Optimum Design of Frequency-Response-Masking Filters Using Convex-Concave Procedure

by

Haiying Chen

B.Eng., University of Electrical Science and Technology of China, 2013

A Dissertation Submitted in partial fulfillment of the

Requirement of the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

University of Victoria

Optimum Design of Frequency-Response-Masking Filters Using Convex-Concave Procedure

by

Haiying Chen

B.Eng., University of Electrical Science and Technology of China, 2013

Supervisory Committee

Dr. Wu-Sheng Lu, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Hong-Chuan Yang, Departmental Member

(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Wu-Sheng Lu, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Hong-Chuan Yang, Departmental Member

(Department of Electrical and Computer Engineering)

Abstract

The class of frequency-response-masking (FRM) filters, first investigated by Y.C. Lim in 1980s, has shown to provide considerably improved performance for the design of finite impulse response (FIR) filters with narrow transition bandwidth relative to its conventional counterparts. In this project report, we present a design method for FRM filters by using a technique known as convex-concave procedure (CCP). For illustration purposes, we begin by reviewing several basic concepts and properties of FIR filters as well as a popular design technique based on windowed Fourier series. This is followed by a chapter to introduce the structure of basic FRM filters and to explain how they work. The CCP is then studied in the next chapter as an effective heuristic for nonconvex constrained optimization problems. In the last chapter, we present a formulation procedure that converts a frequency-weighted minimax design of FRM filters to a problem which can be solved by iteratively performing CCP. Simulation runs are included to illustrate the proposed algorithm and evaluate the design performance in comparison with those achieved by several existing methods.

Contents

List of Figures

List of Tables

List of Abbreviations

CCP	**C**onvex-**C**oncave **P**rocedure
CP	**C**onvex **P**rogramming
DC	**D**ifference of **C**onvex functions
DSP	**D**igital **S**ignal **P**rocessing
FIR	**F**inite **I**mpulse **R**esponse
IFIR	**I**nterpolated **F**inite **I**mpulse **R**esponse
IIR	**I**nfinite **I**mpulse **R**esponse
LP	**L**inear **P**rogramming
SCP	**S**equential **C**onvex **P**rogramming
SSF	**S**ingle **S**tage **F**ilter

ACKNOWLEDGEMENTS

It is an honor for me to take this opportunity to express my gratitude to my supervisor Dr. Wu-Sheng Lu who has been guiding me throughout this work. Not only for his wisdom and wide range of knowledge, but also his unquenched curiosity and wholehearted dedication to research manifest to me the spirit of a real scientist. I am fortunate to have him as my supervisor.

I would like to thank Dr. Hong-Chuan Yang for serving as my committee member. I learned useful knowledge from his course and I am influenced by him in a number of ways.

I am also grateful to the Department of Electrical and Computer Engineering along with the faculty and staff of the University of Victoria who have provided guidance and assistance countless times over years.

It is a pleasure to express my gratitude to my colleagues and friends who made this thesis possible: Jie Yan for his warmhearted assistance in helping me quickly adapted to the life and study in Canada; Hongyu Bao and Zhen Liu for their company and care to make my life in Canada so joyful; Kathy for acting as both a good teacher and a helpful friend during my Coop terms.

I am also grateful to the Department of Electrical and Computer Engineering along with the faculty and staff of the University of Victoria who have provided guidance and assistance countless times over the years.

My deepest gratitude goes to my parents and my sisters who constantly support me when I am in need. Without your love and encouragement, I would not have been where I am today.

DEDICATION

To my family.

Chapter 1

Introduction

1.1 Frequency Response Masking Technique

Finite impulse response (FIR) filters are digital filters with finite impulse responses [1]. Linear phase FIR filters are widely used in signal processing as they have many advantages such as their inherent stability, free of phase distortion, and low coefficient sensitivity. However, the order and complexity of the FIR filters are very high when the transition bandwidth is narrow. According to [3], for linear phase single stage filters (SSF), the filter length is roughly proportional to the inverse of the width of the transition band.

The original concept of frequency-response masking (FRM) was introduced by Neuvo, Cheng-Yu and Mitra in 1984. It has been demonstrated that the complexity of a linear phase FIR filter can be considerably reduced by using FRM techniques. The FRM filter proposed by Neuvo, Cheng-Yu and Mitra is composed of an interpolated FIR (IFIR) filter and a properly designed FIR filter. The transfer function of an IFIR filter can be obtained by replacing each unit delay $z^{-1}$with a delay block z^{-M}, where M is an integer. In this way, the frequency response of the IFIR filter becomes periodic over the base-band with M-times narrower transition band in each frequency period. The FIR filter in the cascade is then used to *mask* the images from the frequency response of the IFIR filter. While this turned out to be a successful design methodology, its use is limited to the design of filters whose passband is not too wide. In 1986, Lim proposed an improved approach which allows the application of FRM technique to designing a much wider range of linear phase FIR filters. It was shown that the approach given in [4] results in a linear phase FIR filter with a small fraction of nonzero coefficients, and thus is suitable for implementing sharp filters with arbitrary bandwidths. Compared to the classical single-filter design, this technique offers the advantages of lower coefficients' sensitivity, higher computation speed and lower power consumption.

1.2 A Note on the Literature

The frequency-response masking (FRM) technique for the design of finite-impulse response (FIR) digital filters with very narrow transition bands has been a subject of study since 1980s. In early years, the sub-filters of the FRM filters were separately designed [4-8]. As a result, the design was only sub-optimal. In [9], the author proposed a two-step technique for reducing the overall arithmetic complexity by simultaneously optimizing all the sub-filters. First, it optimizes the masking filters by using a simple

iterative design scheme. Second, the design is improved by using an efficient unconstrained nonlinear optimization algorithm [24]. This method is applicable to a basic FRM filter. As the number of filter stages increases, the number of sub-filters that need to be optimized increases quickly, making the design process increasingly involved. A new optimization technique is proposed in [2] where all the coefficients of the sub-filters are treated as a single design vector. The method is applicable to multi-stage FRM filters, and is shown to produce designs with improved performance.

1.3 Contents and Organization of the Report

The report is organized as follows.

Chapter 2 introduces some background information relevant to FIR digital filters to facilitate a better understanding of the work in this project. To begin, some basic concepts finite-impulse-response (FIR) digital filters are introduced. This is followed by the description of a design technique based on windowed Fourier series for linear-phase FIR filters.

Chapter 3 presents an overview of the FRM filtering technique. First, we illustrate the narrow-band FIR filter design. Then, we describe the FRM technique in details in terms of its basic structure and properties. Finally we describe a two-step procedure developed by Lim for designing basic FRM filters.

Chapter 4 introduces basic formulations of optimization problems that include convex optimization and non-convex optimization. We then describe the DC (difference of convex functions) problems. Furthermore, we present the basic CCP (convex-concave procedure) algorithm for solving DC programming problems.

Chapter 5 presents a formulation procedure that converts a frequency-weighted minimax design of FRM filters to a problem which can be solved by iteratively performing CCP. Simulation runs are included to illustrate the proposed algorithm and evaluate the design performance in comparison with those achieved by several existing methods.

Chapter 6 summarizes the main contents of the report and offers suggestion for potential future work.

Chapter 2

Design of Finite-Impulse-Response Filters

In this chapter, basic concepts of finite-impulse-response (FIR) digital filters and design technique for linear-phase FIR filters are reviewed.

2.1 Definition and Properties of FIR Digital Filters

In signal processing, *finite impulse response* (FIR) digital filter refers to a digital filter whose impulse response is of finite length. FIR filters are also known as *non-recursive* digital filters because the output of an FIR filter is solely determined by input samples, and does not depend on past output. In other words, as a dynamic system an FIR filter does not have a feedback loop from the output terminal back to the inside of the system. As a result, FIR filters are always stable meaning that with an input in small magnitude an FIR filter always produces an output with proportionally small magnitude.

2.1.1 Frequency Response of an FIR Filter

An FIR filter is characterized by a linear and non-recursive relation between a digital input signal and a digital output signal, and this relation is explicitly characterized by a linear difference equation as

$$\begin{aligned} y(nT) &= h(0)x(nT) + h(T)x((n-1)T) + \cdots + h((N-1)T)x\big((n-N+1)T\big) \\ &= \textstyle\sum_{k=0}^{N-1} h(kT)x((n-k)T) \end{aligned} \tag{2.1}$$

where T denotes the sampling period.

By applying z-transform to the above equation and denoting the z-transforms of input signal $\{x(n)\}$ and output signal $\{y(n)\}$ as $X(z)$ and $Y(z)$, respectively, we obtain a frequency-domain characterization of the filter as

$$Y(z) = H(z)X(z) \quad \text{or} \quad \frac{Y(z)}{X(z)} = H(z) \tag{2.2}$$

where

$$H(z) = \textstyle\sum_{n=0}^{N-1} h[n]z^{-n} \tag{2.3}$$

is called the *transfer function* of the FIR filter. A block diagram of an FIR filter is illustrated in Fig.2.1.

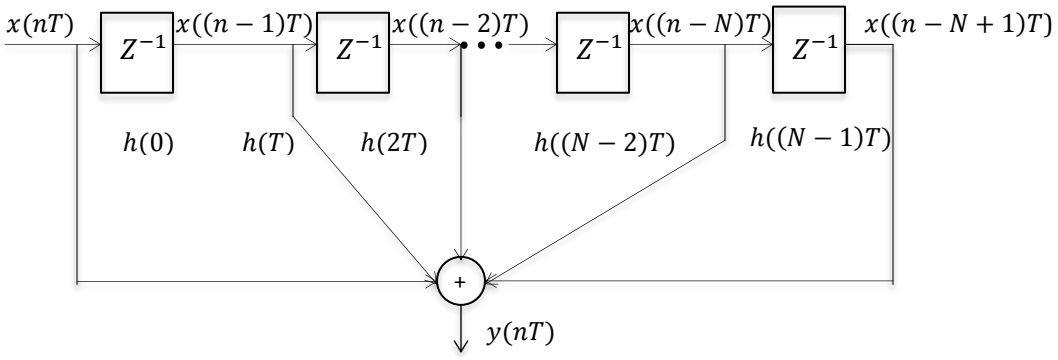


Fig. 2.1. A block diagram of FIR filter

There are a total of *N* terms in $H(z)$ in (2.3), hence the FIR filter is said to have length N. Sometimes the filter is said to have order $N-1$ because $H(z)$ is essentially an $(N-1)$ th-order polynomial in z^{-1}.

The frequency response of the filter is obtained by simply substituting $z = e^{j\omega T}$ into (2.3) that yields

$$H\left(e^{j\omega T}\right) = \sum_{n=0}^{N-1} h(nT)e^{-jn\omega T} \tag{2.4}$$

To explain the term "finite impulse response", let us use the unit pulse, i.e.,

$$x(nT) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{elsewhere} \end{cases} \tag{2.5}$$

as filter's input. It can be verified that the first *N* output samples are exactly the same as the filter's coefficients, i.e.,$\{h[0], h[T], \cdots, h[(N-1)T]\}$, and all output becomes zero afterwards. In other words, the response of an FIR filter to the unit impulse is always finite, thus the term FIR. Also note that for an FIR digital filter its coefficients and impulse response are synonymous and they are often used interchangeably.

Related to the frequency response defined in (2.4) there are the concepts of *amplitude response* and *phase response*, which are defined by

$$M(\omega) = |H\left(e^{j\omega T}\right)|$$

$$\theta(\omega) = \arg[H\left(e^{j\omega T}\right)] \tag{2.6}$$

respectively. And related to the phase response there is the concept of *group delay* which is defined by

$$\tau = -\frac{d\theta(\omega)}{d\omega} \tag{2.7}$$

A digital filter is said to have a *linear phase response* if function $\theta(\omega)$ in (2.6) is linear with respect to frequency ω, i.e.,

$$\theta(\omega) = -\tau\omega + \theta_0 \tag{2.8}$$

Where τ and θ_0 are constants. In many DSP applications, a linear phase response is more desirable relative to a nonlinear phase response. This is because from (2.7) we see that the phase response of a digital filter is linear if its group delay is constant. When an FIR filter applies to an input signal, phase delay is invertible, but delaying all frequency components by a constant amount prevents the filtered signal from phase distortion.

It can be shown [1] that for an FIR filter to have constant group delay with $\theta_0 = 0$, its coefficients must be symmetrical about the midpoint, that is,

$$h(nT) = h((N-1-n)T) \quad \text{for } 0 \le n \le N-1 \tag{2.9}$$

An FIR filter has constant group delay with $\theta_0 = \pm\pi/2$, its impulse response is anti-symmetrical about the midpoint [1], namely,

$$h(nT) = -h((N-1-n)T) \quad \text{for } 0 \le n \le N-1 \tag{2.10}$$

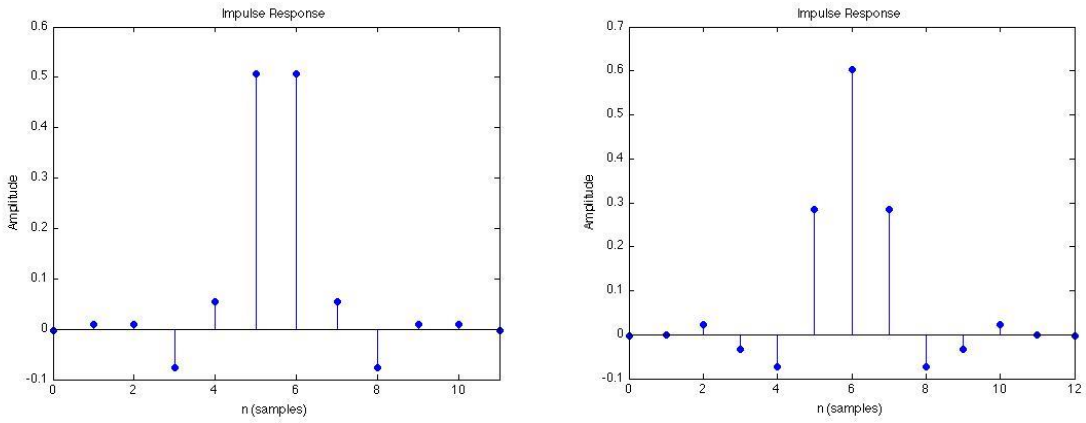


Fig. 2.2. Impulse response for linear phase and constant group delay

for (a) odd N and (b) even N.

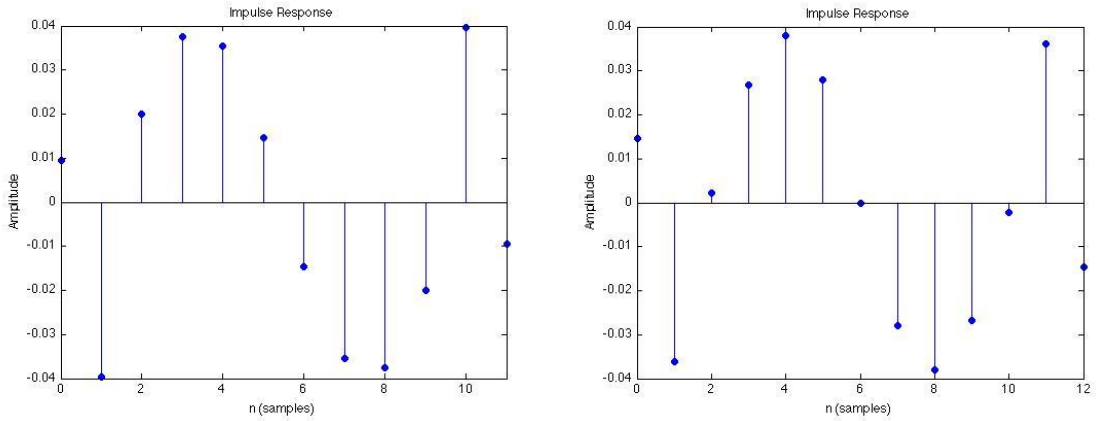


Fig. 2.3. Alternative impulse response for linear phase and constant group delay for (a) odd N and (b) even N.

It is rather straightforward to verify that an FIR filter of length N has linear phase response of form (2.8) where the group delay is given by

$$\tau = \frac{(N-1)T}{2} \tag{2.11}$$

if its impulse response is symmetrical or anti-symmetrical about the midpoint, namely satisfying (2.9) or (2.10). From (2.11), it follows that the group delay of a linear-phase FIR filter is an half of the filter order (hence practically the filter length) times the sampling interval T.

