Chapter 16 DESIGN OF RECURSIVE FILTERS USING OPTIMIZATION METHODS

16.7 Design of Recursive Filters

16.8 Design of Recursive Delay Equalizers

Copyright © 2018 Andreas Antoniou
Victoria, BC, Canada
Email: aantoniou@ieee.org

July 10, 2018

Introduction

The actual design of recursive filters by optimization involves several tasks as follows:

- Formulate the objective function.

Introduction

The actual design of recursive filters by optimization involves several tasks as follows:

- Formulate the objective function.
- Derive the partial derivatives of the objective function.

Introduction

The actual design of recursive filters by optimization involves several tasks as follows:

- Formulate the objective function.
- Derive the partial derivatives of the objective function.
- Choose a suitable algorithm.

Introduction

The actual design of recursive filters by optimization involves several tasks as follows:

- Formulate the objective function.
- Derive the partial derivatives of the objective function.
- Choose a suitable algorithm.
- Design a filter that would satisfy the required specifications with the minimum filter order.

Introduction

The actual design of recursive filters by optimization involves several tasks as follows:

- Formulate the objective function.
- Derive the partial derivatives of the objective function.
- Choose a suitable algorithm.
- Design a filter that would satisfy the required specifications with the minimum filter order.
- Check the stability of the filter designed.

Formulation of Objective Function

- As was shown in a previous presentation, an arbitrary recursive filter can be represented by the transfer function

$$H(z) = H_0 \prod_{j=1}^{J} \frac{a_{0j} + a_{1j}z + z^2}{b_{0j} + b_{1j}z + z^2}$$

where a_{ij} and b_{ij} are real coefficients, $J = N/2$, and H_0 is a positive multiplier constant.

Formulation of Objective Function

- As was shown in a previous presentation, an arbitrary recursive filter can be represented by the transfer function

$$H(z) = H_0 \prod_{j=1}^{J} \frac{a_{0j} + a_{1j}z + z^2}{b_{0j} + b_{1j}z + z^2}$$

  where a_{ij} and b_{ij} are real coefficients, $J = N/2$, and H_0 is a positive multiplier constant.

- Recall that an odd-order transfer function can be obtained by letting $a_{0j} = b_{0j} = 0$ for one value of j.

Formulation of Objective Function *Cont'd*

- The amplitude response of a recursive filter is given by

$$M(\mathbf{x}, \omega) = |H(e^{j\omega T})|$$

where ω is the frequency and

$$\mathbf{x} = [a_{01}\ a_{11}\ b_{01}\ b_{11}\ \cdots\ b_{1J}\ H_0]^T$$

is a column vector with $4J + 1$ elements.

Formulation of Objective Function *Cont'd*

- An approximation error can be constructed as

$$e_i(\mathbf{x}) = M(\mathbf{x}, \omega_i) - M_0(\omega_i)$$

where $M(\mathbf{x}, \omega_i)$ and $M_0(\omega_i)$ are the actual and desired amplitude responses, respectively, and $\{\omega_i\colon\ i = 1, 2, \ldots, K\}$ is a dense set of frequencies which are distributed in some way over the passband(s) and stopband(s) of the filter.

Formulation of Objective Function *Cont'd*

- An approximation error can be constructed as

$$e_i(\mathbf{x}) = M(\mathbf{x}, \omega_i) - M_0(\omega_i)$$

  where $M(\mathbf{x}, \omega_i)$ and $M_0(\omega_i)$ are the actual and desired amplitude responses, respectively, and $\{\omega_i\colon\ i = 1, 2, \ldots, K\}$ is a dense set of frequencies which are distributed in some way over the passband(s) and stopband(s) of the filter.

- Hence an objective function can be constructed as

$$\Psi(\mathbf{x}) = L_p(\mathbf{x}) = ||\mathbf{E}(\mathbf{x})||_p = \left[\sum_{i=1}^{K} |e_i(\mathbf{x})|^p\right]^{1/p}$$

  where $p = 2$ for a least-squares design and $p = \infty$ for a minimax design.

Formulation of Objective Function *Cont'd*

- Straightforward analysis gives the amplitude response as

$$M(\mathbf{x},\omega) = H_0 \prod_{j=1}^{J} \frac{N_j(\omega)}{D_j(\omega)}$$

where

$$N_j(\omega) = [1 + a_{0j}^2 + a_{1j}^2 + 2a_{1j}(1 + a_{0j})\cos\omega T + 2a_{0j}\cos 2\omega T]^{\frac{1}{2}}$$

and

$$D_j(\omega) = [1 + b_{0j}^2 + b_{1j}^2 + 2b_{1j}(1 + b_{0j})\cos\omega T + 2b_{0j}\cos 2\omega T]^{\frac{1}{2}}$$

for $j = 1, 2, \ldots, J$.

Formulation of Objective Function *Cont'd*

- The objective function obtained can be minimized by using a great variety of optimization algorithms.

Formulation of Objective Function *Cont'd*

- The objective function obtained can be minimized by using a great variety of optimization algorithms.
- As stated in the previous presentation, quasi-Newton algorithms are quite efficient as well as robust and, in addition, they do not require the second derivatives of the objective function.

Formulation of Objective Function *Cont'd*

- The objective function obtained can be minimized by using a great variety of optimization algorithms.
- As stated in the previous presentation, quasi-Newton algorithms are quite efficient as well as robust and, in addition, they do not require the second derivatives of the objective function.
- Hence, if there are no constraints other than the stability constraint, these algorithms can be used to design recursive filters that would satisfy arbitrary amplitude-response specifications.

Gradient of Objective Function

- For an objective function defined in terms of the L_p norm, the *gradient* of the objective function can be deduced as

$$\nabla \Psi_k(\mathbf{x}) = \left\{ \sum_{i=1}^{K} \left[\frac{|e_i(\mathbf{x})|}{\widehat{E}(\mathbf{x})} \right]^p \right\}^{(1/p)-1} \sum_{i=1}^{K} \left[\frac{|e_i(\mathbf{x})|}{\widehat{E}(\mathbf{x})} \right]^{p-1} \boldsymbol{\nabla} |e_i(\mathbf{x})|$$

where

$$\boldsymbol{\nabla} |e_i(\mathbf{x})| = [\mathrm{sgn}\ e_i(\mathbf{x})] \boldsymbol{\nabla} e_i(\mathbf{x})$$

with

$$\mathrm{sgn}[e_i(\mathbf{x})] = \begin{cases} 1 & \text{if } e_i(\mathbf{x}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Gradient of Objective Function *Cont'd*

