

Part 2

Signal Flowgraphs

Filter Structures

State Variables

Filter Responses and Properties

FIR Filter Design Techniques

IIR Filter Design Techniques

2.1 Signal Flowgraphs

- Replace block diagrams
- Nodes and Directed Branches
- Node j has a value w_j which is the sum of the inputs to the node.
- Branch ij from node i to node j implements an operation f_{ij}
- The input to branch i is w_i
- The input to node j from branch ij is $f_{ij}(w_i)$

Essentially, each node is a summing node, and each branch is a block.

2.1.1 Examples:

1. 2nd-order FIR (Transverse Form): figure 8.5 (p. 433)
2. Simple example of FIR: moving average (lowpass) filter given by $H_0(z) = (1 + z^{-1})/2$
3. 2nd-order IIR (Direct Form I): figure 8.6 (p. 433)
4. Simple example of IIR: fading memory (lowpass) filter given by $H(z) = k/(1 - az^{-1})$, or $H(z) = k(1 + z^{-1})/(1 - az^{-1})$.

2.1.2 Theoretical:

1. Mason's Gain Formula: gain (transfer function) from input to output involves
 - (a) Gain along *forward paths*: directed paths from input to output in which no node is traversed more than once
 - (b) Gains around *loops*: closed directed paths in which no node is traversed more than once.
 - (c) *Nontouching* paths: no nodes in common.

Then the gain is given by $\sum_i P_i \Delta_i / \Delta$, where

- (a) The sum is over all forward paths
- (b) $\Delta = 1 - (\sum \text{gains of all loops}) + (\sum \prod \text{gains of all pairs of nontouching loops}) - (\sum \prod \text{gains of all triples of nontouching loops}) + \dots$
- (c) $\Delta_i = \Delta$ with the gain of all loops which touch the i -th forward path set to zero

2. Transposed Forms: reverse directions of all arrows; does not change gain of single-input, single-output signal flow-graph — section 8.2.(p. 430)

2.2 Filter Forms

1. FIR drawn as a signal flow-graph: figure 8.5 (p. 433)
2. IIR as a signal flow-graph (Direct Form I): figure 8.13 (p. 438)
3. Direct Form II: figure 8.14 (p. 439)
4. Transposed Direct Form I and II.
5. Cascade Forms: factor numerator and denominator into second-order factors; figures 8.18 and 8.19 (p. 441)
6. Parallel Forms: expand as partial fractions; figures 8.20 and 8.21 (pp. 442, 443)
7. State variable forms (next)
8. Lattice Forms (figure 8.2, p. 430)

2.3 State Variables

For a discrete-time signal flow-graph, the current outputs of the delay branches are a natural set of state variables.

State equations have the form:

$$\begin{aligned}\mathbf{x}(n + 1) &= A\mathbf{x}(n) + Bu_n \\ y_n &= C\mathbf{x}(n) + du_n\end{aligned}$$

— linear, time-invariant, single-input, single-output.

To get state equations (algorithm) from signal flow-graph:

- Write next state in terms of present state and present system input(s)
- Next state = present *inputs* of delay branches
- So write present inputs of delay branches in terms of present system inputs, and present outputs of delay branches
- Also write present system outputs in terms of present system inputs, and present outputs of delay branches
- Works if the signal flow-graph is **well-posed**, that is, there are *no delay-free loops*.

Summary:

Have four descriptions of system:

1. Difference equations
2. Transfer function
3. Signal flow-graph
4. State equations (special type of difference equations)

The \mathcal{Z} -transform and its inverse translate between descriptions 1 and 2; Mason's formula gets from 3 to 2; the procedure described above gets from 3 to 4; to translate any other description to 3, begin by drawing the delay branches, and then connect to the delay branch inputs; and to translate from 4 to 2, use the formula

$$H(z) = C(zI - A)^{-1}B + D$$

To prove this formula, take the \mathcal{Z} -transform of the state equations, solve the first equation for $\mathbf{X}(z)$, and substitute in the second equation.

