}{\cos [(\omega_u - \omega_\ell) / 2]}$ $K = \cot \frac{\omega_u - \omega_\ell}{2} \tan \frac{\omega'_c}{2}$
Bandstop	$z^{-1} \rightarrow \frac{z^{-2} - \alpha_1 z^{-1} + \alpha_2}{\alpha_2 z^{-2} - \alpha_1 z^{-1} + 1}$	$\omega_\ell =$ lower cutoff frequency $\omega_u =$ upper cutoff frequency $\alpha_1 = -2\beta / (K + 1)$ $\alpha_2 = (K - 1) / (K + 1)$ $\beta = \frac{\cos [(\omega_u + \omega_\ell) / 2]}{\cos [(\omega_u - \omega_\ell) / 2]}$ $K = \tan \frac{\omega_u - \omega_\ell}{2} \tan \frac{\omega'_c}{2}$

**Matlab software for above** (All  $\tilde{w}$  are normalized (0 – 1))

LowPass     `butter(N,  $\tilde{w}_n$ ) = [ B, A]`

HighPass     `butter(N,  $\tilde{w}_n$ , 'high') = [ B, A]`

BandPass     `butter(N,  $\tilde{w}_n$ )`

BandStop     `Butter(N,  $\tilde{w}_n$ , 'stop')`

These filters are of order  $2N$

For bandpass and bandstop  $\tilde{w}_n = [ \tilde{w}_1, \tilde{w}_2 ] =$  pass and stop band limits.

## FIR Filters - 1

Only linear phase FIR considered (only reason for FIR vs IIR)

These have form 
$$H(z) = \sum_{n=0}^{N-1} b_n z^{-n} \quad (1)$$

and freq response 
$$H(e^{j\omega}) = \sum b_n e^{-j\omega n}, \quad -\pi \leq \omega \leq \pi, \quad (2)$$

For linear phase  $\angle H(e^{j\omega}) = -\alpha\omega$  and  $h[n] = b_n$  is symmetric

- I'll consider only type 1 (symmetric,  $N = \text{odd}$ ). Then  $\alpha = (N - 1)/2$  is integer,  $h[n] = h[N - 1 - n]$ . Some of 4 possible types not OK for HP, bandstop, LP, etc.
- For FIR, we plot magnitude and phase response of  $H(e^{j\omega})$  (amp response will/can go  $\pm$  )
- Impulse response  $h[n] = b_n$  is finite (0 to  $N - 1$ ), order is  $N - 1$ , causal.

Can show all linear phase FIR filters have form

$$H(e^{j\omega}) = e^{j\beta} e^{-j(N-1)\frac{\omega}{2}} H_r(\omega) \quad (3) \quad H_r = \text{real amplitude response}$$

For Type 1 case,  $\beta = 0, H_r(\omega) = \sum_{n=0}^{(N-1)/2} a[n] \cos \omega n \quad (4)$

where  $a[n]$  is an  $L$  order polynomial  $L = (N - 1)/2$

### DESIGN METHODS (Find the $b_n$ in (1))

Various methods (window, frequency sampling) can't specify  $\omega_p$  and  $\omega_s$  precisely (must accept what you get), must have  $\delta_1 = \delta_2$  or can optimize only  $\delta_2$ , and approximation error (from ideal response) is not uniform over frequency (it is largest near band edge). Method considered is **minimax Parks-McClellan** (see HW2, Lab 3).

Idea of algorithm follows.

Ideal LP linear phase FIR has  $H_d(e^{j\omega}) = 1 e^{-j\alpha\omega}$  for  $|\omega| \leq \omega_c$  and 0 elsewhere. For a given H design, calculate the error  $E(\omega)$  vs  $\omega$  in the amplitude response. We then minimize the maximum error at  $L + 2$  values of  $\omega$  and find the associated  $a[n]$  in (4) that do this. The Parks-McClellan algorithm is an iterative way to select the best  $L + 2$  places and the associated  $L$ -order polynomial in  $H_r$  that fits best. This is  $b_n$  in (1). From it, you get  $a[n]$  in (4), e.g.

$a_0 = h[(N - 1)/2] \dots a_n = 2h[(N - 1)/2 - n]$  for  $n = 1$  to  $(N - 1)/2 - 1$ . Matlab `remez` does this, produces  $h$ . Then implement (VG 3-14).

You must guess  $N$ . Initial good estimate is

$$N = \frac{-20 \log_{10} \sqrt{\delta_1 \delta_2} - 13}{14.6 \Delta f} + 1, \quad \Delta f = \frac{(\tilde{w}_s - \tilde{w}_p)}{\pi} \quad (4)$$

You produce a design (using `remez`) for a given  $N$ , you check the stopband attenuation  $A_s$ ; if it is not adequate, you change  $N$  and get a new design. Check ripple for final design only (HW2).

**DESIGN FIR** [Use Matlab `remez` to calculate the  $b_n = h[n]$

Various `remez` in Matlab (different options) - see **HELP**

Specs give  $\delta_1$  (passband) and  $\delta_2$  (stopband) ripple, these are related to  $R_p$  and  $A_s$  (VG 3-16)

`remez` algorithm tries to make error minimum over all  $\omega$  (i.e. same in pass and stop).

If no weights are given to error, tries to give same ripple (and error) in pass/stop bands.

With weights, can have  $\delta_1 \neq \delta_2$ .

This is more generally desirable (when  $R_p$  and  $A_s$  or  $\delta_1$  and  $\delta_2$  are specified).

Use `remez[ N-1,  $\tilde{f}$ ,  $\underline{m}$ , weights] = [ h ] = b_n` in design.