- The elements of $\nabla e_i(\mathbf{x})$ can be obtained by differentiating the error function:

$$\frac{\partial e_i(\mathbf{x})}{\partial a_{0l}} = \frac{a_{0l} + a_{1l}\cos\omega_i T + \cos 2\omega_i T}{[N_l(\omega_i)]^2} \cdot M(\mathbf{x},\omega_i)$$

$$\frac{\partial e_i(\mathbf{x})}{\partial a_{1l}} = \frac{a_{1l} + (1 + a_{0l})\cos\omega_i T}{[N_l(\omega_i)]^2} \cdot M(\mathbf{x},\omega_i)$$

$$\frac{\partial e_i(\mathbf{x})}{\partial b_{0l}} = -\frac{b_{0l} + b_{1l}\cos\omega_i T + \cos 2\omega_i T}{[D_l(\omega_i)]^2} \cdot M(\mathbf{x},\omega_i)$$

$$\frac{\partial e_i(\mathbf{x})}{\partial b_{1l}} = -\frac{b_{1l} + (1 + b_{0l})\cos\omega_i T}{[D_l(\omega_i)]^2} \cdot M(\mathbf{x},\omega_i)$$

$$\frac{\partial e_i(\mathbf{x})}{\partial H_0} = \frac{1}{H_0} \cdot M(\mathbf{x},\omega_i)$$

for $l = 1, 2, \ldots, J$ and $i = 1, 2, \ldots, K$

Minimum Filter Order

- A problem associated with the design of recursive filters in general is that there are no known methods for predicting the minimum filter order that would satisfy prescribed amplitude and/or phase response specifications.

Minimum Filter Order

- A problem associated with the design of recursive filters in general is that there are no known methods for predicting the minimum filter order that would satisfy prescribed amplitude and/or phase response specifications.
- However, satisfactory results can often be achieved on a *cut-and-try* basis by designing filters of increasing orders until the error is sufficiently small to satisfy the specifications.

Prescribed Specifications

- Minimax optimization algorithms in general tend to *equalize* the amplitude of the error in the various bands such that

$$|e(\breve{\mathbf{x}}, \omega)| = |M(\breve{\mathbf{x}}, \omega) - M_0(\omega)| \leq \delta$$

where δ is the maximum value of $|e(\breve{\mathbf{x}}, \omega)|$.

Prescribed Specifications

- Minimax optimization algorithms in general tend to *equalize* the amplitude of the error in the various bands such that

$$|e(\breve{\mathbf{x}}, \omega)| = |M(\breve{\mathbf{x}}, \omega) - M_0(\omega)| \leq \delta$$

  where δ is the maximum value of $|e(\breve{\mathbf{x}}, \omega)|$.

- In practice, the prescribed maximum passband and minimum stopband attenuations or, equivalently, the *required maximum values of the passband and stopband errors vary wildly* from band to band and from one application to the next.

Prescribed Specifications

- Minimax optimization algorithms in general tend to *equalize* the amplitude of the error in the various bands such that

$$|e(\breve{\mathbf{x}}, \omega)| = |M(\breve{\mathbf{x}}, \omega) - M_0(\omega)| \le \delta$$

  where δ is the maximum value of $|e(\breve{\mathbf{x}}, \omega)|$.

- In practice, the prescribed maximum passband and minimum stopband attenuations or, equivalently, the *required maximum values of the passband and stopband errors vary wildly* from band to band and from one application to the next.

- In order to achieve desired specifications, *we need to be able to control the relative magnitudes of the maximum passband and stopband errors*, and this is achieved through the use of *weighting*, as in the design of nonrecursive filters described in Chap. 15.

Prescribed Specifications *Cont'd*

- Weighting essentially involves constructing a modified error function of the form

$$\begin{aligned} e_i'(\mathbf{x}) &= w_m e_i(\mathbf{x}) \\ &= w_m[M(\mathbf{x}, \omega_i) - M_0(\omega_i)] \end{aligned}$$

where w_m for $m = 1, 2, \ldots, M$ are positive weighting constants which are used to emphasize or deemphasize the approximation error in selected passbands or stopbands so as to decrease or increase its magnitude in those bands.

Prescribed Specifications *Cont'd*

- Weighting essentially involves constructing a modified error function of the form

$$\begin{aligned} e_i'(\mathbf{x}) &= w_m e_i(\mathbf{x}) \\ &= w_m[M(\mathbf{x},\omega_i) - M_0(\omega_i)] \end{aligned}$$

  where w_m for $m = 1, 2, \ldots, M$ are positive weighting constants which are used to emphasize or deemphasize the approximation error in selected passbands or stopbands so as to decrease or increase its magnitude in those bands.

- Typically, w_m is assigned a value greater or less than one to decrease or increase the magnitude of the approximation error in a given band.

Prescribed Specifications *Cont'd*

- If weighting is employed, the optimization algorithm would tend to equalize the weighted error $e'_i(\mathbf{x})$ such that

$$|e'_i(\mathbf{x})| = w_m|[M(\mathbf{x}, \omega_i) - M_0(\omega_i)]| \leq \delta$$

Prescribed Specifications *Cont'd*

- If weighting is employed, the optimization algorithm would tend to equalize the weighted error $e'_i(\mathbf{x})$ such that

$$|e'_i(\mathbf{x})| = w_m|[M(\mathbf{x},\omega_i) - M_0(\omega_i)]| \le \delta$$

- Therefore, the actual error would satisfy the inequality

$$|[M(\mathbf{x},\omega_i) - M_0(\omega_i)]| \le \frac{\delta}{w_m}$$

Prescribed Specifications *Cont'd*

...

$$|[M(\mathbf{x}, \omega_i) - M_0(\omega_i)]| \leq \frac{\delta}{w_m}$$

- If the weighting constants for the various bands are assumed to be m_1, m_2, ..., m_M, then at convergence the maximum band errors would be

$$\delta_1 = \frac{\delta}{w_1}, \quad \delta_2 = \frac{\delta}{w_2}, \quad \ldots, \quad \delta_M = \frac{\delta}{w_M}$$

  since the optimization algorithm would tend to equalize the weighted errors in the various bands.

Prescribed Specifications *Cont'd*

...

$$\delta_1 = \frac{\delta}{w_1},\ \delta_2 = \frac{\delta}{w_2},\ \ldots,\ \delta_M = \frac{\delta}{w_M}$$

- Thus if $\delta_1, \delta_2, \ldots, \delta_M$ are the desired band errors where

$$\delta_i = \frac{10^{0.05A_{p_i}-1}}{10^{0.05A_{p_i}}+1}$$

for a passband with a ripple A_{p_i} dB and

$$\delta_j = 10^{-0.05A_{a_j}}$$

for a stopband with a minimum attenuation A_{a_j} dB, the required weighting constants should be

$$w_1 = \frac{\delta}{\delta_1},\ w_2 = \frac{\delta}{\delta_2},\ \ldots,\ w_k = \frac{\delta}{\delta_k},\ \ldots,\ w_M = \frac{\delta}{\delta_M}$$

Prescribed Specifications *Cont'd*

. . .

$$w_1 = \frac{\delta}{\delta_1},\ w_2 = \frac{\delta}{\delta_2},\ \ldots,\ w_k = \frac{\delta}{\delta_k},\ \ldots,\ w_M = \frac{\delta}{\delta_M}$$

- Assuming that the weighting constant for the kth band is unity, i.e., $w_k = 1$, then $\delta = \delta_k$ and

$$w_1 = \frac{\delta_k}{\delta_1},\ w_2 = \frac{\delta_k}{\delta_2},\ \ \ldots,\ w_M = \frac{\delta_k}{\delta_M}$$

Prescribed Specifications *Cont'd*

...

$$w_1 = \frac{\delta}{\delta_1},\ w_2 = \frac{\delta}{\delta_2},\ \ldots,\ w_k = \frac{\delta}{\delta_k},\ \ldots,\ w_M = \frac{\delta}{\delta_M}$$

- Assuming that the weighting constant for the kth band is unity, i.e., $w_k = 1$, then $\delta = \delta_k$ and

$$w_1 = \frac{\delta_k}{\delta_1},\ w_2 = \frac{\delta_k}{\delta_2},\ \ldots,\ w_M = \frac{\delta_k}{\delta_M}$$

- Usually, it is convenient to assign a weighting constant of unity to a passband.