State Variables (continued)

Roundoff “Noise” and Overflow

- To check for possible overflow, need gain from input to each node in the signal flow graph
- Essential: gain from each input to each state variable
- Roundoff noise: gain from each summing node to output (most DSP architectures use extended precision registers or accumulator, and so most of the error occurs when saving to memory, after a chain of multiply-adds).
- Need to balance between overflow and roundoff error

2.3.1 Examples:

1. Notch filter:

$$(z) = \frac{1 - (2 \cos \theta_0)z^{-1} + z^{-2}}{1 - (2r \cos \theta_0)z^{-1} + r^2 z^{-2}}$$

where θ_0 is the notch angular frequency, and $2(1 - r)$ is its 3-db width. Here $0 < r < 1$, and in most cases, r is very close to 1, e.g., $f_S = 5\text{kHz}$, $f_{notch} = 60\text{ Hz}$, and $BW = 0.2\text{Hz}$. (Poles and zeros are very close to each other — “nearly not minimal”.)

2. Direct form II for notch filter; look at internal amplitudes — overflow
3. Transposed direct form II for notch filter; look at internal amplitudes
4. Direct form I and transposed direct form I — exercise.
5. Also look at gains from summing nodes to output — noise.

2.4 Filter Responses:

- Lowpass:

$$H(f) = \begin{cases} 1 & \text{if } |f| \leq f_c \\ 0 & \text{if } |f| > f_c \end{cases}$$

- Bandpass

$$H(f) = \begin{cases} 1 & \text{if } f_l \leq |f| \leq f_u \\ 0 & \text{otherwise} \end{cases}$$

- Highpass:

$$H(f) = \begin{cases} 0 & \text{if } |f| \leq f_c \\ 1 & \text{if } |f| > f_c \end{cases}$$

Note: No *genuine* high-pass filters in DSP!

Filter Responses (continued)

- Differentiator (pp. 529 – 530):

$$H(f) = jCf$$

- Hilbert transformer

$$H(f) = \begin{cases} j & \text{if } f < 0 \\ -j & \text{if } f > 0 \end{cases}$$

- Notch (p. 386)
- Band-stop, multiband, matched, Wiener, Kalman, . . .

2.4.1 Filter Properties:

FIR Filters:

- intrinsically stable
- linear-phase capability; see section 7.2.2 (p. 363)
- easily implemented
- easily scaled
- easily controlled roundoff and truncation errors
- simple to design
- high-order for sharp transitions
- high delay (for linear phase)

Filter Properties (continued):

IIR Filters:

- nonlinear-phase
- possible instability
- difficult to scale
- difficult to control roundoff and truncation
- complex design
- low-order, sharp transitions

2.5 FIR Filter Design:

- Windowing: section 10.2 (p. 527)
- Equiripple (Computer-Aided): section 10.3 (p. 541)

2.5.1 Windowing:

- Translate given requirements to discrete-time frequency domain, using $\theta = 2\pi f / f_S$
- Apply inverse discrete-time Fourier transform to the resulting frequency response, $G(\theta)$, to get a noncausal impulse response (g_n).
- This response will normally be infinite; to make it finite, we multiply by a *window function* $w_N(n)$, which is zero for $|n| > N$. We then have a finite, noncausal impulse response, (\tilde{g}_n), given by

$$\tilde{g}_n = \begin{cases} g_n w(n) & \text{if } |n| \leq N \\ 0 & \text{if } |n| > N \end{cases}$$

- Then delay this response by N units to get a causal, implementable impulse response (of order $2N$):

$$h_n = \tilde{g}_{n-N}$$

Windowing (continued)

Effect of window function:

Multiplying by g_n by $w(n)$ is equivalent to convolving $G(\theta)$ with $W(\theta)$ in the θ -domain, where $W(\theta)$ is the DTFT of $w(n)$. This in turn is equivalent to replacing each edge in the ideal response by the indefinite integral of $W(\theta)$.

This has two effects:

- Sharp edges are replaced by finite-width transition regions
- Ripple (approximation error) appears in the passbands and stopbands (the same level of ripple in both)

For this reason, the effect of a window is usually described in terms of $W(\theta)$; the important quantities are the *width of the main lobe*, and the *level of the sidelobes*.

The *shape* of the window is determined by the desired level of ripple; the *order* of the window is then determined by the desired width of the transition regions.

