Part 4: IIR Filters – Optimization Approach

Tutorial ISCAS 2007

Victoria, BC, Canada
Email: aantoniou@ieee.org

July 24, 2007

Introduction

IIR filters that would satisfy prescribed specifications can be designed by using a variety of optimization algorithms.

Five general steps are involved, as follows:

1. Formulate a suitable objective (cost) function.
2. Deduce the gradient of the objective function and, if necessary, the Hessian matrix.
3. Choose an appropriate weighting function.
4. Select a suitable optimization algorithm.
5. Run the optimization algorithm for increasing filter orders until an order is found that will satisfy the required specifications.

Note: The material for this module is taken from Antoniou, *Digital Signal Processing: Signals, Systems, and Filters*, Chap. 16.

Problem Formulation

- An Nth-order IIR filter can be represented by a discrete-time transfer function of the form

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

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

Problem Formulation

- An Nth-order IIR filter can be represented by a discrete-time transfer function of the form

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

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

- As presented, $H(z)$ would be of even order; however, an odd-order $H(z)$ can be obtained by letting

$$a_{0j} = b_{0j} = 0$$

  for one value of j.

Problem Formulation *Cont'd*

The amplitude response of an arbitrary IIR filter can be deduced as

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

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.

Problem Formulation *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 $j = 1, 2, \ldots, J$.

Problem Formulation *Cont'd*

- If $M_0(\omega)$ is the specified amplitude response, the approximation error can be expressed as

$$e(\mathbf{x}, \omega) = M(\mathbf{x}, \omega) - M_0(\omega)$$

Problem Formulation *Cont'd*

- If $M_0(\omega)$ is the specified amplitude response, the approximation error can be expressed as

$$e(\mathbf{x}, \omega) = M(\mathbf{x}, \omega) - M_0(\omega)$$

- By sampling $e(\mathbf{x}, \omega)$ at frequencies $\omega_1, \ \omega_2, \ \ldots, \ \omega_K$, the column vector

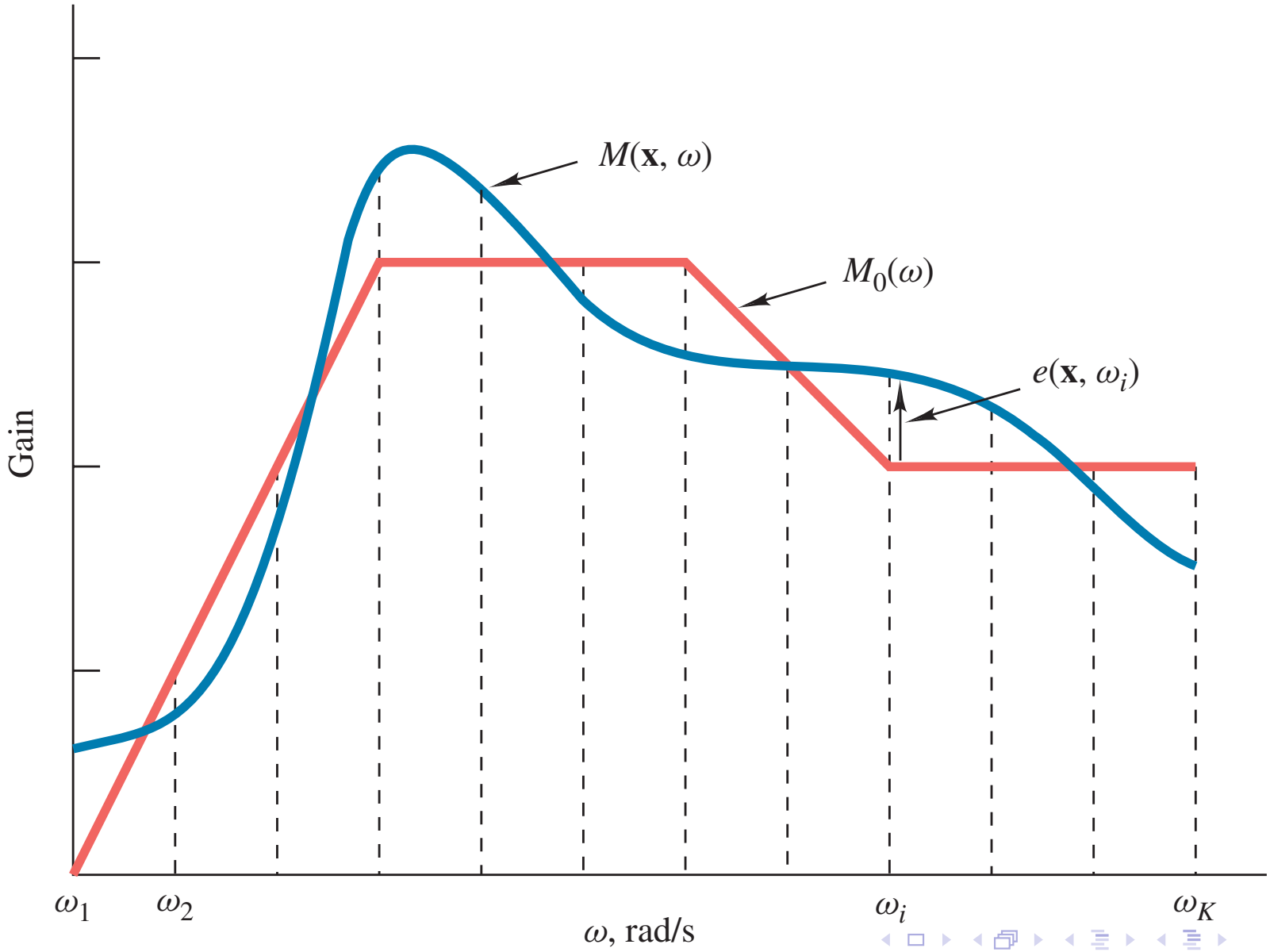
$$\mathbf{E}(\mathbf{x}) = [e_1(\mathbf{x}) \ e_2(\mathbf{x}) \ \ldots \ \ e_K(\mathbf{x})]^T$$