Prescribed Specifications *Cont'd*

Arbitrary filter specifications can be achieved as follows:

1. Choose suitable weighting constants on the basis of the prescribed specifications as described.

Prescribed Specifications *Cont'd*

Arbitrary filter specifications can be achieved as follows:

1. Choose suitable weighting constants on the basis of the prescribed specifications as described.
2. Design filters for increasing orders until a filter is found that would satisfy the required specifications for band k.

   The filter obtained would *usually* satisfy the specifications in all the other bands as well.

Prescribed Specifications *Cont'd*

Arbitrary filter specifications can be achieved as follows:

1. Choose suitable weighting constants on the basis of the prescribed specifications as described.
2. Design filters for increasing orders until a filter is found that would satisfy the required specifications for band k.

   The filter obtained would *usually* satisfy the specifications in all the other bands as well.
3. If the filter *does not satisfy* the specifications in *all the bands*, increase the filter order, say, by two, and try again.

Prescribed Specifications *Cont'd*

Arbitrary filter specifications can be achieved as follows:

1. Choose suitable weighting constants on the basis of the prescribed specifications as described.
2. Design filters for increasing orders until a filter is found that would satisfy the required specifications for band k.

   The filter obtained would *usually* satisfy the specifications in all the other bands as well.
3. If the filter *does not satisfy* the specifications in *all the bands*, increase the filter order, say, by two, and try again.

   Evidently, this is a *trial and error* method and it could entail a considerable amount of computation.

Stability

- Least-squares or minimax algorithms often yield discrete-time transfer functions with poles outside the unit circle $|z| = 1$, and such transfer functions represent *unstable filters*.

Stability

- Least-squares or minimax algorithms often yield discrete-time transfer functions with poles outside the unit circle $|z| = 1$, and such transfer functions represent *unstable filters*.
- Fortunately, it is possible to eliminate this problem through a *stabilization technique* that has been known for a number of years.

Stability *Cont'd*

- Let us assume that an optimization algorithm has produced a discrete-time transfer function

$$H(z) = H_0 \frac{N(z)}{D(z)} = H_0 \frac{N(z)}{D'(z) \prod_{i=1}^{k} (z - \tilde{p}_i)}$$

with *k poles $\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_k$ that lie outside the unit circle*.

Stability *Cont'd*

- Let us assume that an optimization algorithm has produced a discrete-time transfer function

$$H(z) = H_0 \frac{N(z)}{D(z)} = H_0 \frac{N(z)}{D'(z) \prod_{i=1}^{k} (z - \tilde{p}_i)}$$

  with *k poles $\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_k$ that lie outside the unit circle*.

- A *stable* transfer function that yields the same amplitude response can be obtained as

$$H'(z) = H'_0 \frac{N(z)}{D'(z) \prod_{i=1}^{k} (z - 1/\tilde{p}_i)} = H'_0 \frac{N(z)}{1 + \sum_{i=1}^{N} b'_i z^i}$$

  where

$$H'_0 = H_0 \frac{1}{\prod_{i=1}^{k} \tilde{p}_i}$$

Stability *Cont'd*

- Let us assume that an optimization algorithm has produced a discrete-time transfer function

$$H(z) = H_0 \frac{N(z)}{D(z)} = H_0 \frac{N(z)}{D'(z) \prod_{i=1}^{k} (z - \tilde{p}_i)}$$

with *k poles $\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_k$ that lie outside the unit circle*.

- A *stable* transfer function that yields the same amplitude response can be obtained as

$$H'(z) = H_0' \frac{N(z)}{D'(z) \prod_{i=1}^{k} (z - 1/\tilde{p}_i)} = H_0' \frac{N(z)}{1 + \sum_{i=1}^{N} b_i' z^i}$$

where

$$H_0' = H_0 \frac{1}{\prod_{i=1}^{k} \tilde{p}_i}$$

See textbook for details.

Example

- Through the application of the singular-value decomposition, the problem of designing *two-dimensional digital filters* can be broken down into a problem of designing *a set of one-dimensional digital filters*.

Example

- Through the application of the singular-value decomposition, the problem of designing *two-dimensional digital filters* can be broken down into a problem of designing *a set of one-dimensional digital filters*.
- The amplitude responses of the one-dimensional filters so obtained turn out to be *quite irregular* and, consequently, their design can be accomplished only through the use of optimization methods.

Example

- Through the application of the singular-value decomposition, the problem of designing *two-dimensional digital filters* can be broken down into a problem of designing *a set of one-dimensional digital filters*.
- The amplitude responses of the one-dimensional filters so obtained turn out to be *quite irregular* and, consequently, their design can be accomplished only through the use of optimization methods.
- The amplitude response of such a filter is specified at 21 frequency points as shown in the table in the next slide.

Example

- Through the application of the singular-value decomposition, the problem of designing *two-dimensional digital filters* can be broken down into a problem of designing *a set of one-dimensional digital filters*.
- The amplitude responses of the one-dimensional filters so obtained turn out to be *quite irregular* and, consequently, their design can be accomplished only through the use of optimization methods.
- The amplitude response of such a filter is specified at 21 frequency points as shown in the table in the next slide.
- Obtain an eighth-order design using the least-pth minimax algorithm.

Example *Cont'd*

Amplitude Response

ω	Gain	ω	Gain	ω	Gain
0.00	1.0770	0.35	0.0304	0.70	0.7950
0.05	0.9863	0.40	0.1665	0.75	0.7950
0.10	0.9866	0.45	0.4402	0.80	0.7950
0.15	0.8428	0.50	0.6231	0.85	0.7950
0.20	0.8436	0.55	0.7471	0.90	0.7950
0.25	0.6466	0.60	0.7950	0.95	0.7950
0.30	0.3955	0.65	0.7950	1.00	0.7950

Example *Cont'd*

Solution

- A design was obtained by assuming a transfer function of the form

$$H(z) = H_0 \prod_{j=1}^{4} \frac{a_{0j} + a_{1j}z + z^2}{b_{0j} + b_{1j}z + z^2}$$

Example *Cont'd*

Solution

- A design was obtained by assuming a transfer function of the form

$$H(z) = H_0 \prod_{j=1}^{4} \frac{a_{0j} + a_{1j}z + z^2}{b_{0j} + b_{1j}z + z^2}$$

- Typically the number of sample points required to achieve good precision depends critically on the selectivity of the filter.