2.1.2 Frequency Response of a Linear-Phase FIR Filter

For a symmetrical impulse response with N odd, Eq. (2.4) can be expressed as

$$H\left(e^{j\omega T}\right) = \sum_{n=0}^{(N-3)/2} h[nT]e^{-jn\omega T} + h[\tfrac{N-1}{2}T]e^{-\frac{j\omega T(N-1)}{2}} + \sum_{n=(N+1)/2}^{N-1} h[nT]e^{-jn\omega T} \tag{2.12}$$

By using Eq. (2.9) and then letting $N-1-n=m$, and then replacing m by n, the last summation in Eq. (2.12) can be expressed as

$$\sum_{n=(N+1)/2}^{N-1} h(nT)e^{-jn\omega T} = \sum_{n=\frac{N+1}{2}}^{N-1} h((N-1-n)T)e^{-jn\omega T}$$

$$= \sum_{n=0}^{(N-3)/2} h(nT)e^{-j(N-1-n)\omega T} \tag{2.13}$$

From (2.12) and (2.13), we obtain

$$H\left(e^{j\omega T}\right) = e^{-j\omega T(N-1)/2}\left\{h(\tfrac{(N-1)T}{2}) + \sum_{n=0}^{(N-3)/2} 2h(nT)\cos[\omega(\tfrac{N-1}{2}-n)T]\right\} \quad (2.14)$$

By letting $k = \frac{N-1}{2} - n$, Eq. (2.14) can be expressed more compactly as

$$H\left(e^{j\omega T}\right) = e^{-j\omega T(N-1)/2}\sum_{k=0}^{(N-1)/2} a_k \cos \omega kT \quad (2.15)$$

where

$$a_0 = h(\tfrac{(N-1)T}{2}) \quad (2.16)$$

$$a_k = 2h[(\tfrac{N-1}{2} - k)T] \qquad \text{for } k = 1, 2, \ldots, (N-1)/2 \quad (2.17)$$

In a similar way, formulas for frequency responses of FIR filters with an even length N and the two cases of anti-symmetrical impulse responses can also be deduced. These formulas are summarized in Table 1 [1].

Table 1 Frequency responses of linear-phase FIR filters.

$h[nT]$	N	$H\left(e^{j\omega T}\right)$
Symmetrical	odd	$e^{-j\omega(N-1)T/2}\sum_{k=0}^{(N-1)/2} a_k \cos \omega kT$
	even	$e^{-j\omega(N-1)T/2}\sum_{k=1}^{N/2} b_k \cos[\omega(k-\frac{1}{2})T]$
Anti-symmetrical	odd	$e^{-j[\frac{\omega(TN-1)}{2}-\pi/2]}\sum_{k=1}^{(N-1)/2} a_k \sin \omega kT$
	even	$e^{-j[\frac{\omega T(N-1)}{2}-\pi/2]}\sum_{k=1}^{N/2} b_k \sin[\omega(k-\frac{1}{2})T]$
$a_0 = h(\frac{(N-1)T}{2})$	$a_k = 2h(\left(\frac{N-1}{2} - k\right)T)$	$b_k = 2h(\left(\frac{N}{2} - k\right)T)$

2.2 Design of Linear-Phase FIR Filters Using Windowed Fourier Series

2.2.1 Connecting a truncated Fourier series of a frequency response to a transfer function of an FIR filter

The frequency response of an FIR filter is a periodic function of ω, hence it can be expressed as a Fourier series

$$H\left(e^{j\omega T}\right) = \sum_{n=-\infty}^{\infty} h(nT)e^{-jn\omega T} \tag{2.18}$$

where

$$h(nT) = \frac{1}{\omega_s}\int_{-\omega_s/2}^{\omega_s/2} H\left(e^{j\omega T}\right) e^{-jn\omega T} d\omega \tag{2.19}$$

and $\omega_s = \frac{2\pi}{T}$. If we let $e^{j\omega T} = z$ in (2.18), we obtain

$$H(z) = \sum_{n=-\infty}^{\infty} h(nT)z^{-n} \tag{2.20}$$

To generate a transfer function of finite order, the series in Eq. (2.20) is truncated by assigning

$$h(nT) = 0 \quad \text{for } |n| > \frac{N-1}{2}$$

which leads (2.20) to

$$H(z) = h(0) + \sum_{n=1}^{(N-1)/2}[h(-nT)z^{n} + h(nT)z^{-n}] \tag{2.21}$$

Note that with a frequency response $H\left(e^{j\omega T}\right)$ that is even-symmetrical with respect to $\omega = 0$, $\{h(nT)$ for $n = \ldots, -2, -1, 0, 1, 2, \ldots\}$ satisfies $h(-nT) = h(nT)$, hence the coefficients of the polynomial in (2.21) are symmetrical with respect to its midpoint. This $H(z)$ in (2.21) is of interest because:

- It approximates a given frequency response $H\left(e^{j\omega T}\right)$ because $H(z)$ is essentially a main portion of its Fourier expansion;
- It has a finite number of terms;
- Its coefficients are symmetrical about its midpoint.

On the other hand, however, truncating a Fourier series inevitably introduces unwanted oscillations in regions near or across discontinuity of $H\left(e^{j\omega T}\right)$. In addition, $H(z)$ in (2.21) contains positive-power terms which are not implementable for filtering real-time signals because terms with positive-power of z in an FIR filter means that the filter requires the input samples that occur in future times. These two problems will be addressed below.

2.2.2 Using a window function to reduce Gibbs' oscillations

The unwanted oscillations in regions near or across discontinuity of $H\left(e^{j\omega T}\right)$ induced by truncating its Fourier series is known as Gibbs' oscillations [25]. An easy way to reduce Gibbs' oscillations is to modify $h(nT)$ obtained from Eq. (2.19) using a discrete-time window function $\omega(nT)$. Specifically, we modify $h(nT)$ to

$$h_\omega(nT) = \omega(nT)\, h(nT) \tag{2.22}$$

The reason why a properly constructed window function can smooth out Gibbs' oscillations can be better explained in the frequency domain where it can be shown that

$$H_\omega(z) = Z\{\omega(nT)h(nT)\} = \frac{1}{2\pi j}\oint_\Gamma H\left(\upsilon\right)W\left(\frac{z}{\upsilon}\right)\upsilon^{-1}d\upsilon \tag{2.23}$$

where $Z\{\cdot\}$ denotes z-transform as applied to a discrete-time sequence, Γ represents a contour in the common region of convergence of $H(\upsilon)$ and $W\left(\frac{z}{\upsilon}\right)$, and

$$H(z) = Z\{h(nT)\} = \sum_{n=-\infty}^{\infty} h(nT)\, z^{-n}$$

$$W(z) = Z\{\omega(nT)\} = \textstyle\sum_{n=-\infty}^{\infty} \omega(nT)\, z^{-n} \tag{2.24}$$

If we let $\upsilon = e^{j\varpi T}$, $z = e^{j\omega T}$, and assume that $H(\upsilon)$ and $W(z/\upsilon)$ converge on the unit circle of the υ plane, then Eq. (2.23) can be expressed as

$$H_\omega\left(e^{j\omega T}\right) = \frac{T}{2\pi}\int_0^{2\pi/T} H(\, e^{j\varpi T})W(e^{j(\omega-\varpi)T})d\varpi \tag{2.25}$$

From (2.25), we see that in the frequency response $H\left(e^{j\omega T}\right)$ is being smoothed out by kernel function $W\left(e^{j\omega T}\right)$, thereby reducing the Gibbs oscillations in the modified frequency response $H_\omega\left(e^{j\omega T}\right)$. In what follows, we review several popular window functions.

A. Rectangular window

The rectangular window is the simplest window, equivalent to replacing all but *N* values of a data sequence by zeros, making it appear as though the waveform suddenly turns on and off.

Analytically the rectangular window of length *N* assumes the following form

$$\omega_R(nT) = \begin{cases} 1, & \text{for } |n| \le \frac{N-1}{2} \\ 0, & \text{otherwise} \end{cases} \tag{2.26}$$

and its frequency spectrum is given by

$$W_R\left(e^{j\omega T}\right) = \frac{\sin\left(\frac{\omega NT}{2}\right)}{\sin\left(\frac{\omega T}{2}\right)} \tag{2.27}$$

whose main-lobe width is $2\omega_s/N$ and the ripple ratio remains relatively independent of *N*.

B. von Hann and Hamming windows

The von Hann window was named after Julius von Hann, which is also known as the *Hanning* window (that sounds similar in name and form to the Hamming window). The Hamming window is optimal in the sense that it minimizes the maximum (nearest) side lobe, giving it a height of about one-fifth of the height that occurs in the Hann window.

The von Hann and Hamming windows are both given by the raised-cosine function:

$$\omega_H(nT) = \begin{cases} \alpha + (1-\alpha)\cos\frac{2\pi n}{N-1} & \text{for } |n| \le \frac{N-1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{2.28}$$

where $\alpha = 0.5$ for the von Hann window and $\alpha = 0.54$ for the Hamming window. It turns out that the increase in the value of α from 0.5 to 0.54 reduces ripple ratio by 50 percent compared to the von Hann window.

The spectrum of the window functions is found to be

$$W_H\left(e^{j\omega T}\right) = \frac{\alpha\sin\left(\frac{\omega NT}{2}\right)}{\sin\left(\frac{\omega T}{2}\right)} + \frac{1-\alpha}{2}\cdot\frac{\sin\left[\frac{\omega NT}{2} - \frac{N\pi}{N-1}\right]}{\sin\left[\frac{\omega T}{2} - \frac{\pi}{N-1}\right]} + \frac{1-\alpha}{2}\cdot\frac{\sin\left[\frac{\omega NT}{2} + \frac{N\pi}{N-1}\right]}{\sin\left[\frac{\omega T}{2} + \frac{\pi}{N-1}\right]} \tag{2.29}$$

C. Blackman window

The Blackman windows are defined as

$$\omega_B(nT) = \begin{cases} 0.42 + 0.5\cos\frac{2\pi n}{N-1} + 0.08\cos\frac{4\pi n}{N-1} & \text{for } |n| \le \frac{N-1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{2.30}$$

Compared to the previous two windows, Blackman leads to a further reduction in the amplitude of Gibbs' oscillations. As we can see from Table 2, as the ripple ratio decreases from one window to the next one, the main-lobe width increases. This happens to be a fairly general trade-off among many window functions.

Table 2 Parameters of some window functions.

Type of window	Main-lobe width	Ripple ratio, %		
		N=11	*N*=21	*N*=101
Rectangular	$\frac{2\omega_s}{N}$	22.34	21.89	21.70
von Hann	$\frac{4\omega_s}{N}$	2.62	2.67	2.67
Hamming	$\frac{4\omega_s}{N}$	1.47	0.93	0.74
Blackman	$\frac{6\omega_s}{N}$	0.08	0.12	0.12

D. Dolph-Chebyshev window

The Dolph-Chebyshev window is given by

$$\omega_{DC}(nT) = \frac{1}{N}\left[\frac{1}{r} + 2\sum_{i=1}^{(N-1)/2} T_{N-1}\left(x_0 \cos\frac{i\pi}{N}\right)\cos\frac{2ni\pi}{N}\right] \tag{2.31}$$

for $n = 0, 1, 2, \ldots, (N-2)/2$, where r is the required ripple radio as a fraction and

$$x_0 = \cosh(\frac{1}{N-1})\cosh^{-1}\frac{1}{r}$$

Function $T_k(x)$ in (2.31) is the *k*th-order Chebyshev polynomial, which is given by

$$T_k(x) = \begin{cases} \cos(k\cos^{-1}x) & \text{for } |x| \le 1 \\ \cosh(\cosh^{-1}x) & \text{for } |x| > 1 \end{cases} \tag{2.32}$$

The Dolph-Chebyshev window is optimal in that it has a minimum main-lobe width for a given side-lobe attenuation. This property is desirable when it comes to design FIR filters with narrow transition band. Another useful property of the Chebyshev window is that it is *equiripple*, i.e., the side-lobe height remains the same over the entire frequency band. With this property, the approximation error tends to be somewhat more uniformly distributed with respect to frequency.

2.2.3 FIR filter design using windowed Fourier series

In addition to the use of a window function to modify $\{h(nT)\}$ to reduce Gibbs oscillations, we have to deal with the *causality* problem as discussed in Sec. 2.2.1. To this end, we recall Eq. (2.21), namely,

$$H(z) = h(0) + \sum_{n=1}^{(N-1)/2} [h(-nT)z^n + h(nT)z^{-n}]$$

In order to make above $H(z)$ causal without changing its frequency response, we multiply $H(z)$ by $z^{-(N-1)/2}$ which gives

$$H_d(z) = \sum_{n=0}^{N-1} h_d(nT)z^{-n} \qquad (2.33a)$$

where

$$h_d(nT) = h[(n-(N-1)/2)T] \quad \text{for } n = 0,1,\ldots,N-1 \qquad (2.33b)$$

Evidently, $H_d(z)$ in (2.33a) is a causal because it does not contain positive-power terms.

The method for the design of a linear-phase FIR filter of length N that approximates a desired frequency response $H_d(e^{j\omega})$ consists of three steps:

(i) Compute the impulse response of the ideal filter $\{h_d[n], n = 0, 1, \ldots, N-1\}$ using (2.33b) where $h[nT]$ is computed using (2.19);

(ii) Select and compute a window function $\{\omega(nT),\ n = 0,1,\ldots,N-1\}\}$;

(iii) Obtain the impulse response as $\{\omega(nT)h(nT), for\ n = 0,1,\ldots,N-1\}$

Let us take a low-pass linear phase response FIR filter for example, which is defined by

$$H_d(\omega) = \begin{cases} 1 & \text{if } |\omega| \le \omega_c \\ 0 & \text{if } |\omega| > \omega_c \end{cases}$$

It can be shown easily that the impulse response of the filter is given by

$$h(nT) = \frac{1}{\omega_s}\int_{-\frac{\omega_s}{2}}^{\frac{\omega_s}{2}} H\left(e^{j\omega T}\right)e^{-jn\omega T}d\omega = \frac{1}{\omega_s}\int_{-\omega_c}^{\omega_c} e^{j\omega nT}\,d\omega$$

$$= \frac{1}{n\pi}\frac{e^{j\omega_c nT} - e^{-j\omega_c nT}}{2j} = \frac{1}{n\pi}\sin(\omega_c nT)$$

and select rectangular window function:

$$\omega(nT) = \begin{cases} 1, & |n| \le \dfrac{(N-1)}{2} \\ 0, & \text{otherwise} \end{cases}$$

the impulse response is

$$h_\omega(nT) = \omega(nT)h(nT) = \begin{cases} \dfrac{1}{n\pi}\sin(\omega_c nT), |n| \le \dfrac{(N-1)}{2} \\ 0 \quad \text{otherwise} \end{cases}$$

2.3 Advantages and disadvantages of FIR filters

FIR filters have the following advantages:

- FIR filters can achieve linear phase response and pass a signal without phase distortion. This is to say, linear-phase filters do delay the input signal but they don't distort the phase of the input signal.

- They are easier to implement. In effect, for most DSP microprocessors, the calculations an FIR carry out can be done by looping a single instruction.

- FIR filters are realized non-recursively, hence they are inherently stable and free of limit cycle oscillation when implemented on a finite word-length digital system.

However, FIR filters also have some disadvantages:

- FIR digital filters require more memory and/or calculation to achieve a given filter response characteristic relative to their IIR counterparts.

- As the transition bandwidth becomes narrower, the filter order, and correspondingly the arithmetic complexity, increases inversely proportionally to this bandwidth. As a result, an FIR filter of great length is required in order to approximate a frequency response with very narrow transition band.

Chapter 3

Frequency Response Masking FIR Filters

3.1 Introduction

Realization of FIR filters based on frequency-response masking (FRM) [4] has proven efficient, especially for very sharp FIR filters. An FIR is said to have *sharp* frequency response if its transition band(s) are extremely narrow. It is well known that design of an FIR filter with narrow transition bands usually implies a long filter length, however the effective filter length can be considerably reduced when an FRM is employed. In this chapter, we describe the structure of a basic FRM filter and explain how it works. We begin with a discussion on narrow-band FIR filters as a motivation of our study of FRM filters.