↑ Enter  $N$  from (4) - 1 here. Caution, in Matlab help says  $N$  is order (it is  $N - 1$ ).

**What you enter here is the order**

$\tilde{f}$  and  $\underline{m}$  are vectors of normalized frequency and desired response. (up to  $L+2$  samples)

In the example below,  $\underline{f}$  can be Hz, rad/sec, or digital, since normalized.

Can specify  $\underline{f}$  and  $\underline{m}$  at up to  $L + 2$  freqs.

e.g.

$\tilde{f}$	=	0	$\tilde{f}_p$	$\tilde{f}_s$	1	}	I discuss
$\underline{m}$	=	1	1	0	0		
expected ripple		$\delta_1$	$\delta_1$	$\delta_2$	$\delta_2$		
		└───┘		└───┘			
		pass		stop			

if specified weights (for errors) are  $\frac{\delta_2}{\delta_1}$  1 Then the error is  $\frac{\delta_2}{\delta_1} \left[ \begin{matrix} \delta_1 \\ \text{ideal} \end{matrix} \right] = \delta_2$ .

Thus, same error  $\delta_2$   $\delta_2$  in both bands, same ripple in each.

Weights is vector half length of  $\underline{f}$  and  $\underline{m}$  in pairs (bands).

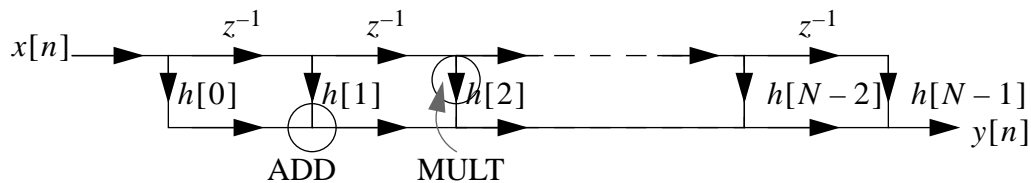
## DSP Considerations

**FIR** Denote the  $b_n$  as the filter  $h[n]$ ,  $n = 0$  to  $N - 1$ ,  $N =$  order. Note, there are  $N$  coefficients.

Difference equation  $x[n]$  = input and  $y[n]$  = output is

$$y[n] = h[0]x[n] + h[1]x[n - 1] + \dots + h[N - 1]x[n - (N - 1)] \quad (1)$$

Flowgraph follows (tapped delay line)



time =  $n$ , delayed input samples  $x[n - k]$  produced by delays or SR. Delayed input  $x[n - k]$  is multiplied by “tap weight”  $h[k]$ . For a given contents of the delay line, the  $x[n - k]$  are each multiplied by the  $h[k]$  and their sum is formed at  $y[n]$ . As the next  $x[n]$  sample enters, the contents of the delay line shift right (the last sample on the right is shifted out).

The output  $y[n] = h[n] * x[n] = \sum_k h[k]x[n - k]$ . Thus, we need to do a convolution (in DSP).

### DSP

**Parallel mult/add instruction** - need  $N$  instructions (cycles) for each input sample (time shift  $n$ ). Do a coefficient multiplication in one section of the delay line in parallel with the running sum of the product produced in the prior section (register)

**Circular addressing** At any time  $n$ , you need the current input  $x[n]$  and the  $N - 1$  prior input samples to calculate  $y[n]$ . These  $N$  samples of  $x[n]$  must be stored (a memory buffer with  $N$  elements). When the next sample arrives, you could put it in the first location in the buffer, shift the prior contents of the buffer down by 1 and have the oldest sample pushed out of the buffer. Moving  $N$  samples takes  $N$  instructions. **Circular addressing** does the shift much faster. **You don't move data**. An address pointer is used that circularly moves in the buffer (when it reaches the bottom it then goes back to the top). Thus, you load the newest sample into the buffer location of the oldest sample that is no longer needed and overwrite it. The pointer now points to the location you just loaded, it is the new  $x[n]$ , the next one is the new  $x[n - 1]$  etc. The C67 has registers that operate independent of the ALU and multiplier. They can calculate the data address and perform the shift in the same instruction cycle in which you multiply and add the products. The BK register should be initialized to  $N$  to control the size of the circular buffer, **the starting buffer address must be a multiple of the smallest power of 2  $\geq N$**  (e.g., if  $N = 20$ , the first address must be  $\geq 2^5$ , set the lower 5 bits of the address to 0).

**Single Repeat** (rpts). Set RC =  $N - 1$ , then rpts will repeat an instruction (mult/add)  $N$  times.

**Debugger** You can view an array as floating point numbers by `disp * (float *) array` if you declare the array as a global. To view register `x` as floating point, use `disp f x`.

## IIR

- *These filters can need very precise coefficient values (floating point EVM needed). The cascade design is better than the direct one in this regard.*
- You may need to scale the values within each biquad within a cascade design.
- Direct form I (Fig G, VG 3-10) produces 2 convolution sums. You can implement them as 2 separate FIR filters and combine their sums to produce the next output.
- Direct form II (Fig I, VG 3-10) also does 2 convolution sums on the  $w[n]$ . You need to calculate  $w[n]$  first.
- For cascade (Fig K, page 3-11), use one length-3 circular buffer for each biquad. For Direct form I (Fig G), use 2 circular buffers (one for the  $x[n]$  input and one for the  $y[n]$  output). Direct form II (Fig I) needs only 2 circular buffers.

## GENERAL

The DSP is floating point. The D/A is 16 bit fixed point. The output must be in the range  $\pm 2^{15}$ .