Example *Cont'd*

Solution

- A design was obtained by assuming a transfer function of the form

$$H(z) = H_0 \prod_{j=1}^{4} \frac{a_{0j} + a_{1j}z + z^2}{b_{0j} + b_{1j}z + z^2}$$

- Typically the number of sample points required to achieve good precision depends critically on the selectivity of the filter.

  The problem was solved by using the least-pth algorithm along with the variable sampling technique of Sec. 16.6 and good results were achieved with 35 actual sample points.

Example *Cont'd*

Solution

- A design was obtained by assuming a transfer function of the form

$$H(z) = H_0 \prod_{j=1}^{4} \frac{a_{0j} + a_{1j}z + z^2}{b_{0j} + b_{1j}z + z^2}$$

- Typically the number of sample points required to achieve good precision depends critically on the selectivity of the filter.

  The problem was solved by using the least-pth algorithm along with the variable sampling technique of Sec. 16.6 and good results were achieved with 35 actual sample points.

  If fixed uniformly-spaced sample frequencies were used, 85 to 170 sample points would be required to achieve similar results.

Example *Cont'd*

- Since the amplitude response is specified at only 21 points, it is necessary to apply interpolation to the numerical values supplied in order to obtain the values of the idealized amplitude response with respect to a dense set of frequencies.

  Quadratic or cubic interpolation can be used for the purpose (see textbook).

Example *Cont'd*

- Since the amplitude response is specified at only 21 points, it is necessary to apply interpolation to the numerical values supplied in order to obtain the values of the idealized amplitude response with respect to a dense set of frequencies.

  Quadratic or cubic interpolation can be used for the purpose (see textbook).

- Any initial point can be used for this problem.

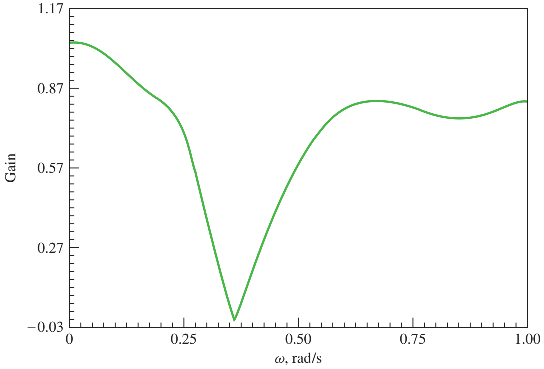
  A good practice is to start with a *stable* transfer function because, usually, stability tends to be preserved during the optimization.

  The initial point used for this design was

$$\mathbf{x} = [1\ 1\ 0.75\ 1\ 1\ 1\ 0.75\ 1\ 1\ -1\ 0.75\ -1\ 1\ -1\ 0.75\ -1\ \ 1]^T$$

Example *Cont'd*

- The amplitude response achieved is shown in the figure.



Example *Cont'd*

- The progress of the algorithm is illustrated in the table shown.

Progress of Algorithm

k	p	$\Psi(\mathbf{x})$
1	2	7.106816E-2
2	4	3.726389E-2
3	8	3.329217E-2
4	16	3.757264E-2
5	32	3.472619E-2
6	64	3.359927E-2
7	128	3.304717E-2

Design of Recursive Equalizers

- In many applications, recursive digital filters are required with a flat delay characteristic or, equivalently, a linear phase response with respect to passbands.

Design of Recursive Equalizers

- In many applications, recursive digital filters are required with a flat delay characteristic or, equivalently, a linear phase response with respect to passbands.
- Unfortunately, most of the available methods for the design of recursive digital filters, for example, the bilinear transformation method, tend to yield filters that have a *nonlinear phase response*.

Design of Recursive Equalizers

- In many applications, recursive digital filters are required with a flat delay characteristic or, equivalently, a linear phase response with respect to passbands.
- Unfortunately, most of the available methods for the design of recursive digital filters, for example, the bilinear transformation method, tend to yield filters that have a *nonlinear phase response*.
- One way to overcome this problem is to design the filter such that the required *amplitude response specifications* and a *linear phase response* with respect to the passbands are achieved simultaneously.

Design of Recursive Equalizers

- In many applications, recursive digital filters are required with a flat delay characteristic or, equivalently, a linear phase response with respect to passbands.

- Unfortunately, most of the available methods for the design of recursive digital filters, for example, the bilinear transformation method, tend to yield filters that have a *nonlinear phase response*.

- One way to overcome this problem is to design the filter such that the required *amplitude response specifications* and a *linear phase response* with respect to the passbands are achieved simultaneously.

  This can be done with optimization but, unfortunately, *constrained optimization* is required in order to achieve a stable design, which tends to be more complicated than unconstrained optimization and, furthermore, it requires a considerable amount of computation.

Design of Recursive Equalizers *Cont'd*

- A simpler approach is first to design a filter that would satisfy the amplitude response specifications ignoring the group delay and then design a *recursive delay equalizer* which can be used in cascade with the filter to compensate for variations in the group delay of the filter.

Design of Recursive Equalizers *Cont'd*

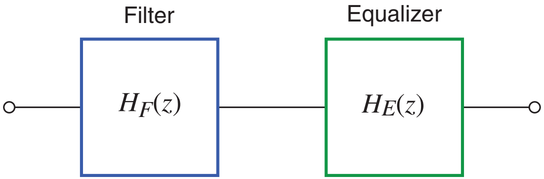
- A simpler approach is first to design a filter that would satisfy the amplitude response specifications ignoring the group delay and then design a *recursive delay equalizer* which can be used in cascade with the filter to compensate for variations in the group delay of the filter.
- The recursive filter can be designed by using the *approach of Chap. 13* whereas the equalizer can be designed by using *unconstrained optimization*.

Design of Recursive Equalizers *Cont'd*

- In a cascade arrangement like the one shown, we have an overall transfer function

$$H_{FE}(z) = H_F(z) \cdot H_E(z)$$

where $H_F(z)$ and $H_E(z)$ are the transfer functions of the filter and equalizer, respectively.



Design of Recursive Equalizers *Cont'd*

...

$$H_{FE}(z) = H_F(z) \cdot H_E(z)$$

- The frequency response of the cascade arrangement is given by

  $$M_{FE}(\omega)e^{j\theta(\omega)} = M_F(\omega)e^{j\theta_F(\omega)} \cdot M_E(\omega)e^{j\theta_E(\omega)}$$

  Hence, the amplitude response, phase response, and group delay characteristic of the arrangement are given by

  $$\begin{aligned} M_{FE}(\omega) &= M_F(\omega) \cdot M_E(\omega) \\ \theta_{FE}(\omega) &= \theta_F(\omega) + \theta_E(\omega) \\ \tau_{FE}(\omega) &= \tau_F(\omega) + \tau_E(\omega) \end{aligned}$$

  respectively.