Windowing (continued)

Some windows:

Rectangular (Truncation):

$$w_N(n) = \begin{cases} 1 & \text{if } |n| \leq N \\ 0 & \text{if } |n| > N \end{cases}$$

Peak Approximation Error: -21 dB

Approximate Main Lobe Width: $2\pi/N$

See figure 10.7(a), p.534

Triangular (Bartlett):

$$w_N(n) = \begin{cases} 1 - \frac{|n|}{N} & \text{if } |n| \leq N \\ 0 & \text{if } |n| > N \end{cases}$$

Peak Approximation Error: -25 dB

Approximate Main Lobe Width: $4\pi/N$

See figure 10.7(b), p.534

Windowing (continued)

Hanning:

$$w_N(n) = \begin{cases} \frac{1}{2} \left(1 + \cos \frac{n\pi}{N} \right) & \text{if } |n| \leq N \\ 0 & \text{if } |n| > N \end{cases}$$

Peak Approximation Error: -44 dB

Approximate Main Lobe Width: $4\pi/N$

See figure 10.7(c), p.534

Hamming:

$$w_N(n) = \begin{cases} 0.5435 + .4565 \cos \frac{n\pi}{N} & \text{if } |n| \leq N \\ 0 & \text{if } |n| > N \end{cases}$$

Peak Approximation Error: -53 dB

Approximate Main Lobe Width: $4\pi/N$

See figure 10.7(d), p.534

Windowing (continued)

Blackman:

$$w_N(n) = \begin{cases} .42 + .5 \cos \frac{n\pi}{N} + .08 \cos \frac{2n\pi}{N} & \text{if } |n| \leq N \\ 0 & \text{if } |n| > N \end{cases}$$

Peak Approximation Error: -74 dB

Approximate Main Lobe Width: $6\pi/N$

See figure 10.7(e), p.534

Windowing (continued)

Kaiser:

$$w_N(n) = \begin{cases} \frac{I_0(\beta\sqrt{1-(n/N)^2})}{I_0(\beta)} & \text{if } |n| \leq N \\ 0 & \text{if } |n| > N \end{cases}$$

where β is a parameter which determines the peak approximation error, and I_0 is the modified Bessel function of the first kind and order 0.

Empirical equations for selecting β :

$$\beta = \begin{cases} .11(A - 8.7) & A > 50 \\ .58(A - 21)^{0.4} + .079(A - 21) & 21 \leq A \leq 50 \\ 0 & A < 21 \end{cases}$$

and

$$N = \frac{A - 8}{4.57\Delta}$$

where $A = -20 \log_{10} \delta$ is the peak approximation error in dB, and Δ is the transition width in terms of the discrete-time angular frequency θ .

Windowing (continued)

Example:

Output filter for CD player with 2 times oversampling:
 low-pass filter, with pass-band 0 – 20kHz, stop-band
 24.1kHz – 44.1kHz, unity gain with 1% accuracy or better in
 the pass-band, and at least 60dB rejection in the stop-band.
 Sampling frequency is 88.2kHz.

Solution: First find normalized frequency specification:

$$f_p = 20\text{kHz} \text{ gives } \theta_p = 2\pi \frac{f_p}{f_S} = \frac{200\pi}{441}$$

$$f_r = 24.1\text{kHz} \text{ gives } \theta_r = 2\pi \frac{f_r}{f_S} = \frac{241\pi}{441}$$

Therefore the transition width is

$$\Delta = \theta_r - \theta_p = \frac{41\pi}{441} \approx 0.2921.$$

Now the peak approximation error must be better than both
 1% and 60dB, so must take 60dB.

Then $N \geq 52/(4.57\Delta) \approx 38.96$, and so $N = 39$

Also, $\beta = .1102(A - 8.7) = 5.653$

The filter length is 79 (78-th order), and the window
 coefficients can be calculated directly.

Kaiser window example (continued)

To find the ideal impulse response:

The IDTFT of the ideal low-pass filter with normalized angular cutoff frequency α is

$$\begin{aligned} h_{ideal}(n) &= \frac{1}{2\pi} \int_{-\alpha}^{\alpha} e^{jn\theta} d\theta \\ &= \begin{cases} \frac{\alpha}{\pi} & \text{if } n = 0 \\ \frac{1}{n\pi} \sin(n\alpha) & \text{if } n \neq 0 \end{cases} \end{aligned}$$