can be formed where

$$e_i(\mathbf{x}) = e(\mathbf{x}, \omega_i)$$

for $i = 1, 2, \ldots, K$.

Problem Formulation *Cont'd*



Problem Formulation *Cont'd*

- An IIR filter can be designed by finding a point $\mathbf{x} = \breve{\boldsymbol{x}}$ such that

$$e_i(\breve{\boldsymbol{x}}) \approx 0 \quad \text{for} \quad i = 1, 2, \ldots, K$$

Problem Formulation *Cont'd*

- An IIR filter can be designed by finding a point $\mathbf{x} = \breve{\boldsymbol{x}}$ such that

$$e_i(\breve{\boldsymbol{x}}) \approx 0 \quad \text{for} \quad i = 1, 2, \ldots, K$$

- Such a point can be obtained by minimizing the *L_p norm* of $\mathbf{E}(\mathbf{x})$, which is defined 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 is a positive integer.

Problem Formulation *Cont'd*

For filter design, the most important norms are the L_2 and L_∞ norms which are defined as

$$L_2(\mathbf{x}) = \left[\sum_{i=1}^{K} |e_i(\mathbf{x})|^2\right]^{1/2}$$

and

$$\begin{aligned} L_\infty(\mathbf{x}) &= \lim_{p\to\infty}\left\{\sum_{i=1}^{K} |e_i(\mathbf{x})|^p\right\}^{1/p} \\ &= \widehat{E}(\mathbf{x}) \lim_{p\to\infty}\left\{\sum_{i=1}^{K}\left[\frac{|e_i(\mathbf{x})|}{\widehat{E}(\mathbf{x})}\right]^p\right\}^{1/p} \\ &= \widehat{E}(\mathbf{x}) \end{aligned}$$

where $\widehat{E}(\mathbf{x}) = \max_{1 \le i \le K} |e_i(\mathbf{x})|$.