Design of Recursive Equalizers *Cont'd*

. . .

$$\begin{aligned} M_{FE}(\omega) &= M_F(\omega) \cdot M_E(\omega) \\ \theta_{FE}(\omega) &= \theta_F(\omega) + \theta_E(\omega) \\ \tau_{FE}(\omega) &= \tau_F(\omega) + \tau_E(\omega) \end{aligned}$$

An equalized filter can be designed as follows:

1. Design a recursive filter such that the amplitude response $M_F(\omega)$ satisfies the required specifications.

Design of Recursive Equalizers *Cont'd*

...

$$\begin{aligned} M_{FE}(\omega) &= M_F(\omega) \cdot M_E(\omega) \\ \theta_{FE}(\omega) &= \theta_F(\omega) + \theta_E(\omega) \\ \tau_{FE}(\omega) &= \tau_F(\omega) + \tau_E(\omega) \end{aligned}$$

An equalized filter can be designed as follows:

1. Design a recursive filter such that the amplitude response $M_F(\omega)$ satisfies the required specifications.

2. Find the group delay characteristic of the recursive filter.

Design of Recursive Equalizers *Cont'd*

...

$$\begin{aligned} M_{FE}(\omega) &= M_F(\omega) \cdot M_E(\omega) \\ \theta_{FE}(\omega) &= \theta_F(\omega) + \theta_E(\omega) \\ \tau_{FE}(\omega) &= \tau_F(\omega) + \tau_E(\omega) \end{aligned}$$

An equalized filter can be designed as follows:

1. Design a recursive filter such that the amplitude response $M_F(\omega)$ satisfies the required specifications.

2. Find the group delay characteristic of the recursive filter.

3. Design an equalizer such that $M_E(\omega) = 1$ at all frequencies and

   $$\tau_{FE}(\omega) = \tau_F(\omega) + \tau_E(\omega) \approx \tau_0$$

   over the passband(s), where τ_0 is a constant.

Design of Recursive Equalizers *Cont'd*

...

$$\begin{aligned} M_{FE}(\omega) &= M_F(\omega) \cdot M_E(\omega) \\ \theta_{FE}(\omega) &= \theta_F(\omega) + \theta_E(\omega) \\ \tau_{FE}(\omega) &= \tau_F(\omega) + \tau_E(\omega) \end{aligned}$$

An equalized filter can be designed as follows:

1. Design a recursive filter such that the amplitude response $M_F(\omega)$ satisfies the required specifications.

2. Find the group delay characteristic of the recursive filter.

3. Design an equalizer such that $M_E(\omega) = 1$ at all frequencies and

$$\tau_{FE}(\omega) = \tau_F(\omega) + \tau_E(\omega) \approx \tau_0$$

   over the passband(s), where τ_0 is a constant.

   An equalizer is essentially a so-called *allpass* filter.

Design of Recursive Equalizers *Cont'd*

- An allpass filter can be obtained by using a transfer function of the form

$$H_E(z) = \prod_{j=1}^{M} \frac{1 + c_{1j}z + c_{0j}z^2}{c_{0j} + c_{1j}z + z^2}$$

Design of Recursive Equalizers *Cont'd*

- An allpass filter can be obtained by using a transfer function of the form

$$H_E(z) = \prod_{j=1}^{M} \frac{1 + c_{1j}z + c_{0j}z^2}{c_{0j} + c_{1j}z + z^2}$$

Note that the numerator coefficients are the same as the denominator coefficients except that *they appear in the reverse order*.

Design of Recursive Equalizers *Cont'd*

- The square of the gain of an allpass filter can be deduced by multiplying the frequency response of the filter by its complex conjugate, i.e.,

$$\begin{aligned}
[M_E(\omega)]^2 &= H_E(z)\cdot H_E(z^{-1})\Big|_{z=e^{j\omega T}} \\
&= \prod_{j=1}^{M} \frac{1+c_{1j}z+c_{0j}z^2}{c_{0j}+c_{1j}z+z^2}\cdot\frac{1+c_{1j}z^{-1}+c_{0j}z^{-2}}{c_{0j}+c_{1j}z^{-1}+z^{-2}}\Bigg|_{z=e^{j\omega T}} \\
&= \prod_{j=1}^{M} \frac{1+c_{1j}z+c_{0j}z^2}{c_{0j}+c_{1j}z+z^2}\cdot\frac{z^2+c_{1j}z+c_{0j}}{c_{0j}z^2+c_{1j}z+1}\Bigg|_{z=e^{j\omega T}} \\
&= 1
\end{aligned}$$

Design of Recursive Equalizers *Cont'd*

- The square of the gain of an allpass filter can be deduced by multiplying the frequency response of the filter by its complex conjugate, i.e.,

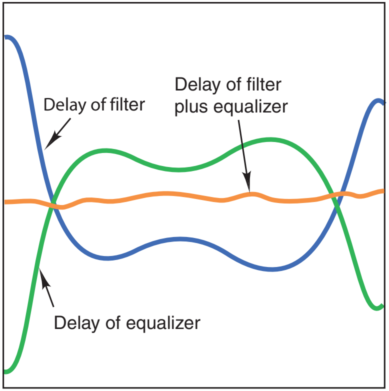
$$\begin{aligned}[M_E(\omega)]^2 &= H_E(z)\cdot H_E(z^{-1})\Big|_{z=e^{j\omega T}}\\ &= \prod_{j=1}^{M}\frac{1+c_{1j}z+c_{0j}z^2}{c_{0j}+c_{1j}z+z^2}\cdot\frac{1+c_{1j}z^{-1}+c_{0j}z^{-2}}{c_{0j}+c_{1j}z^{-1}+z^{-2}}\Bigg|_{z=e^{j\omega T}}\\ &= \prod_{j=1}^{M}\frac{1+c_{1j}z+c_{0j}z^2}{c_{0j}+c_{1j}z+z^2}\cdot\frac{z^2+c_{1j}z+c_{0j}}{c_{0j}z^2+c_{1j}z+1}\Bigg|_{z=e^{j\omega T}}\\ &= 1\end{aligned}$$

Therefore, $M_E(\omega)=1$ *for all frequencies*.

Design of Recursive Equalizers *Cont'd*

. . .

$$\tau_{FE}(\omega) = \tau_F(\omega) + \tau_E(\omega) \approx \tau_0$$



Design of Recursive Equalizers *Cont'd*

. . .

$$\tau_{FE}(\omega) = \tau_F(\omega) + \tau_E(\omega) \approx \tau_0$$

- Next we need to construct an objective function.

  An error function can be formulated as

$$e_i(\mathbf{x}) = \tau_{FE}(\mathbf{x}, \omega_i) - \tau_0 = \tau_F(\omega) + \tau_E(\omega) - \tau_0$$

  where $\mathbf{x} = [c_{01} \;\; c_{11} \;\; c_{02} \;\; \cdots \;\; c_{1M} \;\; \tau_0]^T$ and $\{\omega_i : 1,\; 2,\; \ldots, K\}$ is a *dense set of frequencies in the passband(s)* of the filter.