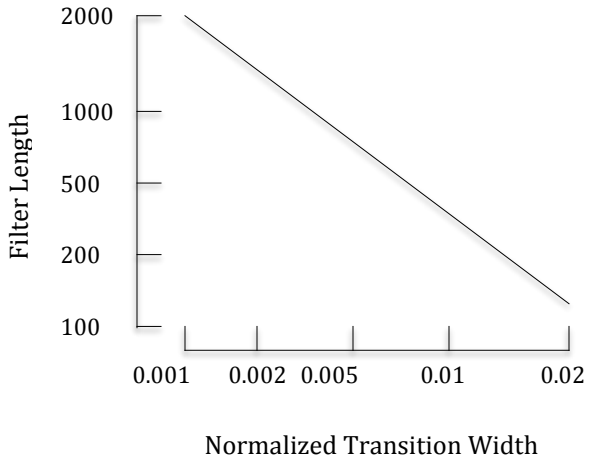


Fig. 3.1. Filter length versus normalized transition bandwidth.

3.2 Narrow-band FIR filter design

As discussed in Section 2.3, a major drawback of linear-phase FIR filters is that its length is inversely proportional to its transition bandwidth. In [3] Kaiser developed a formula to estimate the filter order with respect to the transition bandwidth as well as passband and stopband ripples as

$$N \cong \frac{-20log\sqrt{\delta_p \delta_s} - 13}{14.6(\frac{\omega_s - \omega_p}{2\pi})}$$

where $\omega_p$and ω_s are normalized passband and stopband edges, respectively, and δ_p and δ_s are peak passband and stopband ripples, respectively.

From the above formula, it follows that the filter length, hence the complexity of designing a digital FIR filter, becomes quite high when the filter is required to possess very narrow transition bandwidths and/or small passband and stopband ripples. In [4], Lim provides a plot (see Figure 3.1) of the filter length verses transition bandwidth for a minimax optimum low-pass filter with 0.2 dB peak-to-peak passband ripple and –40dB stopband ripple. We see that the complexity becomes prohibitively high for sharp FIR filters: a normalized transition band of 0.01π, for example, requires a filter length that exceeds three hundred.

FRM FIR filters [4] that were proposed during 1980s with an aim to reduce the extreme computational complexity that accompanies conventional linear-phase FIR filters.

3.3 Frequency-Response Masking (FRM) FIR Filters

3.3.1 Structure and frequency response of FRM FIR filters

A basic FRM filter [4] consists of four modules that are connected in parallel as shown in Fig. 3.2. It involves a linear-phase prototype filter $H_a(z)$ that up-samples input signal by M, a pair of linear-phase masking filters $\{H_{ma}(z), H_{mc}(z)\}$, and a delay line. All three sub-filters involved, namely, $H_a(z)$, $H_{ma}(z)$, and $H_{mc}(z)$ are linear-phase FIR filters of length N, N_a, and N_c, respectively.

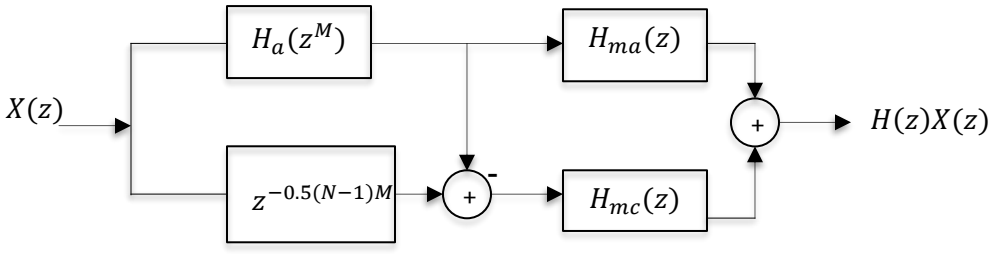


Fig. 3.2. The structure of a filter synthesized using frequency-response masking filters

Following Fig. 3.2, the transfer function of the FRM filter is obtained as

$$H(z) = H_a(z^M) \cdot H_{ma}(z) + H_c(z^M) \cdot H_{mc}(z)$$

where

$$H_c(z^M) = z^{-M((N-1)/2)} - H_a(z^M) \tag{3.1}$$

Hence

$$H(z) = H_a(z^M) \cdot H_{ma}(z) + [z^{-M((N-1)/2)} - H_a(z^M)] \cdot H_{mc}(z) \tag{3.2a}$$

where

$$H_a(z^M) = \sum_{k=0}^{N-1} h_k z^{-kM} \tag{3.2b}$$

$$H_{ma}(z) = \sum_{k=0}^{N_a-1} h_k^{(a)} z^{-k} \tag{3.2c}$$

$$H_{mc}(z) = \sum_{k=0}^{N_c-1} h_k^{(c)} z^{-k} \tag{3.2d}$$

Based on (3.1), the zero-phase frequency response of the FRM filter is given by

$$H(e^{j\omega}) = H_1(e^{j\omega}) + H_2(e^{j\omega}) \tag{3.3a}$$

where

$$H_1(e^{j\omega}) = H_a(e^{jM\omega}) \cdot H_{ma}(e^{j\omega}) \tag{3.3b}$$

and

$$H_2(e^{j\omega}) = [1 - H_a(e^{jM\omega})] \cdot H_{mc}(e^{j\omega}) \tag{3.3c}$$

Two linear-phase filters are said to be a *complementary pair* if the amplitude of the sum of the zero-phase frequency responses of the two filters is identically equal to unity. From (3.1) and (3.3), it is evident that filters $H_a(z)$ and $H_c(z)$ form a complementary pair because $|H_a(e^{j\omega}) + H_c(e^{j\omega})| = 1$.

3.3.2 The role of complementary pair $\{H_a(z), H_c(z)\}$

The frequency response of the prototype filter $H_a(z)$ can be expressed as

$$H_a(e^{j\omega}) = e^{-j\left(\frac{(N-1)}{2}\right)\omega} A(\omega)$$

where $A(\omega)$ is a trigonometric function of ω[10], [11]. Consider a typical case where $H_a(z)$ is a lowpass filter whose zero-phase frequency response is illustrated in Fig.3.3(a) where θ and ϕ are passband and stopband edges, respectively. The zero-phase frequency response of $H_c(z)$, which is known to be complementary to $H_a(z)$, is depicted in Fig. 3.3(b).

We stress that in an FRM filter down-sampling by-*M* version of the complementary pair is used, namely $\{H_a(z^M), H_c(z^M)\}$, whose frequency responses are

$\{H_a(e^{jM\omega}), H_c(e^{jM\omega})\}$. Typically, integer M here is considerably greater than unity, and so over the normalized

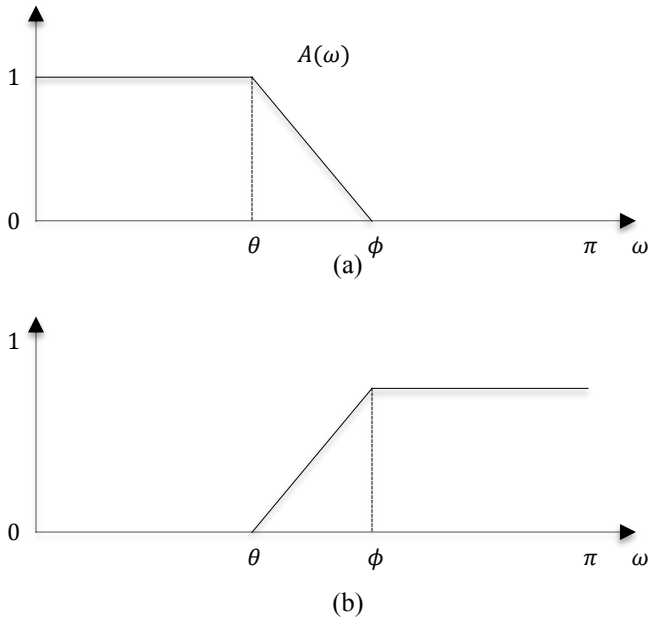


Fig. 3.3. Zero-phase frequency response of (a) $H_a(z)$ and (b) $H_c(z)$.

baseband $[-\pi, \pi]$, the frequency responses $H_a(e^{jM\omega})$ and $H_c(e^{jM\omega}) = 1 - H_a(e^{jM\omega})$ in this case may be thought as a total of M "squeezed (that is M times narrower)" copies of $H_a(e^{j\omega})$ and $1 - H_a(e^{j\omega})$, respectively. As a result, each individual copy of these squeezed copy possesses shorter passband/stopband and *sharper* transition band, see the first two subplots in Fig. 3.4 and Fig. 3.5 for illustration of this matter.

3.3.3 The role of masking filters $H_{ma}(z)$ and $H_{mc}(z)$

Figs. 3.4 and 3.5 illustrate the cases of broad-passband and narrow-passband lowpass FRM filters, respectively. In the case of designing a broad-passband lowpass FRM filter (see Fig. 3.4), the passband and stopband edges are given by [9]

$$\omega_P = \frac{2\pi m + \theta}{M} \tag{3.4a}$$

and

$$\omega_s = \frac{2\pi m + \phi}{M} \tag{3.4b}$$

respectively, where m is an integer less than M. From (3.4), it follows that the transition

band of the FRM filter is equal to $(\phi-\theta)/M$ which is M times narrower than that of the prototype lowpass filter $H_a(z)$. By using lowpass masking filters $H_{ma}(z)$ and $H_{mc}(z)$ with appropriate passbands (that are shown in plots (a) and (b) of Fig. 3.4 as dashed lines), we see that only first section of the complementary pair $H_a(e^{j\omega})$ and $1-H_a(e^{j\omega})$ survives and, as shown in plots (c) and (d) of Fig. 3.4, when the two channels add up, a lowpass filter with flat and broad passband and sharp transition band is produced. We remark that the group delay of masking filters $H_{ma}(z)$ and that of $H_{mc}(z)$, must be equal in order for them to perform frequency-response masking effectively. This means that the length of H_{ma} and H_{mc}, must either be both odd or both even and, if necessary leading delays must be added to either H_{ma} or H_{mc}, to equalize their group delays. Also note that in order to avoid half sample delay $(N-1)M$ must be even.

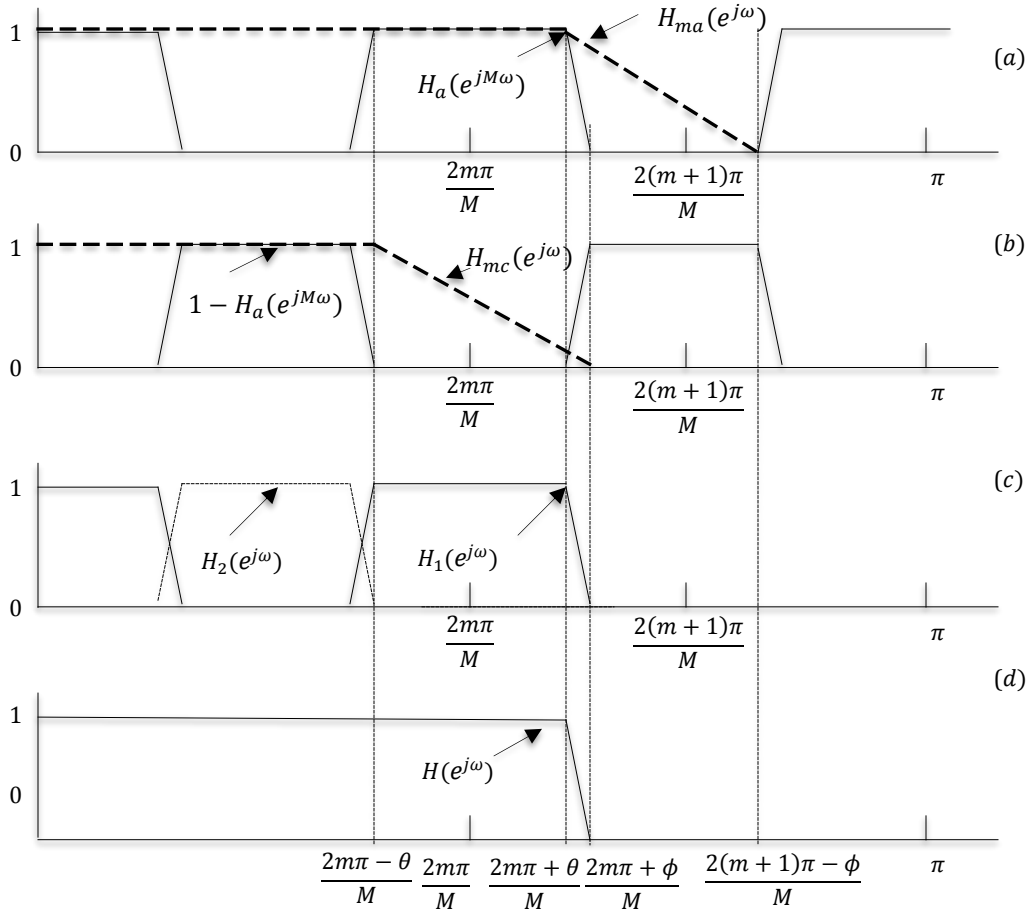


Fig. 3.4. Low-pass FRM filter with broad passband.

As Fig. 3.5 shows, FRM structure can also be utilized to construct a lowpass filter with relatively narrow passband and a sharp transition band. This is achieved by using a lowpass prototype filter with narrow passband and, at the same time, employing lowpass masking filters with appropriate passbands. In this case, the passband and stopband edges, denoted again by ω_P and ω_s, are given by [9]

$$\omega_P = \frac{2\pi m - \phi}{M} \tag{3.5a}$$

and

$$\omega_s = \frac{2\pi m - \theta}{M} \tag{3.5b}$$

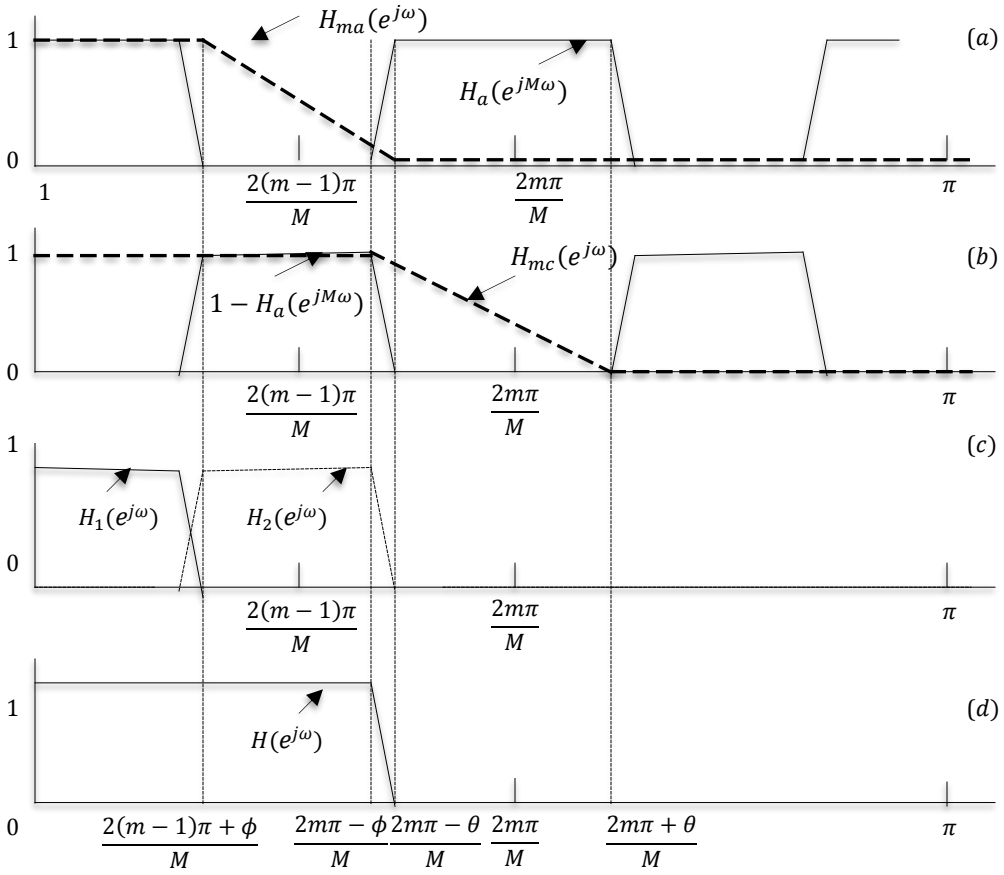


Fig. 3.5. A lowpass FRM filter with narrow passband.