Problem Formulation *Cont'd*

- In summary, an IIR filter with a amplitude response that approaches a specified amplitude response $M_0(\omega)$ can be designed by solving the optimization problem

$$\underset{\boldsymbol{x}}{\text{minimize}}\ \Psi(\mathbf{x})$$

Problem Formulation *Cont'd*

- In summary, an IIR filter with a amplitude response that approaches a specified amplitude response $M_0(\omega)$ can be designed by solving the optimization problem

$$\underset{\boldsymbol{x}}{\text{minimize}}\ \Psi(\mathbf{x})$$

- If

$$\Psi(\mathbf{x}) = L_2(\mathbf{x})$$

a *least-squares* solution is obtained and if

$$\Psi(\mathbf{x}) = L_\infty(\mathbf{x})$$

the outcome will be a so-called *minimax* solution.

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

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

where

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

with

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

Gradient of Objective Function *Cont'd*

The elements of $\nabla e_i(\mathbf{x})$ can be deduced as follows by differentiating the amplitude response:

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

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

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

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

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

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

Optimization Algorithms

- The optimization problem under consideration can be solved by using a variety of algorithms, such as the steepest-descent or Newton algorithm or one of several quasi-Newton algorithms.

Optimization Algorithms

- The optimization problem under consideration can be solved by using a variety of algorithms, such as the steepest-descent or Newton algorithm or one of several quasi-Newton algorithms.
- Quasi-Newton algorithms offer a number of important advantages as follows:
  - They do not require the Hessian matrix.
  - Can be used with inexact line searches which lead to improved efficiency.
  - Are robust.
  - Offer fast convergence.

Quasi-Newton Algorithms

A generic quasi-Newton Algorithm is as follows:

1. Input $\mathbf{x}_0$ and ε. Set $\mathbf{S}_0 = \mathbf{I}_n$ and $k = 0$. Compute $\mathbf{g}_0$.

Quasi-Newton Algorithms

A generic quasi-Newton Algorithm is as follows:

1. Input $\mathbf{x}_0$ and ε. Set $\mathbf{S}_0 = \mathbf{I}_n$ and $k = 0$. Compute $\mathbf{g}_0$.
2. Set $\mathbf{d}_k = -\mathbf{S}_k \mathbf{g}_k$ and find α_k, the value of α that minimizes $f(\mathbf{x}_k + \alpha\, \mathbf{d}_k)$, using a line search.

Quasi-Newton Algorithms

A generic quasi-Newton Algorithm is as follows:

1. Input $\mathbf{x}_0$ and ε. Set $\mathbf{S}_0 = \mathbf{I}_n$ and $k = 0$. Compute $\mathbf{g}_0$.
2. Set $\mathbf{d}_k = -\mathbf{S}_k \mathbf{g}_k$ and find α_k, the value of α that minimizes $f(\mathbf{x}_k + \alpha\, \mathbf{d}_k)$, using a line search.
3. Set $\delta_k = \alpha_{\,k} \mathbf{d}_k$ and $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta_k$, and compute $f_{k+1} = f(\mathbf{x}_{k+1})$.

Quasi-Newton Algorithms

A generic quasi-Newton Algorithm is as follows:

1. Input $\mathbf{x}_0$ and ε. Set $\mathbf{S}_0 = \mathbf{I}_n$ and $k = 0$. Compute $\mathbf{g}_0$.
2. Set $\mathbf{d}_k = -\mathbf{S}_k \mathbf{g}_k$ and find α_k, the value of α that minimizes $f(\mathbf{x}_k + \alpha\, \mathbf{d}_k)$, using a line search.
3. Set $\delta_k = \alpha_k \mathbf{d}_k$ and $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta_k$, and compute $f_{k+1} = f(\mathbf{x}_{k+1})$.
4. If $\|\delta_k\|_2 < \varepsilon$, then output $\breve{\boldsymbol{x}} = \boldsymbol{x}_{k+1}$, $f(\breve{\boldsymbol{x}}) = f_{k+1}$ and stop.