Design of Recursive Equalizers *Cont'd*

. . .

$$\tau_{FE}(\omega) = \tau_F(\omega) + \tau_E(\omega) \approx \tau_0$$

- Next we need to construct an objective function.

  An error function can be formulated as

$$e_i(\mathbf{x}) = \tau_{FE}(\mathbf{x}, \omega_i) - \tau_0 = \tau_F(\omega) + \tau_E(\omega) - \tau_0$$

  where $\mathbf{x} = [c_{01}\ c_{11}\ c_{02}\ \cdots\ c_{1M}\ \tau_0]^T$ and $\{\omega_i : 1,\ 2,\ \ldots, K\}$ is a *dense set of frequencies in the passband(s)* of the filter.

- A minimax optimization problem can now be constructed as

$$\underset{\mathbf{x}}{\text{minimize}} \max_{1 \le i \le K} |e_i(\mathbf{x})|$$

Design of Recursive Equalizers *Cont'd*

...

$$e_i(\mathbf{x}) = \tau_{FE}(\mathbf{x}, \omega_i) - \tau_0 = \tau_F(\omega) + \tau_E(\omega) - \tau_0$$

$$\mathbf{x} = [c_{01}\ c_{11}\ c_{02}\ \cdots\ c_{1M}\ \tau_0]^T$$

- The value of the desired constant delay τ_0 is usually unknown and to provide additional flexibility it is assumed to be an *independent variable*.

Design of Recursive Equalizers *Cont'd*

. . .

$$e_i(\mathbf{x}) = \tau_{FE}(\mathbf{x}, \omega_i) - \tau_0 = \tau_F(\omega) + \tau_E(\omega) - \tau_0$$

$$\mathbf{x} = [c_{01}\ c_{11}\ c_{02}\ \cdots\ c_{1M}\ \tau_0]^T$$

- The value of the desired constant delay τ_0 is usually unknown and to provide additional flexibility it is assumed to be an *independent variable*.
- Formulas for the group delays of the filter and the equalizer, $\tau_F(\omega)$ and $\tau_E(\omega)$, as well as for the partial derivatives of the error function, $e_i(\mathbf{x})$, can be found in the textbook.

Design of Recursive Equalizers *Cont'd*

- As in the case of filters, the equalizer order that would provide sufficient equalization is not known in advance.

Design of Recursive Equalizers *Cont'd*

- As in the case of filters, the equalizer order that would provide sufficient equalization is not known in advance.

  Suffice it to say that the order of the equalizer can be *as high* as the order of the filter.

Design of Recursive Equalizers *Cont'd*

- As in the case of filters, the equalizer order that would provide sufficient equalization is not known in advance.

  Suffice it to say that the order of the equalizer can be *as high* as the order of the filter.

- It turns out that an attempt to design a high-order equalizer using a single minimax optimization can easily yield an *unstable design*.

Design of Recursive Equalizers *Cont'd*

- As in the case of filters, the equalizer order that would provide sufficient equalization is not known in advance.

  Suffice it to say that the order of the equalizer can be *as high* as the order of the filter.

- It turns out that an attempt to design a high-order equalizer using a single minimax optimization can easily yield an *unstable design*.

  Unfortunately, in the case of equalizers, stability cannot be restored using the available technique for recursive filters without causing the phase response to become nonlinear.

Design of Recursive Equalizers *Cont'd*

- As in the case of filters, the equalizer order that would provide sufficient equalization is not known in advance.

  Suffice it to say that the order of the equalizer can be *as high* as the order of the filter.

- It turns out that an attempt to design a high-order equalizer using a single minimax optimization can easily yield an *unstable design*.

  Unfortunately, in the case of equalizers, stability cannot be restored using the available technique for recursive filters without causing the phase response to become nonlinear.

- However, by using an appropriate *sequential optimization algorithm*, as detailed in Chap. 16, good quality, stable equalizers can be designed.

Design of Recursive Equalizers *Cont'd*

The sequential optimization algorithm is based on the following strategy:

1. Design a *1-section* second-order equalizer using several initial points that correspond to stable designs and select the best design.

Design of Recursive Equalizers *Cont'd*

The sequential optimization algorithm is based on the following strategy:

1. Design a *1-section* second-order equalizer using several initial points that correspond to stable designs and select the best design.
2. Design a *2-section* fourth-order equalizer using the 1-section design obtained in Step 1 for the initialization of the first section and initialize the second section using a point in the neighborhood of the solution for the 1-section equalizer.

Design of Recursive Equalizers *Cont'd*

The sequential optimization algorithm is based on the following strategy:

1. Design a *1-section* second-order equalizer using several initial points that correspond to stable designs and select the best design.
2. Design a *2-section* fourth-order equalizer using the 1-section design obtained in Step 1 for the initialization of the first section and initialize the second section using a point in the neighborhood of the solution for the 1-section equalizer.

   Repeat with three other initial points in the neighborhood of the solutions and then select the best stable design.

Design of Recursive Equalizers *Cont'd*

The sequential optimization algorithm is based on the following strategy:

1. Design a *1-section* second-order equalizer using several initial points that correspond to stable designs and select the best design.
2. Design a *2-section* fourth-order equalizer using the 1-section design obtained in Step 1 for the initialization of the first section and initialize the second section using a point in the neighborhood of the solution for the 1-section equalizer.

   Repeat with three other initial points in the neighborhood of the solutions and then select the best stable design.
3. Design a *3-section* sixth-order equalizer using the 2-section design obtained in Step 2 for the initialization of the first 2 sections and initialize the third section using a point in the same neighborhood.

Design of Recursive Equalizers *Cont'd*

The sequential optimization algorithm is based on the following strategy:

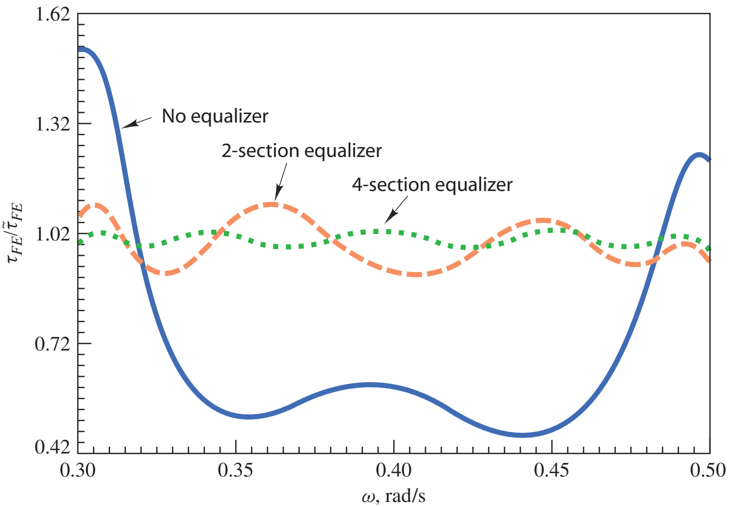
1. Design a *1-section* second-order equalizer using several initial points that correspond to stable designs and select the best design.
2. Design a *2-section* fourth-order equalizer using the 1-section design obtained in Step 1 for the initialization of the first section and initialize the second section using a point in the neighborhood of the solution for the 1-section equalizer.