3.3.4 Parameters for the design of an FRM filter

For the design of an FRM filter, the passband ω_P and stopband ω_s are given, and parameters $m, M, \theta,$ and ϕ must be determined. Among other things, we wish to choose M such that the overall complexity of the filter is minimized with respect to some criterion. Under these circumstances, it is desirable to express θ, ϕ, and m in terms of ω_P, ω_s and M. Note that

$$0 < \theta < \phi < \pi \tag{3.6}$$

To ensure that (3.4a) and (3.4b) yield a solution with $0 < \theta < \phi$, we have

$$m = \lfloor M\omega_P/(2\pi) \rfloor \tag{3.7a}$$

$$\theta = M\omega_p - 2m\pi \tag{3.7b}$$

$$\phi = M\omega_s - 2m\pi \tag{3.7c}$$

where $\lfloor M\omega_P/(2\pi) \rfloor$ is the largest integer that is less than or equal to $M\omega_P/(2\pi)$. To ensure that (3.5a) and (3.5b) yield a solution with $0 < \theta < \phi$, we have

$$m = \lceil M\omega_s/(2\pi) \rceil \tag{3.8a}$$

$$\theta = 2m\pi - M\omega_s \tag{3.8b}$$

$$\phi = 2m\pi - M\omega_p \tag{3.8c}$$

where $\lceil M\omega_s/(2\pi) \rceil$ is the smallest integer that is greater than or equal to $M\omega_s/(2\pi)$. It can be verified that for a given set of ω_P, ω_s and M, only one set of the equations, either from (3.7) or (3.8) (but not both), yields θ and ϕ satisfying the constraint $\theta < \phi$.

Since the transition bandwidth of $H_a(e^{j\omega})$ is equal to $M(\omega_s - \omega_P)$, for a given ω_P and ω_s , the transition bandwidth of $H_a(e^{j\omega})$ increases as factor M increases. Consequently the complexity of $H_a(z)$ decreases as M increases.

In addition, from Fig. 3.4, we observe that the transition bandwidths of $H_{mc}(e^{j\omega})$ and $H_{mc}(e^{j\omega})$ are equal to $(2\pi - \phi - \theta)/M$ and $(\phi + \theta)/M$, respectively. Consequently, the sum of the two transition bandwidths is equal to $2\pi/M$ which decreases as M increseases. The last statement also holds true for the narrow passband case illustrated in Fig. 3.5.

3.3.5 Original FRM filter design technique

The technique proposed originally in 1980s for the design of broadband as well as narrowband FRM filters with passband and stopband ripples δ_p and δ_s is a two-step procedure which can be outlined as follows [9].

Step 1. Design the masking filters $H_{ma}(z)$ and $H_{mc}(z)$ such that their zero-phase frequency responses approximate unity in their passbands with tolerance less than or equal to $0.9\delta_p$ and zero in their stopbands with tolerance less than or equal to $0.9\delta_s$.

Step 2. Design $1 - H_a(e^{jM\omega})$ such that the overall response $H(e^{j\omega})$, as given by Eqs. (3.3a)– (3.3c), approximates unity with tolerance less than or equal to $\delta_p$on the following passband (see Figs. 3.4 and 3.5)

$$\Omega_p = \begin{cases} \left[\frac{2m\pi - \theta}{M}, \frac{2m\pi + \theta}{M}\right] & \text{for broad passband} \\ \left[\frac{2(m-1)\pi + \phi}{M}, \frac{2m\pi - \phi}{M}\right] & \text{for narrow passband} \end{cases} \tag{3.9a}$$

and approximates zero with tolerance less than or equal to δ_s on the following stopband

$$\Omega_s = \begin{cases} \left[\frac{2m\pi + \phi}{M}, \frac{2(m+1)\pi - \phi}{M}\right] & \text{for broad passband} \\ \left[\frac{2m\pi - \theta}{M}, \frac{2m\pi + \theta}{M}\right] & \text{for narrow passband} \end{cases} \tag{3.9b}$$

Specifically, the design involved in Step 1 can be accomplished by using the Remez multiple exchange algorithm [12]. The design of $H_a(e^{j\omega})$ in Step 2 can be performed either using linear programming [13] or with the aid of the Remez algorithm. The order of $H_{ma}(z)$ [$H_{mc}(z)$] can be considerably reduced by allowing larger ripples on those regions of $H_{ma}(z)$ [$H_{mc}(z)$], where $H_a(e^{jM\omega})$ has one of its stop-bands [one of its pass-bands]. As a rule of thumb, the ripples on these regions can be selected to be ten times larger [4].

Chapter 4

Convex-Concave Procedure

4.1 Introduction

The convex-concave procedure (CCP), also known as concave-convex procedure [14] is a majorization-minimization algorithm [15] that is often used to find local solutions to difference of convex (DC) programming problems.

CCP is applicable to functions that are in the form of difference of two convex functions. The wide applicablity of CCP is due to the fact [14] that any function, subject to mild differentiablity conditions, can be expressed as sum of a convex function and a concave function, or equivalently difference of two convex functions, although such decomposition is not unique.

Let $F(\boldsymbol{x})$ be a smooth function of interest with bounded Hessian $\nabla^2 F(\boldsymbol{x})$. If $F(\boldsymbol{x})$ is convex, then global minimizer(s) of $F(\boldsymbol{x})$ exist and can be found using fast and reliable convex programming (CP) solvers. However, if $F(\boldsymbol{x})$ is nonconvex, developing fast optimization algorithms that are guaranteed to converge to a local optimum remains a challenge. Following [14], we consider a convex-concave decomposition of a smooth $F(\boldsymbol{x})$ as

$$F(\boldsymbol{x}) = F_{vex}(\boldsymbol{x}) + F_{cav}(\boldsymbol{x})$$

where $F_{vex}(\boldsymbol{x})$ is convex and $F_{cav}(\boldsymbol{x})$ is concave. Obviously, this decomposition can also be written as

$$F(\boldsymbol{x}) = F_{vex1}(\boldsymbol{x}) - F_{vex2}(\boldsymbol{x}) \tag{4.1}$$

where both $F_{vex1}(\boldsymbol{x}) = F_{vex}(\boldsymbol{x})$ and $F_{vex2}(\boldsymbol{x}) = -F_{cav}(\boldsymbol{x})$ are convex. A CCP-based algorithm for minimizing a smooth objective function $F(\boldsymbol{x})$ is an iterative process in which the kth iterate $\boldsymbol{x}_k$ is updated to the $(k+1)th$ iterate $\boldsymbol{x}_{k+1}$ such that:

$$\nabla F_{vex1}(\boldsymbol{x}_{k+1}) = \nabla F_{vex2}(\boldsymbol{x}_k) \tag{4.2}$$

In other words, the iterate update is carried out by matching the gradient of the convex component function to the negative gradient of the concave component function at the preceding iterate [14]. To understand (4.2), let us assume that a sequence of points $\{\boldsymbol{x}_k, k = 0, 1, \ldots\}$ that satisfy (4.2) is produced and that $\boldsymbol{x}_k$ converges to point $\boldsymbol{x}^*$ as k goes to

infinity. As k approaches infinity and assume the gradients of $F_{vex1}(\boldsymbol{x})$ and $F_{vex2}(\boldsymbol{x})$ are continuous, then (4.2) implies that

$$\nabla F_{vex1}(\boldsymbol{x}^*) = \nabla F_{vex2}(\boldsymbol{x}^*)$$

which in conjunction with (4.1) yields $\nabla F(\boldsymbol{x}^*) = \mathbf{0}$. Therefore, $\boldsymbol{x}^*$ is a stationary point of $F(\boldsymbol{x})$ and hence has a potential of being a minimizer of $F(\boldsymbol{x})$.

Later in this chapter, we will present an explanation of (4.2) when a CCP-based algorithm for nonconvex optimization problems is concretely developed. The main reason to include this chapter in the project report is that the problem of designing an FRM filter is a nonconvex problem, and the CCP turns out to be an intuitively natural tool to tackle the design problem. In the rest of the chapter, we first give a brief overview of constrained optimization. We then present some technical details of how CCP is applied to solve general nonconvex problems.

4.2 Constrained optimization problems

A constrained optimization problem assumes the form

$$\text{minimize } f_0(\boldsymbol{x})$$

$$\text{subject to: } f_i(\boldsymbol{x}) \leq 0, i = 1, 2 \ldots, m \qquad (4.3)$$

The problem seeks to find a best possible choice of a vector (which may be viewed as a point) in R^n from a set of all candidate points that satisfy constraints $f_i(\boldsymbol{x}) \leq 0$ for i = 1, 2, …, m. [18]. In (4.3), vector $\boldsymbol{x} = (x_1, \cdots, x_n)$ represents variable of the optimization problem, the m constraints $\{f_i(\boldsymbol{x}) \leq 0, i = 1, 2, \ldots, m\}$ represent requirements or specifications that limit the candidate points to a region known as feasible region in R^n, and the objective value $f_0(\boldsymbol{x})$ represents the cost of choosing x. A solution of the optimization problem (4.3), known as a *minimizer* of the problem, is a vector $\boldsymbol{x}$ that has minimum cost among all candidate points in the feasible region.

There are various classes of optimization problems, each characterized by particular form of the objective and constraint functions. As an important example, the optimization problem (4.3) is called a *linear program* if the objective and constraint functions $f_0, \cdots, f_m$ are all affine functions of the form $\boldsymbol{a}^T\boldsymbol{x} + a_0$ where real-valued vector $\boldsymbol{a}$ and constant a_0 are given. If some of the functions involved in (4.3) are not linear, (4.3) is called a *nonlinear program*.

A solution method for a class of optimization problems is essentially an algorithm that computes a solution of the problem to some accuracy. Since late 1940s, a large effort has been devoted to developing algorithms for solving various classes of optimization problems, analyzing their properties, and developing software implementations. The

effectiveness of these algorithms, i.e., the ability to solve the optimization problem in (4.3), varies considerably, and depends on factors such as the particular forms of the objective and constraint functions, the number of variables and constraints involved, and problem structure.

The general optimization problem in (4.3), even when the objective and constraint functions are smooth (for example, polynomials), is surprisingly difficult to solve. Approaches to the general problems, therefore, often involve some kind of compromise, such as long computation time or the possibility of unable to find a satisfactory local solution. On the other hand, there are some important exceptions to the general rule that most optimization problems are difficult to solve. This is indeed the case when the objective is convex function and feasible region is a convex set. Problems of this sort are known as convex programming (CP) problems, which admit effective and reliable algorithms for fast solutions even when the problem size is large.

4.2.1 Convex optimization

A convex optimization [18] problem assumes the form

$$\text{minimize } f_0(\boldsymbol{x})$$

$$\text{subject to: } f_i(\boldsymbol{x}) \leq 0, i = 1, 2 \ldots, m \qquad (4.4)$$

where $f_i(\boldsymbol{x})$ for i = 0, 1, …, m are all convex function. Function $f(\boldsymbol{x})$ is said to be convex if

$$f(\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{y})$$

holds for all $\boldsymbol{x}, \boldsymbol{y} \in R^n$ and $0 \leq \alpha \leq 1$.

In general, there is no analytical formula to solve the convex optimization problems, but there are very effective methods for solving them, i.e. bundle methods, sub-gradient projection methods [16] and interior-point methods [17]. In some cases, CP problem can be solved to a given accuracy with a number of operations that does not exceed a polynomial of the problem dimensions. With only a bit of exaggeration, we can also say that if we can formulate a problem as a convex optimization problem, then we can solve it efficiently.

4.2.2 Nonconvex optimization

Nonconvex optimization is an optimization problem where the objective or constraint functions, or both are not convex. A *non-convex* function "curves up and down", it is neither convex nor concave. Familiar examples of nonconvex functions are the sinusoidal functions. Many practical problems of importance are nonconvex, and most nonconvex problems

are hard to solve exactly in a reasonable time. As nonconvex optimization is challenging in general, people have tried to simplify these problems and proposed algorithms to deal with them.

4.2.3 Local optimization

In local optimization, we seek to find a point to minimize the objective function among feasible points that are near it. Evidently, such a point can only be a local solution as it is not guaranteed to have a lower objective value than all other feasible points beyond a local region. A large fraction of the research on general nonlinear programming has focused on methods for local optimization, which as a consequence are now well developed.

Local optimization methods can be fast, can handle large-scale problems, and are widely applicable, since they only require differentiability of the objective and constraint functions. As a result, local optimization methods are widely used in applications where it is desirable to find a "good" point, if not the very best point. In an engineering design application, for example, local optimization can be used to improve the performance of a design originally obtained by manual or other design methods.

There are several disadvantages of utilizing local optimization methods, beyond (possibly) not finding globally optimal solution. Among other things, these methods require an initial guess for the optimization variables. Because of the nonconvex nature of the problems involved, making an initial guess (also known as starting point) is critical as it can greatly affect the objective value of the local solution obtained. Often times little information is available about how far the local solution is from (globally) optimal.

An interesting comparison can be made between local optimization methods for nonlinear programming and convex optimization. Since differentiability of objective and constraint functions is the only requirement for most local optimization methods, formulating a practical problem as a nonlinear optimization problem is relatively straightforward. Once the problem is formulated, the art in local optimization is in solving the problem (in the weakened sense of finding a locally optimal point). In convex optimization, the situation is opposite: the art and challenge lie in problem formulation, once a problem is formulated as a convex optimization problem, it is relatively straightforward to solve it and a good algorithm for CP problems shall converge to a global solution regardless of where it is initiated.

4.2.4 Global optimization

The methods for global optimization seek to find global solutions of the optimization problem with possible compromise in efficiency. In general the worst-case complexity of

global optimization methods grows exponentially with the problem sizes n and m; the hope is that in practice, for the particular problem instances encountered, the method is much faster relative to the worst-case complexity. While this favorable situation does occur, it is not typical. Even small problems with just a few tens of variables, it can take a very long time (e.g., hours or days) to solve [15].

Global optimization is used for problems with a small number of variables, where computing time is not critical, and the value of finding the true global solution is considered very high.

4.2.5 Role of convex optimization in non-convex problems

The convex optimization plays an important role in problems that are not convex [15].

(i) Initialization for local optimization

By reformulating the nonconvex problem to an approximate convex problem, we can solve this approximate problem, which can be done easily and without an initial guess. The solution to this approximate problem can be used as the starting point for finding the local optimization of this nonconvex problem.

(ii) Convex heuristics for nonconvex optimization

Convex optimization is the basis for several heuristics for solving non-convex problems. One interesting example is the problem of finding a sparse vector $\boldsymbol{x}$ that satisfies some constraints. While this is a difficult combinatorial problem, there are some simple heuristics, based on convex optimization, that often find fairly sparse solutions.

(iii) Bounds for global optimization

Many methods for global optimization require a cheaply computable lower bound on the optimal value of the nonconvex problem. Two standard methods for doing this are based on convex optimization. Using a relaxation technique, each nonconvex constraint is replaced with a looser, but convex, constraint. In Lagrangian relaxation, the Lagrangian dual problem is solved. This problem is convex, and provides a lower bound on the optimal value of the nonconvex problem.

4.3 Difference of Convex Programming

It can be shown that a nonconvex function can always be decomposed into sum of a convex function and a concave function or, equivalently, difference of two convex functions. This leads to the so-called *difference of convex functions programming* (DC programming) [19]. DC programming plays an important role in the field of non-convex optimization, because of the desirable theoretical properties of the method as well as its wide range of applications. It can be shown that the set of DC functions defined on a

compact convex set of R^n is dense in the set of continuous functions that are defined on the same convex set. Therefore, in principle, every continuous function can be approximated by a DC function with any desired precision.

Example: Consider the nonconvex function $f(x) = |x| + \log(x)$, which can be decomposed as the difference of $f_1(x) = |x|$ and $f_2(x) = -\log(x)$, where $f_1(x)$ and $f_2(x)$ are two convex functions as shown in the following figure.