The windowed, noncausal response then is

$$h_{nc}(n) = \begin{cases} \frac{220.5}{441} = \frac{1}{2} & \text{if } n = 0 \\ 0 & \text{if } |n| > 39 \\ w_{39}(n) \frac{1}{n\pi} \sin(n\frac{\pi}{2}) & \text{if } 0 < |n| \leq 39 \end{cases}$$

where

$$w_{39}(n) = \frac{I_0(5.653\sqrt{1 - (n/39)^2})}{I_0(5.653)}$$

The final impulse response then is $h_n = h_{nc}(n - 39)$

Windowing (continued)

Other Examples:

- High-pass: ideal response is

$$h_{ideal}(n) = \delta(n) - h_{ideal,lowpass}(n)$$

- Band-pass: get ideal response by difference of two low-pass or high-pass

- Band-limited differentiator: ideal response for $n \neq 0$

$$h_n = \frac{\alpha}{n\pi} \cos(n\alpha) - \frac{1}{n^2\pi} \sin(n\alpha)$$

and $h_0 = 0$.

- Band-limited Hilbert transformer (from α up to β): ideal response for $n \neq 0$

$$h_n = \frac{1}{n\pi} [\cos(n\alpha) - \cos(n\beta)]$$

and $h_0 = 0$.

2.5.2 Equiripple (Computer-Aided) Design:

As before, the first and last steps are:

- Translate given requirements to discrete-time frequency domain, using $\theta = 2\pi fT$
- Use procedure below to get a finite, symmetric, noncausal response g_n
- Then delay this response by L units to get a causal, implementable impulse response (of order $2L$):

$$h_n = g_{n-L}$$

We assume that the g_n are symmetric.

(See section 7.9)

Equiripple Design (continued)

To find the noncausal impulse response, solve a *min-max* optimization problem:

Find the g_n coefficients which minimize the functional

$$E_p(g_0, g_1, \dots, g_L) = \text{Max}_\theta [|E(\theta)|]$$

where (for an even frequency response), the error function $E(\theta)$ is given by

$$E(\theta) = W(\theta) \left(D(\theta) - g_0 - 2 \sum_{n=1}^L g_n \cos n\theta \right)$$

Here $W(\theta)$ is a positive weight function, and $D(\theta)$ is the desired response.

The optimum has the *equiripple* property:

$E_{opt}(\theta)$ must achieve the peak error at at least $L + 2$ points, and the errors must alternate in sign.

See figures 10.17 and 10.18, (p.553).

This optimization problem can be solved iteratively:

- Remez exchange algorithm (general)
- Parks-McClellan algorithm (FIR filter design)

These are two-step iterative algorithms: initialize by picking $L + 2$ points $0 \leq \theta_0 < \theta_1 < \dots < \theta_{L+1} \leq \pi$, and then loop:

1. Solve the following $L + 2$ equations in $L + 2$ unknowns:

$$W(\theta_k) \left(D(\theta_k) - g_0 - 2 \sum_{n=1}^L g_n \cos n\theta_k \right) = \pm e$$

(Note: if these were the equiripple points, and e were the peak error, then we would have the solution.)

2. Evaluate the resulting function

$$G(\theta) = g_0 + \sum_{n=1}^L g_n \cos n\theta$$

and find $L + 2$ extreme points of $G(\theta)$; use these as the new set of $\{\theta_k\}$

3. Iterate.

2.6 IIR filter design methods:

Normally uses knowledge of classical continuous-time filters

- Impulse Invariance –emphasizes time-domain performance (can also use step-invariance).
- Bilinear Transform – emphasizes frequency-domain performance.

2.6.1 Impulse Invariance:

Given a continuous-time impulse response $h(t)$, find a discrete-time system with impulse response $h_n = h(nT)$.

Can be done by expanding $H_c(s) = \mathcal{L}\{h(t)\}$ in partial fractions; for simple poles:

$$H_c(s) = \sum_{k=1}^N \frac{A_k}{s - s_k}$$

Then

$$H(z) = T \sum_{k=1}^N \frac{A_k}{1 - e^{T s_k} z^{-1}}$$

Since h_n is just the sampled version of $h(t)$, the relationship between the frequency responses $H(e^{j\omega T})$ and $H_c(j\omega)$ is given by the sampling theorem.

(See equations 9.55 – 9.58 (p. 517).)

2.6.2 Bilinear Transform:

A transform from the discrete-time z -domain to the continuous-time S -domain which has many of the properties of $z = e^{sT}$; it is defined by:

$$S = C \frac{z - 1}{z + 1} = C \frac{1 - z^{-1}}{1 + z^{-1}}$$

where C is a positive constant.