   Repeat with three other initial points in the neighborhood of the solutions and then select the best stable design.
3. Design a *3-section* sixth-order equalizer using the 2-section design obtained in Step 2 for the initialization of the first 2 sections and initialize the third section using a point in the same neighborhood.
4. Proceeding as in Step 3, add new second-order sections until the required amount of equalization is achieved.

Example

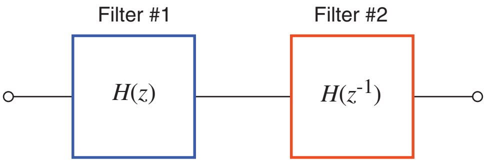
- The results obtained for a bandpass elliptic filter are shown in the next slide.

Example *Cont'd*



Zero-Phase Filters

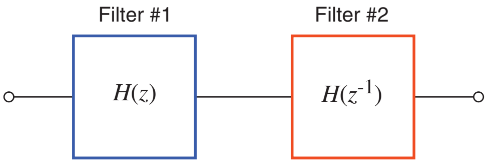
- Consider a cascade arrangement of two digital filters as shown with transfer functions $H(z)$ and $H(z^{-1})$.



Zero-Phase Filters

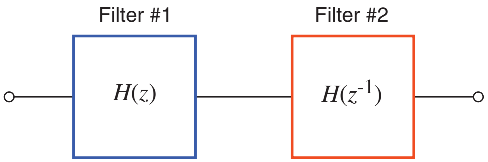
- Consider a cascade arrangement of two digital filters as shown with transfer functions $H(z)$ and $H(z^{-1})$.
- The frequency response of the cascade arrangement is given by

$$H_0(e^{j\omega T}) = H(e^{j\omega T}) \cdot H(e^{-j\omega T}) = |H(e^{j\omega T})|^2$$



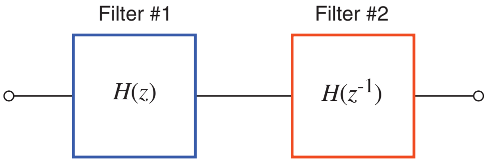
Zero-Phase Filters

- Consider a cascade arrangement of two digital filters as shown with transfer functions $H(z)$ and $H(z^{-1})$.
- The frequency response of the cascade arrangement is given by
$$H_0(e^{j\omega T}) = H(e^{j\omega T}) \cdot H(e^{-j\omega T}) = |H(e^{j\omega T})|^2$$
- Since the frequency response of the arrangement is real, the *phase response as well as the group delay are zero*.



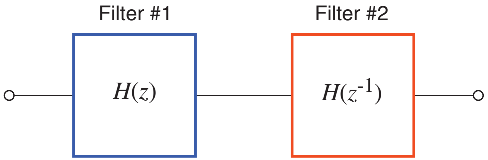
Zero-Phase Filters *Cont'd*

- If Filter #1 is a causal filter with an impulse response $h(nT)$, then Filter #2 can be shown to be a *noncausal* filter with an impulse response $h(-nT)$, i.e., the impulse response of the second filter is the mirror-image of that first filter with respect to the y axis.



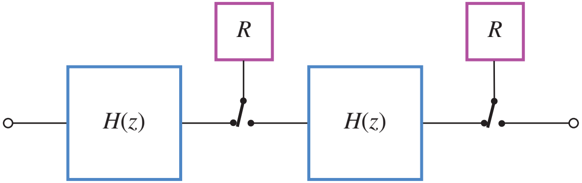
Zero-Phase Filters *Cont'd*

- If Filter #1 is a causal filter with an impulse response $h(nT)$, then Filter #2 can be shown to be a *noncausal* filter with an impulse response $h(-nT)$, i.e., the impulse response of the second filter is the mirror-image of that first filter with respect to the y axis.
- Obviously, such a design cannot be implemented in real time.



Zero-Phase Filters *Cont'd*

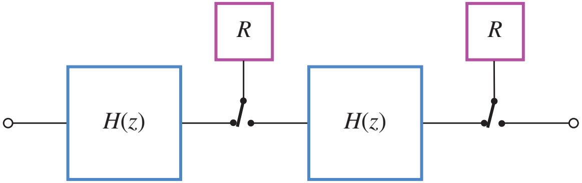
- However, a *nonreal-time implementation* is possible by using two copies of Filter #1 as shown in the figure.



Zero-Phase Filters *Cont'd*

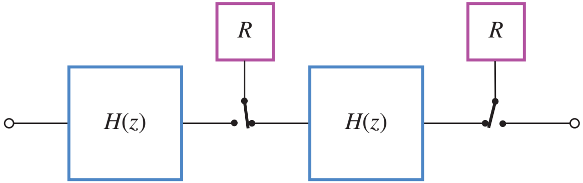
- However, a *nonreal-time implementation* is possible by using two copies of Filter #1 as shown in the figure.
- Devices R are simple *first-in-last-out* data registers.

  The first register simply saves the output of the first filter and feeds it backwards into the second filter whereas the second register reverses the output of the second filter.



Zero-Phase Filters *Cont'd*

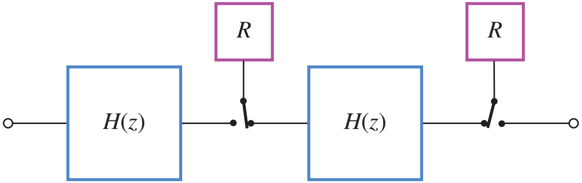
- Put another way, the first filter delays the signal by $\tau(\omega)$ s.



Zero-Phase Filters *Cont'd*

- Put another way, the first filter delays the signal by $\tau(\omega)$ s.

  The second filter delays the reversed version of the signal by $\tau(\omega)$ s or, equivalently, it delays the signal by $-\tau(\omega)$, i.e., it advances the signal by $\tau(\omega)$.

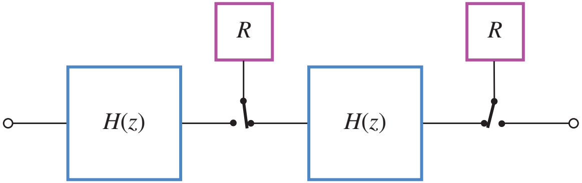


Zero-Phase Filters *Cont'd*

- Put another way, the first filter delays the signal by $\tau(\omega)$ s.

  The second filter delays the reversed version of the signal by $\tau(\omega)$ s or, equivalently, it delays the signal by $-\tau(\omega)$, i.e., it advances the signal by $\tau(\omega)$.

- Therefore, an overall delay of $\tau(\omega) - \tau(\omega) = 0$ s is achieved but the signal at the output of the second filter *is reversed*.



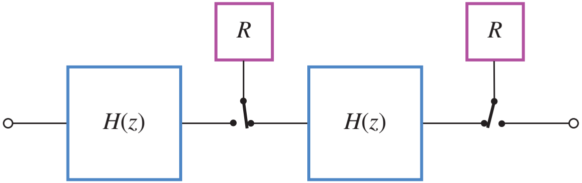
Zero-Phase Filters *Cont'd*

- Put another way, the first filter delays the signal by $\tau(\omega)$ s.