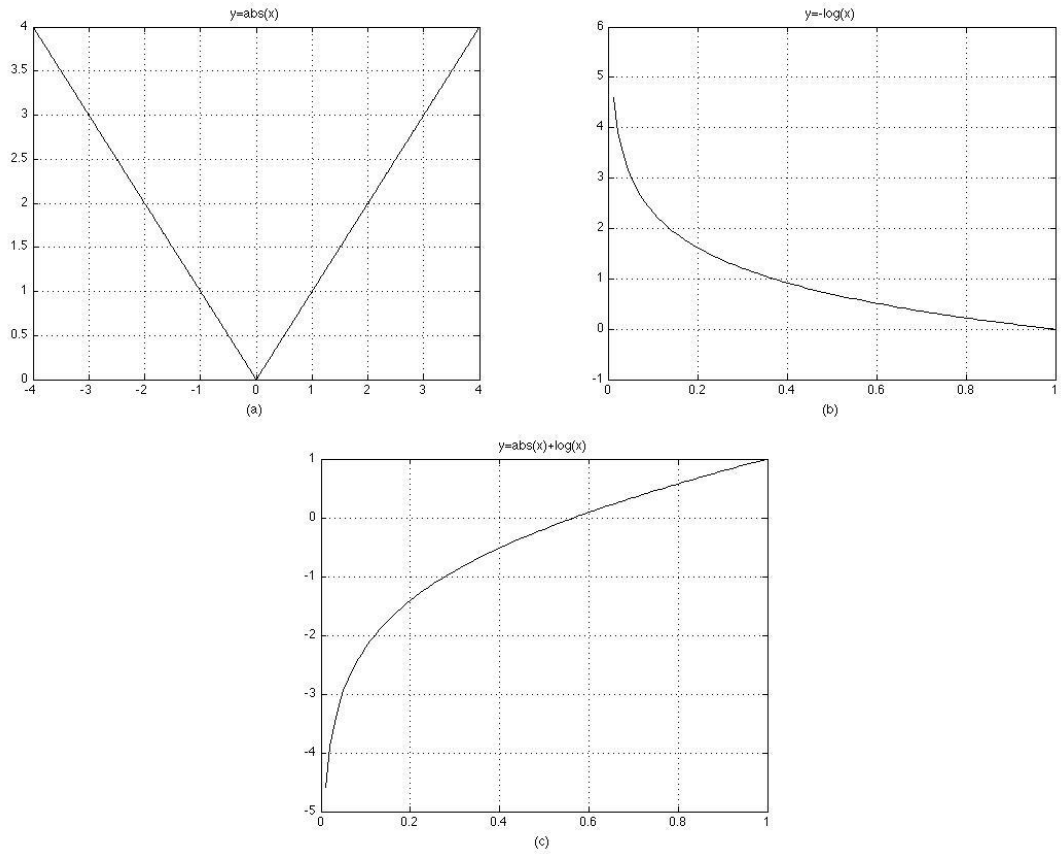


Fig. 4.1. Nonconvex function $f(x) = |x| + \log(x)$ is seen as difference of two convex functions.

4.3.1 DC functions

Definition:

Let *C* be a convex subset of R^n. A real-valued function $f: C \to R$ is called DC on C, if there exist two convex functions *g, h:* $C \to R$ such that f can be expressed in the form [19]

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) - h(\boldsymbol{x}) \tag{4.5}$$

And if $C = R^n$, then f is called a *DC function*. Each representation of the form (4. 5) is said to be a DC decomposition of *f.* We call a function $f: R^n \to R$ is *locally DC*, if for every $x_0 \in R^n$ there exists $\epsilon > 0$ such that f is DC in the ball

$$B(\boldsymbol{x}_0, \epsilon) = \{\boldsymbol{x} \in R^n : \|\boldsymbol{x} - \boldsymbol{x}_0\| \leq \epsilon\} \tag{4.6}$$

The following propositions show that the class of DC functions is closed under several operations encountered frequently in optimization [19].

Proposition 1

Let f and $f_i, i = 1, \ldots, m,$ be DC functions. Then the following functions are also DC:

- $\sum_{i=1}^{m} \lambda_i f_i, \lambda_i \in R \quad i = 1, \ldots, m,$
- $\max\limits_{i=1,\ldots,m} f_i(\boldsymbol{x})$, $\min\limits_{i=1,\ldots,m} f_i(\boldsymbol{x})$,
- $|f(\boldsymbol{x})|$, $f^+(x) = \max\{0, f(\boldsymbol{x})\}$, $f^-(\boldsymbol{x}) = \min\{0, f(\boldsymbol{x})\}$,
- $\prod_{i=1}^{m} f_i(\boldsymbol{x})$, where $\prod$ denote product.

It is worth noting that the above results can be proved constructively; i.e., for each of the functions defined above, an explicit DC decomposition can be constructed.

For example, consider the function

$$f(\boldsymbol{x}) = \max\{f_i(\boldsymbol{x}) = g_i(\boldsymbol{x}) - h_i(\boldsymbol{x}), i = 1, \ldots, m\} \tag{4.7}$$

with g_i and h_i being convex functions, $i = 1, \ldots, m.$ Since, for each $i = 1, \ldots, m,$

$$f_i = g_i + \sum_{\substack{j=1 \\ j \neq i}}^{m} h_j - \sum_{j=1}^{m} h_j \tag{4.8}$$

we obtain the following DC decomposition of f as

$$f = g - h \tag{4.9}$$

where

$$g = \max_{i=1,\ldots,m} \left\{ g_i + \sum_{\substack{j=1 \\ j \neq i}}^{m} h_j \right\} \text{ and } h = \sum_{j=1}^{m} h_j$$

Proposition 2: Every locally DC function is a DC function.

Proposition 3:

- Every function $f : R^n \to R$ whose second partial derivatives are continuous everywhere is a DC function.
- Let *C* be a compact convex subset of R^n. Then every continuous function on *C* is the limit of a sequence of DC functions which converge uniformly on *C,* namely,

for any continuous function c: $C \to R$ and for any $\epsilon > 0$, there exists a DC function f: $C \to R$ such that $|c(\boldsymbol{x}) - f(\boldsymbol{x})| < \epsilon, \forall x \in C$.

- Let f: $R^n \to R$ be DC, and let g: $R \to R$ be convex. Then the composite function $(g \circ f)(\boldsymbol{x}) = g(f(\boldsymbol{x}))$ is DC.

4.3.2 DC programming problems

The general form of DC programming problems is given by

$$\min\{f_0(\boldsymbol{x}): \boldsymbol{x} \in X, f_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m\} \tag{4.10}$$

where $f_i = g_i - h_i, i = 0, \dots, m,$ are DC functions and X is a closed convex subset of R^n. The related problem

$$\min\{c(\boldsymbol{x}): \boldsymbol{x} \in X, \psi(\boldsymbol{x}) \leq 0\} \tag{4.11}$$

where c is a linear function, $X \subset R^n$ is closed and convex, and $\psi(\boldsymbol{x})$ is a concave function, is often called a *canonical DC program*. Ever DC programming problem of the form (4.10) can be transformed into the canonical form (4.11) as follows.

By introducing an additional variable x_{n+1}, problem (4.10) can be rewritten as

$$\min\{x_{n+1}: \boldsymbol{x} \in X, x_{n+1} \in R, f_0(\boldsymbol{x}) - x_{n+1} \leq 0, f_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m\} \tag{4.12}$$

Next, we define a DC function f: $R^{n+1} \to R$ by

$$f(\boldsymbol{x}, x_{n+1}) = \max\{(f_0(\boldsymbol{x}) - x_{n+1}), f_1(\boldsymbol{x}), \dots, f_m(\boldsymbol{x})\} \tag{4.13}$$

and let $f = g - h$ be a DC decomposition determined as in (4.9). Finally, by introducing another additional variable x_{n+2}, and setting

$$\boldsymbol{z} = \{\boldsymbol{x}, x_{n+1}, x_{n+2}\} \in R^{n+2}$$

$$Z = \{\boldsymbol{z} \in R^{n+2}: \boldsymbol{x} \in X, g(\boldsymbol{x}, x_{n+1}) - x_{n+2} \leq 0\}$$

$$\psi(\boldsymbol{z}) = x_{n+2} - h(\boldsymbol{x}, x_{n+1}) \tag{4.14}$$

we obtain the canonical DC program

$$\min\{c(\boldsymbol{z}) = x_{n+2}: \boldsymbol{z} \in Z, \psi(\boldsymbol{z}) \leq 0\} \tag{4.15}$$

For the establishment of optimality conditions and for the development of solution methods for problem (4.11), in many cases, the linear function c can be replaced by a convex function. In what follows, we call problem (4.11) a generalized canonical DC program, if the function c is convex.

4.4 Basic CCP Algorithm

As illustrated in Section 4.1 , the convex-concave procedure can be applied to (almost) any optimization problems and many existing algorithms can be interpreted in terms of CCP. It is therefore natural to try to apply the CCP to find local solutions for DC programming problems.

In what follows, we consider DC programming problem of the form

$$\begin{aligned} &\text{minimize } f_0(\boldsymbol{x}) - g_0(\boldsymbol{x}) \\ &\text{subject to: } f_i(\boldsymbol{x}) - g_i(\boldsymbol{x}) \leq 0, i = 1, \ldots, m \end{aligned} \tag{4.16}$$

where $\boldsymbol{x} \in R^n$ is the optimization variable, $f_i: R^n \to R$ and $g_i: R^n \to R$ for $i = 0, \ldots, m$ are convex. The first thing to note is that a DC program is *not* convex unless all the functions g_i are affine, and is hard to solve in general. To see this, we can cast the Boolean linear program (LP)

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{c}^T \boldsymbol{x} \\ &\text{subject to:} \quad x_i \in \{0,1\}, \mathrm{i} = 1, \ldots, n \\ &\qquad\qquad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \end{aligned} \tag{4.17}$$

where $\boldsymbol{x} \in R^n$ is the optimization variable and $\boldsymbol{c} \in R^n, \boldsymbol{A} \in R^{m \times n}$, and $\boldsymbol{b} \in R^m$ are problem data, as a DC program in (4.16) as

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{c}^T \boldsymbol{x} \\ &\text{subject to: } x_i^2 - x_i \leq 0, \ i = 1, \ldots, n \\ &\qquad\qquad x_i - x_i^2 \leq 0, \ i = 1, \ldots, n \\ &\qquad\qquad \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \leq \boldsymbol{0} \end{aligned} \tag{4.18}$$

In (4.18), the objective and the 1st set and 3rd constraint functions are convex, but the second set of n inequality constraint functions are nonconvex (actually they are concave). Thus the Boolean LP problem (4.17) is a DC program. We mention that the Boolean LP, in turn, is a representative from the many problems that are thought to be hard to solve, like the traveling salesman problem, etc., for which no polynomial-time algorithms are available, and it is widely believed such algorithms do not exist.

It is known that the global solution to (4.17) can be found through general branch-and-bound methods. From above analysis, it follows that alternatively one may attempt to find a local solution to this problem through the techniques for nonlinear optimization, including CCP.

We now present the basic convex-concave procedure, also known as the concave-convex procedure. This is a heuristic for finding a local optimum of problem (4.16) that leverages the ability to efficiently solve convex optimization problems.

We assume that all f_i and g_i are differentiable for the ease of technical treatment, but the analysis below also holds for non-smooth functions where the gradient at a point is replaced by a *subgradient* at that point. This basic version of the algorithm requires an initial feasible point $\boldsymbol{x}_0$, i. e., $f_i(\boldsymbol{x}_0) - g_i(\boldsymbol{x}_0) \leq 0, for\ i = 1, \ldots, m.$

Algorithm 1: Basic CCP algorithm.

Given an initial feasible point $\boldsymbol{x}_0$. Set $k := 0$.

repeat

1. **Convexify**. Form $\hat{g}_i(\boldsymbol{x}; \boldsymbol{x}_k) \triangleq g_i(\boldsymbol{x}_k) + \nabla g_i(\boldsymbol{x}_k)^T(\boldsymbol{x} - \boldsymbol{x}_k)$ for $i = 0, \ldots, m.$

2. **Solve**. Set the value of x_{k+1} to a solution of the convex problem

$$\text{minimize } f_0(\boldsymbol{x}) - \hat{g}_0(\boldsymbol{x}; \boldsymbol{x}_k)$$

$$\text{subject to: } f_i(\boldsymbol{x}) - \hat{g}_i(\boldsymbol{x}; \boldsymbol{x}_k) \leq 0, \quad i = 1, \ldots, m.$$

3. **Update iteration**. $k := k + 1$.

until a stopping criterion is satisfied.

A reasonable stopping criterion is that the improvement in the objective value is less than some threshold δ, i.e.,

$$(f_0(\boldsymbol{x}_k) - g_0(\boldsymbol{x}_k)) - (f_0(\boldsymbol{x}_{k+1}) - g_0(\boldsymbol{x}_{k+1})) \leq \delta \qquad (4.19)$$

Later we will show that the left-hand side of (4.19) is always nonnegative. It is important to note that that the subproblem in step 2 of algorithm 1,

$$\text{minimize } f_0(\boldsymbol{x}) - (g_0(\boldsymbol{x}_k) + \nabla g_0(\boldsymbol{x}_k)^T(\boldsymbol{x} - \boldsymbol{x}_k))$$

$$\text{subject to: } f_i(\boldsymbol{x}) - (g_i(\boldsymbol{x}_k) + \nabla g_i(\boldsymbol{x}_k)^T(\boldsymbol{x} - \boldsymbol{x}_k)) \leq 0, i = 1, \ldots, m \qquad (4.20)$$

is a *convex* problem because both the objective and constraint functions are convex, and can therefore be solved efficiently.

We are now in a position to explain Eq. (4.2). To this end, suppose we are given an objective function $F(\boldsymbol{x})$ for minimization without constraint. To apply CCP to the problem at hand, first we use (4.1) to write $F(\boldsymbol{x}) = F_{\text{vex1}}(\boldsymbol{x}) - F_{\text{vex2}}(\boldsymbol{x})$ and convexify the objective function at a point $\boldsymbol{x}_k$ to

$$\hat{F}(\boldsymbol{x}) = F_{\text{vex1}}(\boldsymbol{x}) - (F_{\text{vex2}}(\boldsymbol{x}_k) + \nabla F_{\text{vex2}}(\boldsymbol{x}_k)^T(\boldsymbol{x} - \boldsymbol{x}_k))$$

According to CCP, the next iterate $\boldsymbol{x}_{k+1}$ is obtained by minimizing $\hat{F}(\boldsymbol{x})$. Since function $\hat{F}(\boldsymbol{x})$ is convex, we have $\nabla\hat{F}(\boldsymbol{x}_{k+1}) = \boldsymbol{0}$. This in conjunction with the above expression of $\hat{F}(\boldsymbol{x})$ gives $\nabla\hat{F}(\boldsymbol{x}_{k+1}) = \nabla F_{\text{vex1}}(\boldsymbol{x}_{k+1}) - \nabla F_{\text{vex2}}(\boldsymbol{x}_k) = \boldsymbol{0}$, i.e., $\nabla F_{\text{vex1}}(\boldsymbol{x}_{k+1}) = \nabla F_{\text{vex2}}(\boldsymbol{x}_k)$ which verifies Eq. (4.2).

We now follow [14] to make a remark on initialization. It is important to stress that CCP is a *local heuristic*. As such the final point found often depend on the initial point $\boldsymbol{x}_0$, and it is typically advantageous to initialize the algorithm with several randomly chosen (feasible) $\boldsymbol{x}_0$ and take as the final choice of $\boldsymbol{x}_0$ the final point found with the lowest objective value over the different runs.

As is shown in [21], at the iterates obtained by solving the convex suproblem in step 2 of the original objective algorithm (see (4.16)) is monotonically non-increasing, i.e.,

$$f_0(\boldsymbol{x}_{k+1}) - g_0(\boldsymbol{x}_{k+1}) \leq f_0(\boldsymbol{x}_k) - g_0(\boldsymbol{x}_k) \tag{4.21}$$

Suppose $\boldsymbol{x}_k$ is a feasible point for (4.16). We know that $\boldsymbol{x}_k$ is also a feasible point for the convexified subproblem (4.20) because

$$f_i(\boldsymbol{x}_k) - g_i(\boldsymbol{x}_k; \boldsymbol{x}_k) \leq f_i(\boldsymbol{x}_k) - g_i(\boldsymbol{x}_k) \leq 0 \tag{4.22}$$

This means that the feasible region for the convex subproblem in step 2 is non-empty as long as the feasible region for the original DC problem in (4.16) is non-empty. The convexity of g_i gives us $\widehat{g_i}(\boldsymbol{x}; \boldsymbol{x}_k) \leq g_i(\boldsymbol{x})$, for all $\boldsymbol{x}$, so

$$f_i(\boldsymbol{x}) - g_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}) - \widehat{g_i}(\boldsymbol{x}; \boldsymbol{x}_k) \tag{4.23}$$

It then follows that $\boldsymbol{x}_{k+1}$ must be a feasible point of (4.16) since

$$f_i(\boldsymbol{x}_{k+1}) - g_i(\boldsymbol{x}_{k+1}) \leq f_i(\boldsymbol{x}_{k+1}) - \widehat{g_i}(\boldsymbol{x}_{k+1}; \boldsymbol{x}_k) \leq 0 \tag{4.24}$$

Thus, because $\boldsymbol{x}_0$ was chosen feasible, all iterates are feasible for the original DC problem.