Note capital S is used to distinguish this from the real-world transformation

$$z = e^{sT}$$

which can't be used for design.

On the unit circle (frequency domain):

$$\begin{aligned} S &= \Sigma + j\Omega \\ &= C \frac{e^{j\theta} - 1}{e^{j\theta} + 1} \\ &= C \frac{e^{j\theta/2} - e^{-j\theta/2}}{e^{j\theta/2} + e^{-j\theta/2}} \\ &= jC \tan \frac{\theta}{2} \end{aligned}$$

So discrete-time frequency is mapped to continuous-time frequency, but there is *frequency warping* given by

$$\Omega = C \tan \frac{\theta}{2}$$

The Bilinear Transform has several other valuable properties:

- Inside of the unit circle is mapped to the left half-plane, and so stability and minimum phase are preserved.
- Since both S and $\frac{z-1}{z+1}$ are first-order rational functions, rational functions are mapped to rational functions, and order is preserved.
- Many frequency-domain properties (e.g., equiripple, maximally flat, etc.) are preserved.
- Filter form (e.g., cascade, parallel, etc.) is preserved.
- Since the transform is a non-linear rescaling of the frequency axis, it works best for piecewise constant amplitude responses, and does not preserve phase responses.

The Bilinear Transform can therefore be used to go from analog to digital filters; it is the most commonly used IIR design technique.

Bilinear Transform (continued)

Bilinear Transform Design Procedure:

1. Translate given requirements to discrete-time frequency domain, using $\theta = 2\pi fT$
2. Apply frequency warping (with $C = 1$) to translate the θ -domain requirements to the (fake) S -domain
3. Find the appropriate filter in the S -domain
4. Use the substitution

$$S = \frac{z - 1}{z + 1} = \frac{1 - z^{-1}}{1 + z^{-1}}$$

(the bilinear transform with $C = 1$) to obtain the desired discrete-time filter.

Bilinear Transform (continued)

Simplified Procedure for Low Frequency:

1. Find the appropriate filter in the S -domain
2. Use the substitution

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

(the bilinear transform with $C = \frac{2}{T}$) to obtain the desired discrete-time filter.

Bilinear Transform (continued)

Example:

The second-order, lowpass, Butterworth filter with cutoff (3-dB) angular frequency ω_c has a transfer function given by

$$H_a(s) = \frac{\omega_c^2}{s^2 + \sqrt{2}\omega_c s + \omega_c^2}$$

Find a discrete-time, second-order, lowpass, Butterworth filter with cutoff (3-dB frequency 22.05kHz, to be used in a system whose sampling frequency is 88.2kHz.

Solution:

$$\text{First, } \theta_c = 2\pi f_c / f_S = \pi/2$$

$$\text{Then warp frequency: } \Omega_c = \tan(\theta_c/2) = 1.$$

So the analog filter is

$$\begin{aligned} H_a(s) &= \frac{\Omega_c^2}{s^2 + \sqrt{2}\Omega_c s + \Omega_c^2} \\ &= \frac{1}{s^2 + \sqrt{2}s + 1} \end{aligned}$$

Solution (continued)

The required digital filter then is

$$\begin{aligned}
 H(z) &= H_a \left(\frac{z-1}{z+1} \right) \\
 &= \frac{1}{\left(\frac{1-z^{-1}}{1+z^{-1}} \right)^2 + \sqrt{2} \left(\frac{1-z^{-1}}{1+z^{-1}} \right) + 1} \\
 &= \frac{(1+z^{-1})^2}{(1-z^{-1})^2 + \sqrt{2}(1-z^{-2}) + (1+z^{-1})^2} \\
 &= \frac{1+2z^{-1}+z^{-2}}{2+\sqrt{2}+(2-\sqrt{2})z^{-2}} \\
 &= \frac{1}{2+\sqrt{2}} \times \frac{1+2z^{-1}+z^{-2}}{1+\left(\frac{2-\sqrt{2}}{2+\sqrt{2}}\right)z^{-2}} \\
 &= 0.2928932 \frac{1+2z^{-1}+z^{-2}}{1+0.1715729z^{-2}}
 \end{aligned}$$

Difference equations:

$$t_n = u_n + 2u_{n-1} + u_{n-2} - 0.1715729 t_{n-2}$$

$$y_n = 0.2928932 t_n$$