Quasi-Newton Algorithms

A generic quasi-Newton Algorithm is as follows:

1. Input $\mathbf{x}_0$ and ε. Set $\mathbf{S}_0 = \mathbf{I}_n$ and $k = 0$. Compute $\mathbf{g}_0$.
2. Set $\mathbf{d}_k = -\mathbf{S}_k \mathbf{g}_k$ and find α_k, the value of α that minimizes $f(\mathbf{x}_k + \alpha\, \mathbf{d}_k)$, using a line search.
3. Set $\delta_k = \alpha_k \mathbf{d}_k$ and $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta_k$, and compute $f_{k+1} = f(\mathbf{x}_{k+1})$.
4. If $\|\delta_k\|_2 < \varepsilon$, then output $\breve{\boldsymbol{x}} = \boldsymbol{x}_{k+1}$, $f(\breve{\boldsymbol{x}}) = f_{k+1}$ and stop.
5. Compute $\mathbf{g}_{k+1}$ and set $\boldsymbol{\gamma}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$.

Quasi-Newton Algorithms

A generic quasi-Newton Algorithm is as follows:

1. Input $\mathbf{x}_0$ and ε. Set $\mathbf{S}_0 = \mathbf{I}_n$ and $k = 0$. Compute $\mathbf{g}_0$.
2. Set $\mathbf{d}_k = -\mathbf{S}_k \mathbf{g}_k$ and find α_k, the value of α that minimizes $f(\mathbf{x}_k + \alpha\, \mathbf{d}_k)$, using a line search.
3. Set $\delta_k = \alpha_k \mathbf{d}_k$ and $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta_k$, and compute $f_{k+1} = f(\mathbf{x}_{k+1})$.
4. If $\|\delta_k\|_2 < \varepsilon$, then output $\breve{\boldsymbol{x}} = \boldsymbol{x}_{k+1}$, $f(\breve{\boldsymbol{x}}) = f_{k+1}$ and stop.
5. Compute $\mathbf{g}_{k+1}$ and set $\boldsymbol{\gamma}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$.
6. Compute $\mathbf{S}_{k+1} = \mathbf{S}_k + \mathbf{C}_k$.

Quasi-Newton Algorithms

A generic quasi-Newton Algorithm is as follows:

1. Input $\mathbf{x}_0$ and ε. Set $\mathbf{S}_0 = \mathbf{I}_n$ and $k = 0$. Compute $\mathbf{g}_0$.
2. Set $\mathbf{d}_k = -\mathbf{S}_k \mathbf{g}_k$ and find α_k, the value of α that minimizes $f(\mathbf{x}_k + \alpha\, \mathbf{d}_k)$, using a line search.
3. Set $\delta_k = \alpha_k \mathbf{d}_k$ and $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta_k$, and compute $f_{k+1} = f(\mathbf{x}_{k+1})$.
4. If $\|\delta_k\|_2 < \varepsilon$, then output $\breve{\boldsymbol{x}} = \boldsymbol{x}_{k+1}$, $f(\breve{\boldsymbol{x}}) = f_{k+1}$ and stop.
5. Compute $\mathbf{g}_{k+1}$ and set $\gamma_k = \mathbf{g}_{k+1} - \mathbf{g}_k$.
6. Compute $\mathbf{S}_{k+1} = \mathbf{S}_k + \mathbf{C}_k$.
7. Check $\mathbf{S}_{k+1}$ for positive definiteness and if it is found to be nonpositive definite force it to become positive definite.

Quasi-Newton Algorithms

A generic quasi-Newton Algorithm is as follows:

1. Input $\mathbf{x}_0$ and ε. Set $\mathbf{S}_0 = \mathbf{I}_n$ and $k = 0$. Compute $\mathbf{g}_0$.
2. Set $\mathbf{d}_k = -\mathbf{S}_k \mathbf{g}_k$ and find α_k, the value of α that minimizes $f(\mathbf{x}_k + \alpha\, \mathbf{d}_k)$, using a line search.
3. Set $\delta_k = \alpha_k \mathbf{d}_k$ and $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta_k$, and compute $f_{k+1} = f(\mathbf{x}_{k+1})$.
4. If $\|\delta_k\|_2 < \varepsilon$, then output $\breve{\boldsymbol{x}} = \boldsymbol{x}_{k+1}$, $f(\breve{\boldsymbol{x}}) = f_{k+1}$ and stop.
5. Compute $\mathbf{g}_{k+1}$ and set $\gamma_k = \mathbf{g}_{k+1} - \mathbf{g}_k$.
6. Compute $\mathbf{S}_{k+1} = \mathbf{S}_k + \mathbf{C}_k$.
7. Check $\mathbf{S}_{k+1}$ for positive definiteness and if it is found to be nonpositive definite force it to become positive definite.
8. Set $k = k + 1$ and go to step 2.