We will now show that at the iterate generated by CCP, the objective value converges [22].

Let $v_k = f_0(\boldsymbol{x}_k) - g_0(\boldsymbol{x}_k)$. We can write

$$v_k = f_0(\boldsymbol{x}_k) - g_0(\boldsymbol{x}_k) = f_0(\boldsymbol{x}_k) - \widehat{g_0}(\boldsymbol{x}_k; \boldsymbol{x}_k) \geq f_0(\boldsymbol{x}_{k+1}) - \widehat{g_0}(\boldsymbol{x}_{k+1}; \boldsymbol{x}_k) \tag{4.25}$$

where the last inequality follows because at each iteration k we minimize the value of $f_0(\boldsymbol{x}) - \widehat{g_0}(\boldsymbol{x}; \boldsymbol{x}_k)$. Thus

$$v_k \geq f_0(\boldsymbol{x}_{k+1}) - \widehat{g_0}(\boldsymbol{x}_{k+1}; \boldsymbol{x}_k) \geq v_{k+1} \tag{4.26}$$

Thus the sequence $\{v_i\}_{i=0}^{\infty}$ is non-increasing and shall converge, possible to negative infinity. A proof showing the convergence of iterates $\{\boldsymbol{x}_k, k = 0, 1, \ldots\}$ to a critical point of the original problem can be found in [22].

We remark that although the objective value is shown to converge, the iterates $\{\boldsymbol{x}_k, k = 0, 1, \ldots\}$ do not necessarily converge to a local minimizer. Consider for example the minimization problem

$$\text{minimize } 2x^4 - 3x^2,$$

where $x \in \boldsymbol{R}$ is the variable, which has optimal value -1.125 at $x = \mp\frac{\sqrt{3}}{2}$. If the algorithm is initialized with $x_0 = 0$, then the algorithm will converge in one step to the local maxmizer $x = 0$, see Fig. 4.3.

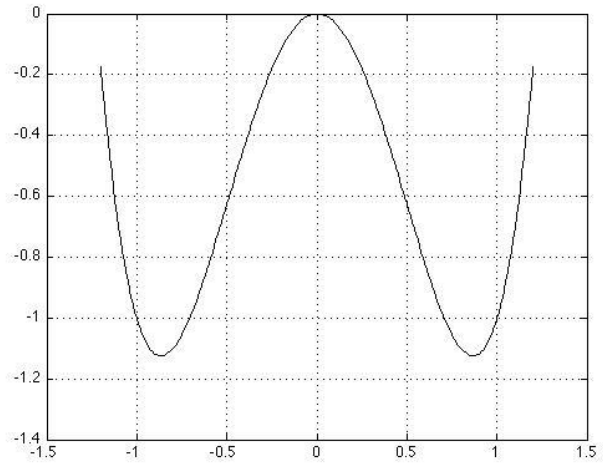


Fig 4.2: CCP fails to work for function $f(x) = 2x^4 - 3x^2$ if it starts at $x_0 = 0$.

Chapter 5

CCP Based FRM Filters Design

This chapter is devoted to the design of FRM filters where all sub-filters are assumed to have linear-phase responses, and the lengths of the masking filters are either both even or both odd. Our method to achieve the design goal will be based on CCP.

5.1 Basic Structure of the FRM Filter

As can be seen from Fig. 5.1, both the prototype filter $H_a(z)$ and its complement $H_c(z) = z^{-0.5(N-1)} - H_a(z)$ are up-sampled by M, yielding sparse filter coefficients and a reduced transition width for each of them. They are then connected in cascade with a pair of frequency-response masking filters in order to approximate a desired sharp frequency response.

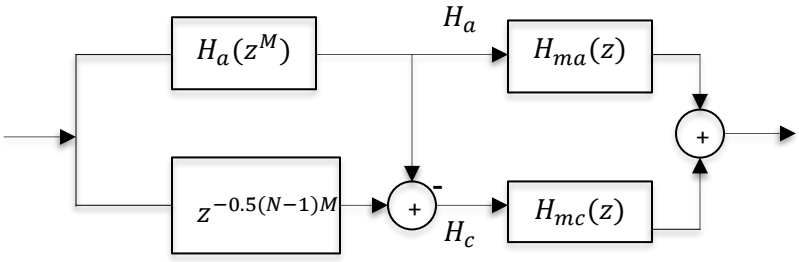


Fig 5.1. Basic realization structure of an FRM filter.

The transfer functions of the sub-filters in a basic FRM filter are given by

$$H_a(z) = \sum_{k=0}^{N-1} h_k z^{-k} \tag{5.1}$$

$$H_{ma}(z) = \sum_{k=0}^{N_a-1} h_k^{(a)} z^{-k} \tag{5.2}$$

$$H_{mc}(z) = \sum_{k=0}^{N_c-1} h_k^{(c)} z^{-k} \tag{5.3}$$

If all sub-filters have linear phase responses with lengths N_a and N_c either both even or both odd, then the FRM filter has a linear phase response with the group delay

$$D = 0.5(N-1)M + \max\{0.5(N_a-1), 0.5(N_c-1)\} \tag{5.4}$$

Now let us consider the design of a lowpass filter with sampling factor M, normalized passband edge ω_p and stopband edge ω_a, a reasonable *initial* design of sub-filters $H_a(z), H_{ma}(z)$ and $H_{mc}(z)$ can be obtained by using a standard method [20]. To design a lowpass FRM filter, the three sub-filters have to be lowpass filters whose passband and stopband edges are determined as follows [2]

1) For $H_a(z)$, the passband edge θ and stopband edge ϕ are given by

$$\theta = \omega_p M - 2m\pi \tag{5.5a}$$

$$\phi = \omega_s M - 2m\pi \tag{5.5b}$$

$$m = \lfloor \omega_s M/2\pi \rfloor \tag{5.5c}$$

where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x, or by

$$\theta = 2m\pi - \omega_s \pi \tag{5.6a}$$

$$\phi = 2m\pi - \omega_p \pi \tag{5.6b}$$

$$m = \lceil \omega_s M/2\pi \rceil \tag{5.6c}$$

where $\lceil x \rceil$ denotes the smallest integer larger than or equal to x, depending on which set of $\{\theta, \phi\}$ satisfies $0 < \theta < \phi < \pi$.

2) If (5.5) is used to determine the values of θ and ϕ, then the passband and stopband edges of $H_{ma}(z)$ are given by $(2m\pi + \theta)/M$ and $[2(m+1)\pi - \phi)]/M$, respectively, and the passband and stopband edges of $H_{mc}(z)$ are given by $(2m\pi - \theta)/M$ and $(2m\pi + \phi)/M$, respectively.

3) If (5.6) is used to determine the values of θ and ϕ, then the passband and stopband edges of $H_{ma}(z)$ are given by $[2(m-1)\pi - \phi)]/M$ and $(2m\pi - \theta)/M$, respectively, and the passband and stopband edges of $H_{mc}(z)$ are given by $(2m\pi - \phi)/M$ and $(2m\pi + \theta)/M$, respectively.

5.2 Optimal Design of FRM Filters Using CCP

5.2.1 Frequency Response and Its Gradient

The reader is referred to Eqs. (5.1) – (5.3) and Fig. 5.1 for the structure of an FRM. Since every compoment filters in an FRM has a linear-phase response, the design of a linear-phase FRM can be carried out without a concern with its phase response. Consequently, the FRM filter can be treated as a zero-phase FIR filter, without loss of generality. Thus the frequency response of the FRM filter is given by

$$H(\omega, \boldsymbol{x}) = [\boldsymbol{a}^T \boldsymbol{c}(\omega)][\boldsymbol{a}_a^T \boldsymbol{c}_a(\omega) - \boldsymbol{a}_c^T \boldsymbol{c}_c(\omega)] + \boldsymbol{a}_c^T \boldsymbol{c}_c(\omega) \tag{5.7}$$

where

$$\boldsymbol{a} = \begin{cases} [h_{\frac{N-1}{2}}\ 2h_{\frac{N+1}{2}}\ \cdots\ 2h_{N-1}]^T, & \text{for odd } N \\ 2[h_{N/2}\ \cdots\ h_{N-1}]^T, & \text{for even } N \end{cases} \tag{5.8}$$

$$\boldsymbol{c}(\omega) = \begin{cases} [1 \cos M\omega\ \ \cdots \cos[(N-1)M\omega/2]]^T, & \text{for odd } N \\ [\cos\left(\frac{M\omega}{2}\right) \cdots \cos[(N-1)M\omega/2]]^T, & \text{for even } N \end{cases} \tag{5.9}$$

$$\boldsymbol{a}_a = \begin{cases} [h^a_{\frac{N_a-1}{2}}\ 2h^a_{\frac{N_a+1}{2}}\ \cdots\ 2h^a_{N_a-1}]^T, & \text{for odd } N_a \\ 2[h^a_{\frac{N_a}{2}}\ \cdots\ h^a_{N_a-1}]^T, & \text{for even } N_a \end{cases} \tag{5.10}$$

$$\boldsymbol{c}_a(\omega) = \begin{cases} [1 \cos\omega\ \cdots \cos[(N_a-1)\omega/2]]^T, & \text{for odd } N_a \\ [\cos\left(\frac{\omega}{2}\right) \cdots \cos[N_a-1)\omega/2]]^T, & \text{for even } N_a \end{cases} \tag{5.11}$$

$$\boldsymbol{a}_c = \begin{cases} [h^c_{\frac{N_c-1}{2}}\ 2h^c_{\frac{N_c+1}{2}}\ \cdots\ 2h^c_{N_c-1}]^T, & \text{for odd } N_c \\ 2[h^c_{\frac{N_c}{2}}\ \cdots\ h^c_{N_c-1}]^T, & \text{for even } N_c \end{cases} \tag{5.12}$$

$$\boldsymbol{c}_c(\omega) = \begin{cases} [1 \cos\omega\ \cdots \cos[(N_c-1)\omega/2]]^T, & \text{for odd } N_c \\ [\cos\left(\frac{w}{2}\right) \cdots \cos[N_c-1)\omega/2]]^T, & \text{for even } N_c \end{cases} \tag{5.13}$$

In (5.7), vector $\boldsymbol{x}$ collects all design variables in the form

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{a}_a \\ \boldsymbol{a}_c \end{bmatrix} \tag{5.14}$$

The group delay of the FRM filter is given by

$$D = \frac{(N-1)M}{2} + d \tag{5.15}$$

where

$$d = \max\left(\frac{(N_a-1)}{2}, \frac{(N_c-1)}{2}\right)$$

Therefore, group delay D is a constant for given filter lengths and sampling factor M. Using (5.7) to (5.14), the gradient of $H(\omega, \boldsymbol{x})$ with respect to $\boldsymbol{x}$ can be evaluated, and is found to be

$$\boldsymbol{g}(\omega, \boldsymbol{x}) = \begin{bmatrix} y(\omega)\boldsymbol{c}(\omega) \\ [\boldsymbol{a}^T\boldsymbol{c}(\omega)]\boldsymbol{c}_a(\omega) \\ [1 - \boldsymbol{a}^T\boldsymbol{c}(\omega)]\boldsymbol{c}_c(\omega) \end{bmatrix} \tag{5.16}$$

where

$$y(\omega) = \boldsymbol{a}_a^T \boldsymbol{c}_a(\omega) - \boldsymbol{a}_c^T \boldsymbol{c}_c(\omega).$$

5.2.2 Desired frequency response and weighting function

For the sake of presentation clarity, we consider the case of designing a lowpass FRM filter with up-sampling factor M, normalized passband edge ω_p and stopband edge ω_s. The desired $H_d(\omega)$ in this case becomes

$$H_d(\omega) = \begin{cases} 1, & \text{for } 0 \le \omega \le \omega_p \\ 0, & \text{for } \omega_s \le \omega \le \pi \end{cases} \tag{5.17}$$

In our design formulation, a piece-wise constant weighting function

$$W(\omega) = \begin{cases} 1, & \text{for } 0 \le \omega \le \omega_p \\ w, & \text{for } \omega_s \le \omega \le \pi \\ 0, & \text{elsewhere} \end{cases} \tag{5.18}$$

will be utilized to emphasize or deemphasize certain frequency regions, where w is a positive scalar to weigh the importance of the stopband relative to the passband.

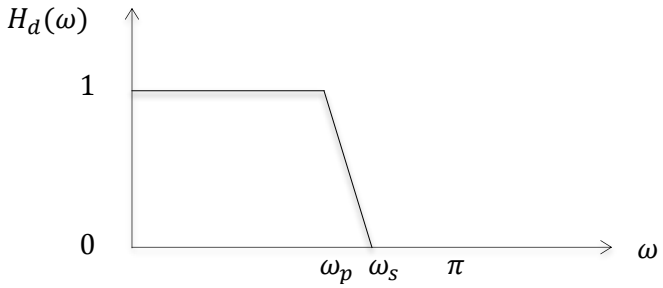


Fig. 5.2 Desired frequency response.

5.2.3 A CCP-based design

Let $H_d(\omega)$ be a desired real-valued or complex-valued function of frequency ω, and $H(\omega, \boldsymbol{x})$ be a real-valued or complex-valued function of ω and a real-valued parameter vector $\boldsymbol{x} \in R^{n\times 1}$. We seek to find a vector $\boldsymbol{x}^*$ that solves the *weighted minimax optimization problem*

$$\text{minimize } \{\max_{\omega\in\Omega} W(\omega)\,|H(\omega, \boldsymbol{x}) - H_d(\omega)|\} \tag{5.19}$$

where $W(\omega) \ge 0$ is a given nonnegative function that frequency-wisely weighs the importance the approximation error $|H(\omega, \boldsymbol{x}) - H_d(\omega)|$, and Ω denotes the set of frequencies where the filter to be designed is required to best approximate the desired frequency response so that the largest approximation error over Ω is minimized.

If we let η be an upper bound of $W(\omega)|H(\omega,\boldsymbol{x})-H_d(\omega)|$ on Ω, and treat η as an auxiliary design variable, the problem in (5.19) can be solved by minimizing the upper bound η. In this way, we convert the problem in (5.19) into a constrained minimization problem

$$\text{minimize} \quad \eta \tag{5.20a}$$

$$\text{subject to: } W(\omega)|H(\omega,\boldsymbol{x})-H_d(\omega)| \le \eta, \quad \text{for } \omega \in \Omega \tag{5.20b}$$

where the objective function is linear, hence the burden of the design is to deal with the constraints. The constraints in (5.20b) can be rewritten as $|H(\omega,\boldsymbol{x})-H_d(\omega)| \le \eta'$ where $\eta' = \eta / W(\omega)$, thus (5.20b) becomes

$$\begin{cases} H(\omega,\boldsymbol{x})-H_d(\omega) \le \eta' \\ -H(\omega,\boldsymbol{x})+H_d(\omega) \le \eta' \end{cases} \tag{5.21}$$

and the problem in (5.20) can be expressed as

$$\text{minimize} \quad \eta \tag{5.22a}$$

$$\text{subject to:} \quad H(\omega,\boldsymbol{x})-H_d(\omega)-\eta' \le 0 \tag{5.22b}$$

$$-H(\omega,\boldsymbol{x})+H_d(\omega)-\eta' \le 0 \tag{5.22c}$$

The class of CCP problems most relevant to the design problem at hand is given by

$$\text{minimize} \quad f_0(\boldsymbol{x}) - g_0(\boldsymbol{x}) \tag{5.23a}$$

$$\text{subject to: } f_i(\boldsymbol{x}) - g_i(\boldsymbol{x}) \le 0, i = 1, \ldots, m \tag{5.23b}$$

where $f_0(\boldsymbol{x}) = \eta$ and $g_0(\boldsymbol{x}) = 0$, namely,

$$\text{minimize} \quad \eta$$

$$\text{subject to: } f_i(\boldsymbol{x}) - g_i(\boldsymbol{x}) \le 0, i = 1, \ldots, m \tag{5.24}$$

To convert the optimization problem to a CCP problem, first, we need to express each of the constraints $H(\omega,\boldsymbol{x})-H_d(\omega)-\eta'$and $-H(\omega,\boldsymbol{x})+H_d(\omega)-\eta'$ as difference of two convex functions.