  The second filter delays the reversed version of the signal by $\tau(\omega)$ s or, equivalently, it delays the signal by $-\tau(\omega)$, i.e., it advances the signal by $\tau(\omega)$.

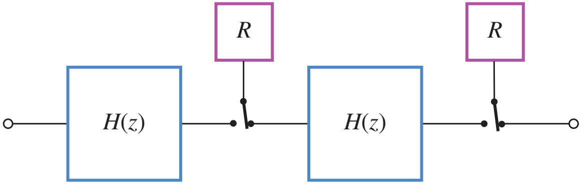
- Therefore, an overall delay of $\tau(\omega) - \tau(\omega) = 0$ s is achieved but the signal at the output of the second filter *is reversed*.

  Thus, another register is needed at the output!



Zero-Phase Filters *Cont'd*

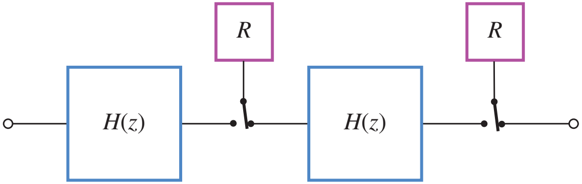
- Note that the overall maximum passband ripple and minimum stopband attenuation of the cascade arrangement would be *twice* the corresponding parameters of the individual filters.



Zero-Phase Filters *Cont'd*

- Note that the overall maximum passband ripple and minimum stopband attenuation of the cascade arrangement would be *twice* the corresponding parameters of the individual filters.

  In other words, if the specifications call for an overall passband ripple and a minimum stopband attenuation of A_p and A_a, respectively, then the corresponding specifications for each filter should be $A_p/2$ and $A_a/2$, respectively.

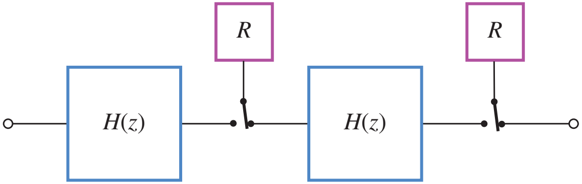


Zero-Phase Filters *Cont'd*

- Note that the overall maximum passband ripple and minimum stopband attenuation of the cascade arrangement would be *twice* the corresponding parameters of the individual filters.

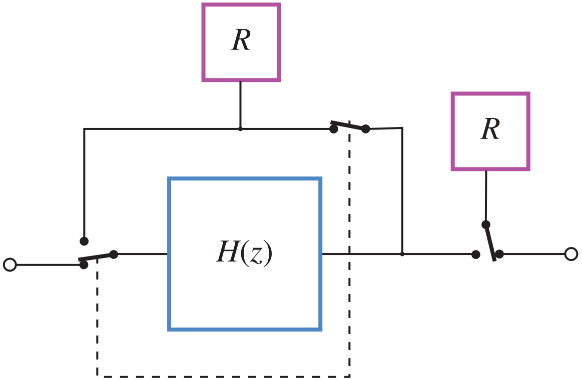
  In other words, if the specifications call for an overall passband ripple and a minimum stopband attenuation of A_p and A_a, respectively, then the corresponding specifications for each filter should be $A_p/2$ and $A_a/2$, respectively.

  See Sec. 13.5.2 of textbook for more details.



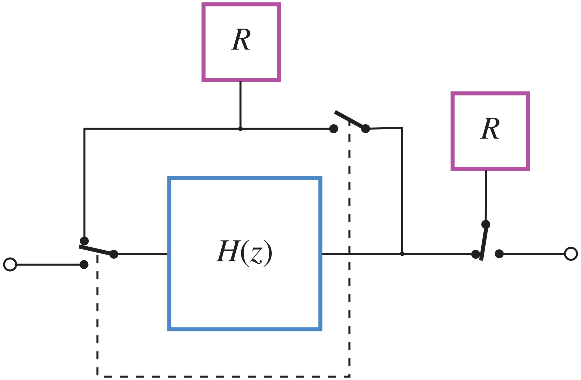
Zero-Phase Filters *Cont'd*

- Single-filter implementation – Phase #1:



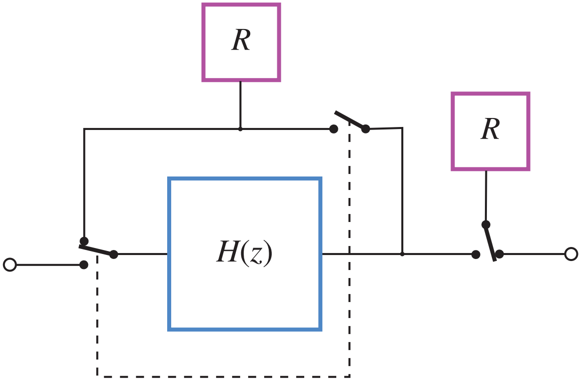
Zero-Phase Filters *Cont'd*

- Single-filter implementation – Phase #2:



Zero-Phase Filters *Cont'd*

- Single-filter implementation – Phase #3:



Summary

- A great variety of optimization algorithms can be used for the design of recursive filters.

Summary

- A great variety of optimization algorithms can be used for the design of recursive filters.
- Quasi-Newton algorithms work very well.

Summary

- A great variety of optimization algorithms can be used for the design of recursive filters.
- Quasi-Newton algorithms work very well.
- The optimization approach is very flexible in that it can be used to design filters with arbitrary amplitude and/or phase responses.

Summary

- A great variety of optimization algorithms can be used for the design of recursive filters.
- Quasi-Newton algorithms work very well.
- The optimization approach is very flexible in that it can be used to design filters with arbitrary amplitude and/or phase responses.
- In FIR filters as well as recursive filters based on the bilinear transformation, techniques are available that can be used to predict the required filter order to achieve prescribed specifications.

Summary

- A great variety of optimization algorithms can be used for the design of recursive filters.
- Quasi-Newton algorithms work very well.
- The optimization approach is very flexible in that it can be used to design filters with arbitrary amplitude and/or phase responses.
- In FIR filters as well as recursive filters based on the bilinear transformation, techniques are available that can be used to predict the required filter order to achieve prescribed specifications.

  Unfortunately, in recursive filters designed by optimization the filter order can only be deduced through a cut-and-try approach.

Summary

- A great variety of optimization algorithms can be used for the design of recursive filters.
- Quasi-Newton algorithms work very well.
- The optimization approach is very flexible in that it can be used to design filters with arbitrary amplitude and/or phase responses.
- In FIR filters as well as recursive filters based on the bilinear transformation, techniques are available that can be used to predict the required filter order to achieve prescribed specifications.

  Unfortunately, in recursive filters designed by optimization the filter order can only be deduced through a cut-and-try approach.
- As in any other methodology based on optimization, a large amount of computation is required to complete a design.

This slide concludes the presentation.
Thank you for your attention.