Quasi-Newton Algorithms *Cont'd*

Notes:

- $\boldsymbol{x}_0$ is the initial point.
- ε is a termination tolerance.
- $\boldsymbol{I}_n$ is the $n \times n$ identity matrix.
- $\boldsymbol{d}_k$ is the direction vector at the start of the kth iteration.
- $\boldsymbol{g}_k$ is the gradient at the start of the kth iteration.
- $\boldsymbol{S}_k$ is an approximation to the inverse Hessian at the start of the kth iteration.
- $\boldsymbol{C}_k$ is a correction matrix.
- $\breve{\boldsymbol{x}}$ is the minimum point and $f(\breve{\boldsymbol{x}})$ is the minimum value of the objective function.
- Efficiency can be achieved by using Fletcher's inexact line search in Step 2.

Quasi-Newton Algorithms *Cont'd*

- Several quasi-Newton algorithms are available. They differ from one another in the formula used to update matrix $\boldsymbol{S}_{k+1}$ in Step 7 of the generic quasi-Newton algorithm presented.

Quasi-Newton Algorithms *Cont'd*

- Several quasi-Newton algorithms are available. They differ from one another in the formula used to update matrix $\boldsymbol{S}_{k+1}$ in Step 7 of the generic quasi-Newton algorithm presented.
- The two most important algorithms of this family are the Davidon-Fletcher-Powell (DFP) algorithm in which

$$\mathbf{S}_{k+1} = \mathbf{S}_k + \frac{\delta_k \delta_k^T}{\gamma_k^T \delta_k} - \frac{\mathbf{S}_k \gamma_k \gamma_k^T \mathbf{S}_k}{\gamma_k^T \mathbf{S}_k \gamma_k}$$

and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm in which

$$\mathbf{S}_{k+1} = \mathbf{S}_k + \left(1 + \frac{\gamma_k^T \mathbf{S}_k \gamma_k}{\gamma_k^T \delta_k}\right) \frac{\delta_k \delta_k^T}{\gamma_k^T \delta_k} - \frac{(\delta_k \gamma_k^T \mathbf{S}_k + \mathbf{S}_k \gamma_k \delta_k^T)}{\gamma_k^T \delta_k}$$

Choice between L_2 and L_∞ Solutions

- L_2 solutions are easier to obtain and filters based on these solutions will reject more signal power in stopbands.

  However, the passband error tends to have large peaks near passband edges, which are undesirable in practice.

Choice between L_2 and L_∞ Solutions

- L_2 solutions are easier to obtain and filters based on these solutions will reject more signal power in stopbands.

  However, the passband error tends to have large peaks near passband edges, which are undesirable in practice.

- L_∞ solutions are more difficult to obtain because they require the application of *minimax algorithms* which entail much more computation.

  However, they offer the advantage that the approximation error tends to be uniformly distributed in passbands, i.e., it tends to be equiripple as in elliptic filters.

Minimax Algorithms

- Minimax algorithms are essentially sequential algorithms that involve a series of unconstrained optimizations.
- A representative algorithm of this class is the so-called *least-pth algorithm*.
- Another popular minimax algorithm is one proposed by Charalambous (see References).

Least-pth Minimax Algorithm

1. Input $\breve{\boldsymbol{x}}_0$ and ε_1. Set $k = 1$, $p = 2$, $\mu = 2$, $\widehat{E}_0 = 10^{99}$.

Least-pth Minimax Algorithm

1. Input $\breve{\boldsymbol{x}}_0$ and ε_1. Set $k = 1$, $p = 2$, $\mu = 2$, $\widehat{E}_0 = 10^{99}$.
2. Initialize frequencies $\omega_1, \omega_2, \ldots, \omega_K$.