To this end, we propose the following procedures:

1. Add a convex function $P(\boldsymbol{x},\omega)$ to both sides of (5. 22b) and (5.22c), where

$$P(\boldsymbol{x},\omega) = [a^T c(\omega)]^2 + \tfrac{1}{2}[a_a^T c_a(\omega)]^2 + \tfrac{1}{2}[a_c^T c_c(\omega)]^2 \tag{5.25}$$

and write

$$P(\boldsymbol{x},\omega) + H(\omega,\boldsymbol{x}) = [a^T c(\omega)]^2 + \tfrac{1}{2}[a_a^T c_a(\omega)]^2 + \tfrac{1}{2}[a_c^T c_c(\omega)]^2$$

$$+[a^T c(\omega)][a_a^T c_a(\omega) - a_c^T c_c(\omega)] + a_c^T c_c(\omega)$$

$$= \tfrac{1}{2}[a^T c(\omega) + a_a^T c_a(\omega)]^2 + \tfrac{1}{2}[a^T c(\omega) - a_c^T c_c(\omega)]^2 + a_c^T c_c(\omega)$$

$$P(\boldsymbol{x}, \omega) - H(\omega, \boldsymbol{x}) = [a^T c(\omega)]^2 + \tfrac{1}{2}[a_a^T c_a(\omega)]^2 + \tfrac{1}{2}[a_c^T c_c(\omega)]^2$$
$$-[a^T c(\omega)][a_a^T c_a(\omega) - a_c^T c_c(\omega)] - a_c^T c_c(\omega)$$

$$= \tfrac{1}{2}[a^T c(\omega) - a_a^T c_a(\omega)]^2 + \tfrac{1}{2}[a^T c(\omega) + a_c^T c_c(\omega)]^2 - a_c^T c_c(\omega)$$

2. Define

$$U(\boldsymbol{x}, \omega) = \tfrac{1}{2}[a^T c(\omega) + a_a^T c_a(\omega)]^2 + \tfrac{1}{2}[a^T c(\omega) - a_c^T c_c(\omega)]^2 + a_c^T c_c(\omega)$$
$$-H_d(\omega) - \eta'$$

$$V(\boldsymbol{x}, \omega) = \tfrac{1}{2}[a^T c(\omega) - a_a^T c_a(\omega)]^2 + \tfrac{1}{2}[a^T c(\omega) + a_c^T c_c(\omega)]^2 - a_c^T c_c(\omega)$$
$$+H_d(\omega) - \eta'$$

Note that both $U(\boldsymbol{x}, \omega)$ and $V(\boldsymbol{x}, \omega)$ are convex.

3. Now the constraints in (5.22b) and (5.22c) can be expressed as

$$U(\boldsymbol{x}, \omega) - P(\boldsymbol{x}, \omega) \leq 0, \qquad \omega \in \Omega$$

$$V(\boldsymbol{x}, \omega) - P(\boldsymbol{x}, \omega) \leq 0, \qquad \omega \in \Omega$$

which obviously fit into the form in (5.24). Therefore the CCP applies and in doing so we obtain the convexified constraints

$$U(\boldsymbol{x}, \omega) \leq \tilde{P}(\boldsymbol{x}, \boldsymbol{x}_k, \omega), \qquad \omega \in \Omega$$

$$V(\boldsymbol{x}, \omega) \leq \tilde{P}(\boldsymbol{x}, \boldsymbol{x}_k, \omega), \qquad \omega \in \Omega$$

where $\tilde{P}(\boldsymbol{x}, \boldsymbol{x}_k, \omega)$ is a linear appoximation of $P(\boldsymbol{x}, \omega)$ at $\boldsymbol{x} = \boldsymbol{x}_{k,}$ and is given by

$$\tilde{P}(\boldsymbol{x}, \boldsymbol{x}_k, \omega) = P(\boldsymbol{x}_k, \omega) + \nabla^T P(\boldsymbol{x}_k, \omega)(\boldsymbol{x} - \boldsymbol{x}_k)$$

where, by using (5.25) the gradient of $P(\boldsymbol{x}, \omega)$ at $\boldsymbol{x} = \boldsymbol{x}_k$ is found to be

$$\nabla P(\boldsymbol{x}_k, \omega) = \begin{bmatrix} 2[a^T c(\omega)] \cdot c(\omega) \\ [a_a^T c_a(\omega)] \cdot c_a(\omega) \\ a_c^T c_c(\omega) \cdot c_c(\omega) \end{bmatrix}$$

Note that the convexity of $P(\boldsymbol{x}, \omega)$ implies that $P(\boldsymbol{x}, \omega) \geq \tilde{P}(\boldsymbol{x}, \boldsymbol{x}_k, \omega)$.

In summary, the design problem at hand has now been formulated as the *sequential convex problem* (SCP)

$$\text{minimize} \quad \eta \tag{5.26a}$$

$$\text{subject to:} \quad U(\boldsymbol{x}, \omega) \leq \tilde{P}(\boldsymbol{x}, \boldsymbol{x}_k, \omega), \qquad \omega \in \Omega \tag{5.26b}$$

$$V(\boldsymbol{x}, \omega) \leq \tilde{P}(\boldsymbol{x}, \boldsymbol{x}_k, \omega), \qquad \omega \in \Omega \tag{5.26c}$$

We initiate the design with a reasonable initial point $\boldsymbol{x}_0$ which represents a design obtained for example by that proposed in [4]. By solving problem (5.26) with $\boldsymbol{x}_k = \boldsymbol{x}_0$, which can be done using a reliable solver for convex programming problems such as CVX [23], we obtain a global solution which we denote by $\boldsymbol{x}_1$. Iterate $\boldsymbol{x}_1$ then serves as a new initial point to initiate the next convex problem in (5.26) with $\boldsymbol{x}_k = \boldsymbol{x}_1$. In this manner, this sequential convex minimization continues until the difference in 2-norm between the two most current iterates, $\boldsymbol{x}_{k-1}$ and $\boldsymbol{x}_k$, is less than a prescribed tolerance ε. The iterate $\boldsymbol{x}_k$ is then taken to be the solution of the design problem. In principle, a solution with improved performance may be found by using a reduced tolerance at the cost of increased amount of computations.

5.3 Design Examples

Given design specifications M, N, N_a, N_c, ω_p and ω_s, a reasonable initial point $\boldsymbol{x}_0$ can be obtained by designing lowpass $H_a(z), H_{ma}(z)$ and $H_{mc}(z)$ following reference [4]. As will be demonstrated by simulations shortly, it is important to stress that although the optimized $H_a(z)$ does not look like a lowpass filter, the initial design prepared here worked flawlessly in a variety of FRM designs we have attempted. For illustration and comparison purposes, all design examples presented in this section are linear-phase FIR lowpass FRM filers.

In the construction of a set of frequency grids, Ω, placing a relatively denser frequency grids in the regions near passband as well as stopband edges helps to avoid using unnecessarily large number of total grid points. We recommend that about 25% of the grid points be placed in the 10% of the band nearest to the band edge.

Example 1

The first design has the same design parameters as the first example in [9], i.e., $N = 45, N_a = 41, N_c = 33, M = 9, \omega_p = 0.6\pi$, and $\omega_s = 0.61\pi$. The design specifications lead to transfer functions of the sub-filters of the form

$$H_a(z) = \sum_{k=0}^{44} h_k z^{-k} \tag{5.27}$$

$$H_{ma}(z) = \sum_{k=0}^{40} h_k^{(a)} z^{-k} \tag{5.28}$$

$$H_{mc}(z) = \sum_{k=0}^{32} h_k^{(c)} z^{-k} \tag{5.29}$$

where the impulse responses of these sub-filters are symmetrical with respect to their respective midpoint, and Eqs. (5.8) - (5.13) in this case assume the form

$$\boldsymbol{a} = [h_{22}\ 2h_{23}\ \cdots\ 2h_{44}]^T \tag{5.30}$$

$$\boldsymbol{c}(\omega) = [1\cos 9\omega\ \ \cdots \cos[(44\times 9\omega/2]]^T \tag{5.31}$$

$$\boldsymbol{a}_a = [h_{20}^a\ \ 2h_{21}^a\ \ \cdots\ 2h_{22}^a\ ]^T \tag{5.32}$$

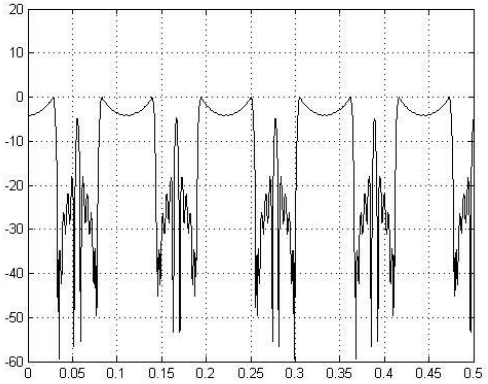
$$\boldsymbol{c}_a(\omega) = [1\cos\omega\ \cdots \cos[20\omega]]^T \tag{5.33}$$

$$\boldsymbol{a}_c = [h_{16}^c\ \ 2h_{17}^c\ \ \cdots\ 2h_{32}^c\ ]^T \tag{5.34}$$

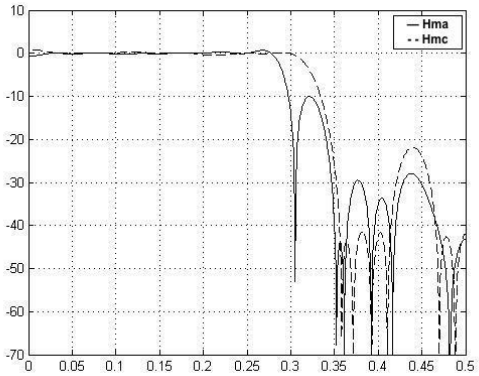
$$\boldsymbol{c}_c(\omega) = \ [1\cos\omega\ \cdots \cos[16\omega]^T \tag{5.35}$$

The weight was set to $\omega = 1$, and the total number of grids was $K = 1200$. With 70 iterations, the CCP-based algorithm converges to an FRM filter with the amplitude response of its sub-filter $H_a(z^M), H_{ma}(z)$ and $H_{mc}(z)$ shown in Fig. 5.3(a) and (b), respectively, and the amplitude response of the FRM filter and its passband ripples shown in Fig. 5.3(c) and (d), respectively. The maximum passband ripple was found to be 0.0673 dB and the minimum stopband attenuation was 42.25dB. By comparison, the passband ripple and stop-band attenuation of the design in [4] were 0.0896 dB and 40.96 dB, respectively. Compared with the well-known method "Parks-McClellan Algorithm", which is an iterative algorithm for designing FIR filters. To have the same performance with our design example, the number of multiplexers that is needed by using Parks-McClellan Algorithm is 209, while in our case, it is 61.

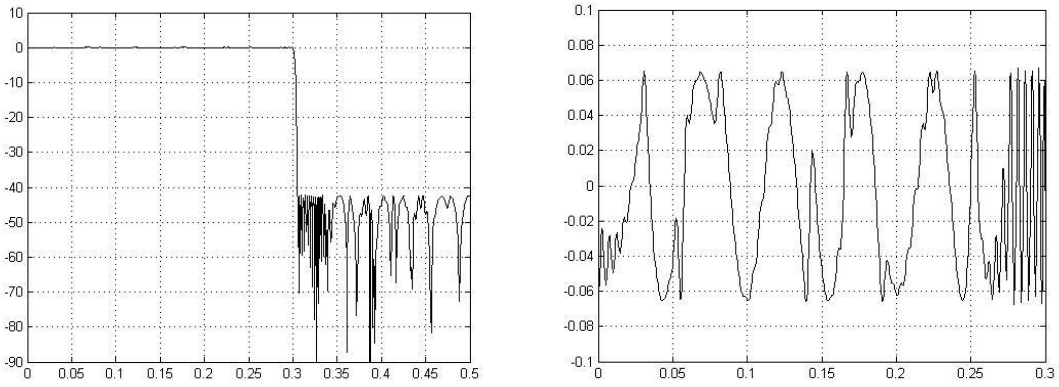
As can be seen from Fig. 5.3, the masking filters $H_{ma}(z)$ and resulted from the joint optimization remain low-pass with very similar passband widths, but the optimized prototype filter $H_a(z^M)$ does not appear to be a lowpass filter.



(a)



(b)



(c) (d)

Fig. 5.3 Amplitude responses of (a) prototype filter $H_a(z^M)$; (b) masking filters $H_{ma}(z)$ and $H_{mc}(z)$; (c) FRM filter; and (d) passband ripples of the FRM filter for Example 1, all in decibels.

Example 2

The design specifications of the second example were set to $N = 43,\ N_a = 37,\ N_c = 29,$ $M = 9,\ \omega_p = 0.6\pi,$ and $\omega_s = 0.61\pi$. The transfer functions of the sub-filters in a basic FRM filter are:

$$H_a(z) = \sum_{k=0}^{42} h_k z^{-k} \tag{5.36}$$

$$H_{ma}(z) = \sum_{k=0}^{36} h_k^{(a)} z^{-k} \tag{5.37}$$

$$H_{mc}(z) = \sum_{k=0}^{28} h_k^{(c)} z^{-k} \tag{5.38}$$

and Eqs. (5.8) - (5.13) become

$$\boldsymbol{a} = [h_{21}\ 2h_{22}\ \cdots\ 2h_{42}]^T \tag{5.39}$$

$$\boldsymbol{c}(\omega) = [1\ \cos 9\omega\ \ \cdots \cos[(42\times 9\omega/2]]^T \tag{5.40}$$

$$\boldsymbol{a}_a = [h_{18}^a\ \ 2h_{19}^a\ \ \cdots\ 2h_{36}^a\ ]^T \tag{5.41}$$

$$\boldsymbol{c}_a(\omega) = [1\ \cos\omega\ \cdots \cos[18\omega]]^T \tag{5.42}$$

$$\boldsymbol{a}_c = [h_{14}^c\ \ 2h_{15}^c\ \ \cdots\ 2h_{28}^c\ ]^T \tag{5.43}$$

$$\boldsymbol{c}_c(\omega) = \ [1\ \cos\omega\ \cdots \cos[14\omega]^T \tag{5.44}$$

The weight was set to $\omega = 1$, and the total number of grids was $K = 1200$. With 965 iterations, the algorithm converges to an FRM filter with the amplitude response of its sub-filter $H_a(z^M), H_{ma}(z)$, and $H_{mc}(z)$ shown in Fig. 5.4(a) and (b), respectively, and the amplitude response of the FRM filter and its passband ripples shown in Fig. 5.4(c) and (d), respectively. The maximum passband ripple was found to be 0.0765 dB and the minimum stopband attenuation was 41.0755dB. To have the same performance with our design example, the number of multiplexers that is needed by using Parks-McClellan Algorithm is 202, while in our case, it is 56.