Least-pth Minimax Algorithm

1. Input $\breve{\boldsymbol{x}}_0$ and ε_1. Set $k = 1$, $p = 2$, $\mu = 2$, $\widehat{E}_0 = 10^{99}$.
2. Initialize frequencies $\omega_1, \omega_2, \ldots, \omega_K$.
3. Using $\breve{\boldsymbol{x}}_{k-1}$ as initial value, minimize

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

where

$$\widehat{E}(\mathbf{x}) = \max_{1 \le i \le K} |e_i(\mathbf{x})|$$

with respect to $\mathbf{x}$, to obtain $\breve{\boldsymbol{x}}_k$. Set $\widehat{E}_k = \widehat{E}(\breve{\boldsymbol{x}})$.

Least-pth Minimax Algorithm

1. Input $\breve{\boldsymbol{x}}_0$ and ε_1. Set $k = 1$, $p = 2$, $\mu = 2$, $\widehat{E}_0 = 10^{99}$.
2. Initialize frequencies $\omega_1, \omega_2, \ldots, \omega_K$.
3. Using $\breve{\boldsymbol{x}}_{k-1}$ as initial value, minimize

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

where

$$\widehat{E}(\mathbf{x}) = \max_{1 \le i \le K} |e_i(\mathbf{x})|$$

with respect to $\mathbf{x}$, to obtain $\breve{\boldsymbol{x}}_k$. Set $\widehat{E}_k = \widehat{E}(\breve{\boldsymbol{x}})$.

4. If $|\widehat{E}_{k-1} - \widehat{E}_k| < \varepsilon_1$, then output $\breve{\boldsymbol{x}}_k$ and $\widehat{E}_k$, and stop. Otherwise, set $p = \mu p$, $k = k + 1$ and go to step 3.

Stability

- Least-squares or minimax algorithms will 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 will 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

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

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

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

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

$$H'(z) = H_0\frac{N(z)}{D'(z)\prod_{i=1}^{k}(z-1/\tilde{p}_i)} = \frac{\sum_{i=0}^{N} a'_i z^i}{1+\sum_{i=1}^{N} b'_i z^i}$$

where

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

Prescribed Specifications

- Optimization algorithms in general tend to yield filters in which the maximum values of the approximation error in the various passbands and stopbands are of the same order.

Prescribed Specifications

- Optimization algorithms in general tend to yield filters in which the maximum values of the approximation error in the various passbands and stopbands are of the same order.
- In practice, the prescribed passband ripples and minimum stopband attenuations vary wildly from band to band and from one application to the next, which means that the required maximum values of the passband and stopband errors also vary wildly.

Prescribed Specifications

- Optimization algorithms in general tend to yield filters in which the maximum values of the approximation error in the various passbands and stopbands are of the same order.
- In practice, the prescribed passband ripples and minimum stopband attenuations vary wildly from band to band and from one application to the next, which means that the required maximum values of the passband and stopband errors also vary wildly.
- In order to achieve desired specifications, we need be able to control the relative magnitudes of the maximum passband and stopband errors, and this is achieved through the use of a *weighting*.

Prescribed Specifications *Cont'd*

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

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

where $W(\omega)$ is a scalar weighting function which is 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

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

  where $W(\omega)$ is a scalar weighting function which is 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, $W(\omega)$ is a piecewise constant function which is assigned a value greater or less than one to decrease or increase the magnitude of the approximation error is a given band.

Prescribed Specifications *Cont'd*

Arbitrary filter specifications can be achieved as follows:

1. Choose a suitable weighting function on the basis of the prescribed specifications.

Prescribed Specifications *Cont'd*

Arbitrary filter specifications can be achieved as follows:

1. Choose a suitable weighting function on the basis of the prescribed specifications.
2. Design filters for increasing filter orders until a filter order is found that would satisfy the required specifications.

   Evidently, this is a cut-and-try method and it could entail a considerable amount of computation.

Prescribed Specifications *Cont'd*

- Assuming a filter with m bands, the weighting function can be deduced from the specified passband ripples and the minimum stopband attenuations as detailed below.

Prescribed Specifications *Cont'd*

- Assuming a filter with m bands, the weighting function can be deduced from the specified passband ripples and the minimum stopband attenuations as detailed below.
- Let $\delta_1, \delta_2, \ldots, \delta_k, \ldots, \delta_m$ be the maximum passband and stopband errors imposed by the filter specifications where

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

for a passband with ripple A_{p_i} dB and

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

for a stopband with a minimum attenuation A_{a_j} dB.

Prescribed Specifications *Cont'd*

- Now assume that the weighting function $W(\omega)$ is a piecewise-constant function with positive values W_1, W_2, ..., W_k, ..., W_m for bands 1, 2, ..., k, ..., m, respectively.

Prescribed Specifications *Cont'd*

- Now assume that the weighting function $W(\omega)$ is a piecewise-constant function with positive values W_1, W_2, ..., W_k, ..., W_m for bands 1, 2, ..., k, ..., m, respectively.
- Suitable scaling constants can be assigned as

$$W_1 = \frac{\delta_k}{\delta_1}, \quad W_2 = \frac{\delta_k}{\delta_2}, \quad \ldots, \quad W_{k-1} = \frac{\delta_k}{\delta_{k-1}}$$
$$W_k = 1$$
$$W_{k+1} = \frac{\delta_k}{\delta_{k+1}}, \quad W_{k+2} = \frac{\delta_k}{\delta_{k+2}}, \quad \ldots, \quad W_m = \frac{\delta_k}{\delta_m}$$

From example, in a BP filter we could assign $W_1 = \delta_2/\delta_1$, $W_2 = 1$, $W_3 = \delta_2/\delta_3$ or $W_1 = 1$, $W_2 = \delta_1/\delta_2$, $W_3 = \delta_1/\delta_3$ or $W_1 = \delta_3/\delta_1$, $W_2 = \delta_3/\delta_2$, $W_3 = 1$.

Design Examples

A variety of design examples can be found in Antoniou, *Digital Signal Processing,* Chap. 14, and in Antoniou and Lu, *Practical Optimization,* Chaps. 9 and 16.

Summary

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

Summary

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

Summary

- A great variety of optimization algorithms can be used for the design of IIR 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 IIR 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 IIR filters based on the bilinear transformation, techniques are availalble that can be used to predict the required filter order to achieve prescribed specifications.

  Unfortunately, in IIR 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 IIR 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 IIR filters based on the bilinear transformation, techniques are availalble that can be used to predict the required filter order to achieve prescribed specifications.

  Unfortunately, in IIR 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.

References

- A. Antoniou, *Digital Signal Processing: Signals, Systems, and Filters*, Chap. 16, McGraw-Hill, 2005.
- A. Antoniou and W.-S. Lu *Practical Optimization: Algorithms and Engineering Applications*, Chaps. 9 and 16, Springer, 2007.
- C. Charalambous, "Acceleration of the least *p*th algorithm for minimax optimization with engineering applications," *Mathematical Programming*, vol. 17, pp. 270–297, 1979.
- A. Antoniou, "Improved minimax optimisation algorithms and their application in the design of recursive digital filters," *Proc. Inst. Elect. Eng.*, Part G, vol. 138, pp. 724–730, Dec. 1991.

References

- A. Antoniou and W.-S. Lu, “Design of two-dimensional digital filters by using the singular value decomposition,” *IEEE Trans. Circuits Syst.*, vol. 34, pp. 1191–1198, Oct. 1987.
- W.-S. Lu and A. Antoniou, “Design of digital filters and filter banks by optimization: A state of the art review,” in *Proc. 2000 European Signal Processing Conference*, vol. 1, pp. 351–354, Tampere, Finland, Sept. 2000.

This slide concludes the presentation.

Thank you for your attention.