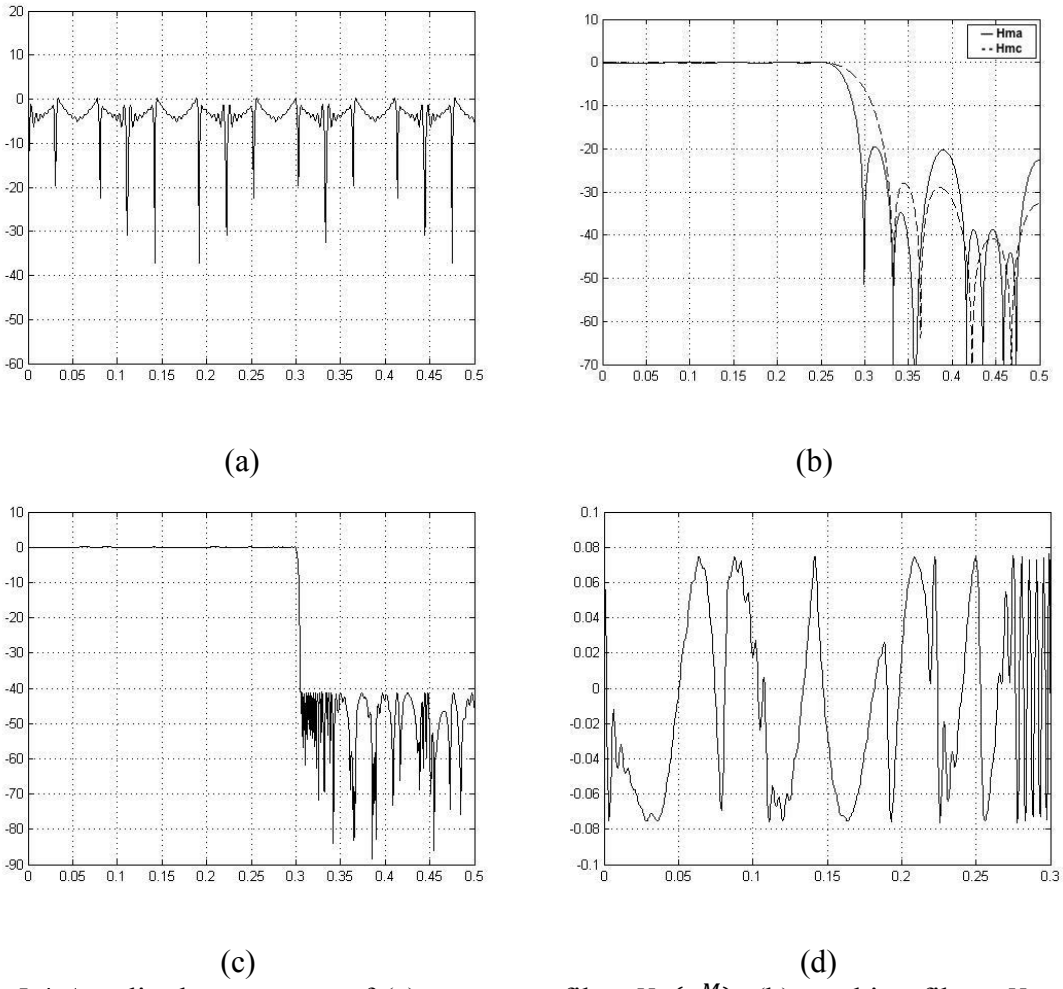


Fig. 5.4 Amplitude responses of (a) prototype filter $H_a(z^M)$; (b) masking filters $H_{ma}(z)$ and $H_{mc}(z)$; (c) FRM filter; and (d) pass-band ripples of the FRM filter for Example 2, all in decibels.

Example 3

In this example, the design specifications were given by $N = 57, N_a = 31, N_c = 25, M = 7, \omega_p = 0.65\pi$, and $\omega_s = 0.66\pi$. These parameters lead to the following transfer functions of the subfilters:

$$H_a(z) = \sum_{k=0}^{56} h_k z^{-k} \tag{5.45}$$

$$H_{ma}(z) = \sum_{k=0}^{30} h_k^{(a)} z^{-k} \tag{5.46}$$

$$H_{mc}(z) = \sum_{k=0}^{24} h_k^{(c)} z^{-k} \tag{5.47}$$

and Eqs. (5.8) - (5.13) assume the form

$$\boldsymbol{a} = [h_{28}\ 2h_{29}\ \cdots\ 2h_{56}]^T \tag{5.48}$$

$$\boldsymbol{c}(\omega) = [1\ \cos 7\omega\ \ \cdots \cos[(56\times 7\omega/2]]^T \tag{5.49}$$

$$\boldsymbol{a}_a = [h_{15}^a\ \ 2h_{16}^a\ \ \cdots\ 2h_{30}^a\ ]^T \tag{5.50}$$

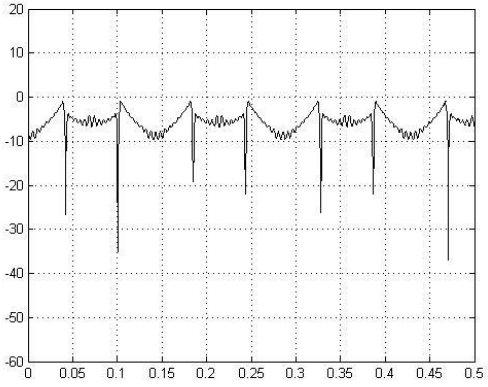
$$\boldsymbol{c}_a(\omega) = [1\ \cos\omega\ \cdots \cos[15\omega]]^T \tag{5.51}$$

$$\boldsymbol{a}_c = [h_{12}^c\ \ 2h_{13}^c\ \ \cdots\ 2h_{24}^c\ ]^T \tag{5.52}$$

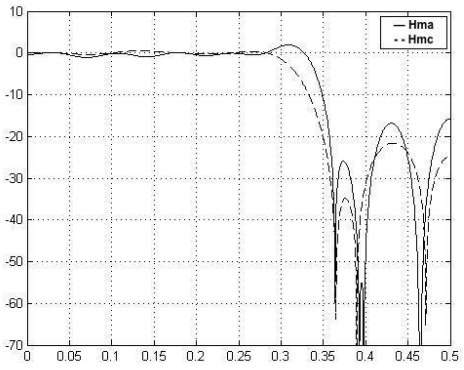
$$\boldsymbol{c}_c(\omega) = \ [1\ \cos\omega\ \cdots \cos[12\omega]^T \tag{5.53}$$

The weight was set to $\omega = 1$, and the total number of grids was $K = 2400$. With 2275 iterations, the algorithm converges to an FRM filter with the amplitude response of its sub-filter $H_a(z^M), H_{ma}(z)$ and$H_{mc}(z)$ shown in Fig. 5.5(a) and (b), respectively, and the amplitude response of the FRM filter and its passband ripples shown in Fig. 5.5(c) and (d), respectively. The maximum passband ripple was found to be 0.0727dB and the minimum stopband attenuation was 41.5831dB. To have the same performance with our design example, the number of multiplexers that is needed by using Parks-McClellan Algorithm is 204, while in our case, it is 58.

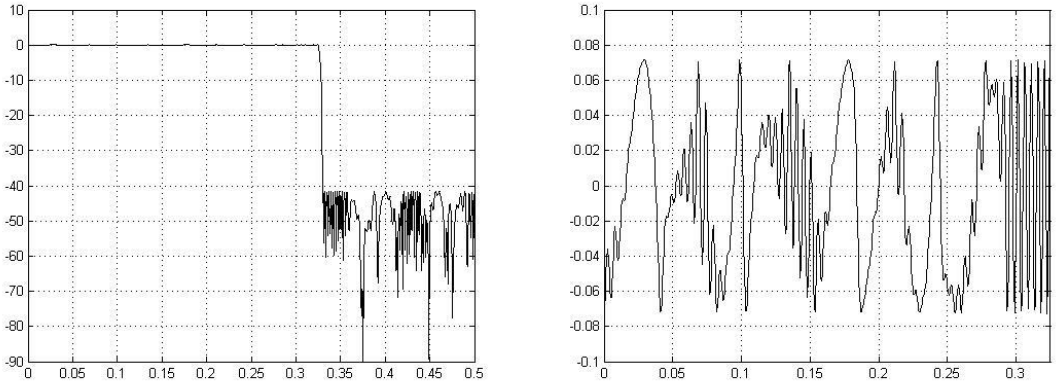
From these design examples, it is observed that the CCP-based algorithm works well for the design of FRM filters of various lengths and sampling factors and offers excellent performance relative to that of the design benchmark [2].



(a)



(b)



(c) (d)

Fig. 5.5 Amplitude responses of (a) prototype filter $H_a(z^M)$; (b) masking filters $H_{ma}(z)$ and $H_{mc}(z)$; (c) FRM filter; and (d) pass-band ripples of the FRM filter for Example 3, all in decibels.

Chapter 6

Conclusions and Future Work

In this report, we have reviewed the method of the frequency-response masking filters and proposed a methodology for optimum design of FRM filters. As illustrated in [2], we treat the coefficients of all sub-filters as a single set of design variables regardless of the number of stages that the FRM filter has. As a result, the joint optimization of all participating sub-filters leads to improved design performance. In this process, we have converted the design problem at hand to an iterative CCP problem, where the convex-concave procedure produces convex sub-problems whose solutions are found to converge to a local solution of the design problem with satisfactory performance. Numerical simulations are included to demonstrate that the CCP-based algorithm can provide comparable or improved designs relative to well established methods from the literature.

In this report only one-stage basic FRM filters are addressed. To date the problem of whether the design of multistage FRM filters can also be handled by a CCP-based technique remains open, and seems worthwhile for consideration as a topic for future research.

Bibliography

[1] A. Antoniou, *Digital Signal Processing: Signal, Systems, and Filters*, second edition, McGraw Hill, 2006.

[2] W.-S. Lu and T. Hinamoto, "Optimal design of frequency-response-masking filters using semidefinite programming," *IEEE Trans. circuits and systems I*, vol. 50, no. 4, pp. 557-568, April 2003.

[3] J. F. Kaiser, "Nonrecursive digital filter design using i_0-sinh window function," in *Proc. IEEE Int. Symp. Circuits Syst*., Apr. 1974, pp. 20–30.

[4] Y. C. Lim, "Frequency-response masking approach for the synthesis of sharp linear phase digital filters," *IEEE Trans. Circuits Syst.*, vol. 33, pp. 357–364, Apr. 1986.

[5] Y. C. Lim and Y. Lian, "The optimum design of one- and two-dimensional FIR filters using the frequency response masking technique," *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 88–95, Feb. 1993.

[6] Y. C. Lim and Y. Lian, "Frequency-response masking approach for digital filter design: Complexity reduction via masking filter factorization," *IEEE Trans. Circuits Syst. II*, vol. 41, pp. 518–525, Aug. 1994.

[7] T. Saramaki. "Design of computationally efficient FIR filters using periodic subfilters as building blocks," in *Circuits and Filters Handbook*. edited by W.-K. Chen, CRC Press, pp. 2578-2601, 1995.

[8] T. Saramaki and Y. C. Lim. "Use of the Remez algorithm for desigiiing FIR filters utilizing the frequency-response masking approach," in *Proc. IEEE 7th. Symp. Circuits Syst*., pp. 449-455, Orlando, FL. May 1999.

[9] H. Johansson, T. Saramaki and J. Yli-Kaakinen, "Optimization of Frequency-Response Masking based FIR Filters", *J. Circuits, Systems, and Computers*, vol. 12, no. 5, pp. 563–591, 2003.

[10] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Prentice-Hall, 1975.

[11] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Prentice-Hall, 1975. (Chapters 4-5).

[12] E. Ya. Remez, "Sur la détermination des polynômes d'approximation de degré donnée", Comm. Soc. Math. Kharkov 10, 41 (1934); "Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation, Compt.

Rend. Acad. Sc. 198, 2063 (1934); "Sur le calcul effectiv des polynômes d'approximation des Tschebyscheff", Compt. Rend. Acade. Sc. 199, 337 (1934).

[13] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998. (pp. 221–222).

[14] A. L. Yuille and Anand Rangarajan, "The concave-convex procedure (CCCP)," Smith-Kettlewell Eye Research Institute, San Francisco, CA 94115, USA.

[15] D. R. Hunter and K. Lange, "A tutorial on MM algorithms," *The American Statistician*, vol. 58, pp. 30–37, 2004.

[16] D. P. Bertsekas, *Nonlinear Programming,* Second ed., Athena Scientific*,* 1999.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, 2007. (Section 10.11, Linear Programming: Interior-Point Methods).

[18] S. P. Boyd and L. Vandenberghe: Convex Optimization, Cambridge University Press, 2004.

[19] R. Horst and N. V. Thoai, "DC Programming: Overview," *J. Optimization Theory and Applications*, vol.103, no.1, pp. 1-43, October 1999.

[20] A. Antoniou, *Digital Filters: Analysis, Design, and Applications*, 2nd ed., McGraw-Hill, 1993.

[21] T. Lipp and S. Boyd: Variations and extensions of the convex-concave procedure, *Optimization and Engineering*, vol. 17, no. 2, pp. 263-287, June 2016.

[22] G. R. Lanckreit and B. K. Sriperumbudur, „On the convergence of the concave-convex procedure," in *Advances in Neural Information Processing Systems*, pp. 1759–1767, 2009.

[23] CVX Users' Guide Release 2.1, CVX Research, Inc.

[24] S. R. K. Dutta and M. Vidyasagar, "New algorithms for constrained minimax optimization," *Math. Programming*, vol. 13, pp. 140–155, 1977.

[25] Gibbs, J. Willard (1898), "Fourier's Series", Nature, **59** (1522): 200, doi: 10.1038/059200b0*,* ISSN 0028-0836

Appendix

MATLAB Code

```
function [h,ha,hc,xk,Er_p,Er_a] = FRM_CCP
(N,Na,Nc,M,fp,fa,w,K,Ki,x0)
n = 0.5*(N+1);
na = 0.5*(Na+1);
nc = 0.5*(Nc+1);
ind = 0:1:(n-1);
ind = ind(:);
inda = 0:1:(na-1);
inda = inda(:);
indc = 0:1:(nc-1);
indc = indc(:);
if nargin > 9,
   L = length(x0);
   xk = x0;
   basic_eval(xk,N,Na,Nc,M);
else
   [xk,L] = basic_initial(N,Na,Nc,M,fp,fa);
end
k = 0;
ak = xk(1:n);
aak = xk((n+1):(n+na));
ack = xk((n+na+1):(n+na+nc));
Mp = round(K*fp/(fp+1-fa));
Ma = K - Mp;
W = [ones(Mp,1); w*ones(Ma,1)];
Wi = 1./W;
fpp = 0:fp/(Mp-1):fp;
omi1 = fpp*pi;
faa = fa:(1-fa)/(Ma-1):1;
omi2 = faa*pi;
omi = [omi1(:); omi2(:)];
Ad = [ones(Mp,1); zeros(Ma,1)];
C = []; Ca = []; Cc = [];
for i = 1:K,
    fi = omi(i);
    C = [C cos(fi*M*ind)];
    Ca = [Ca cos(fi*inda)];
    Cc = [Cc cos(fi*indc)];
end
Er_p = [];
Er_a = [];
za = zeros(na,1);
```

```
zc = zeros(nc,1);
I2 = eye(2);
format long
while k < Ki,
  cvx_begin quiet
     variable dt(1,1)
     variable a(n,1)
     variable aa(na,1)
     variable ac(nc,1)
     minimize(dt)
     subject to
     x = [a; aa; ac];
     for i = 1:K,
        wii = Wi(i);
        ci = C(:,i);
        cai = Ca(:,i);
        cci = Cc(:,i);
        ii = ak'*ci;
        iai = aak'*cai;
        ici = ack'*cci;
        pxi = ii^2 + 0.5*(iai^2 + ici^2) + wii*dt;
        dpi = [2*ii*ci; iai*cai; ici*cci];
        q1 = [ci; cai; zc];
        q2 = [ci; za; -cci];
        Q1i = [q1'; q2'];
        q3 = [ci; -cai; zc];
        q4 = [ci; za; cci];
        Q2i = [q3'; q4'];
        qti = cci'*ac - Ad(i);
        pti = pxi + dpi'*(x - xk);
        et1 = 2*(pti - qti);
        et2 = 2*(pti + qti);
        gke1 = Q1i*x;
        gke2 = Q2i*x;
        [et1 gke1'; gke1 I2] == semidefinite(3);
        [et2 gke2'; gke2 I2] == semidefinite(3);
     end
  cvx_end
  ak = a;
  aak = aa;
  ack = ac;
  k = k + 1;
  xk = [ak; aak; ack];
  [hz,haz,hcz,er_pz,er_az] = final_eval_m(xk,N,Na,Nc,M,fp,fa);
  Er_p = [Er_p er_pz];
  Er_a = [Er_a er_az];
  current_stutus = [k dt er_pz er_az]
  basic_eval(xk,N,Na,Nc,M);
end
format short
xs = xk;
[h,ha,hc,er_p,er_a] = final_eval(xs,N,Na,Nc,M,fp,fa);
```

```
function [x,L] = basic_initial(N,Na,Nc,M,fp,fa)
omip = fp*pi;
omia = fa*pi;
m = floor(omip*M/(2*pi));
thet = omip*M - 2*m*pi;
phi = omia*M - 2*m*pi;
fc = 0.5*(thet+phi)/pi;
fca = (2*m+1)/M - (fa-fp)/2;
fcc = 2*m/M + (fa-fp)/2;
if phi >= pi,
   m = ceil(omia*M/(2*pi));
   thet = 2*m*pi - omia*M;
   phi = 2*m*pi - omip*M;
   fc = 0.5*(thet+phi)/pi;
   fca = (2*m-1)/M + (fa-fp)/2;
   fcc = 2*m/M - (fa-fp)/2;
end
h0 = fir1(N-1,fc);
h0 = h0(:);
ha0 = fir1(Na-1,fca);
ha0 = ha0(:);
hc0 = fir1(Nc-1,fcc);
hc0 = hc0(:);
Nw = 0.5*N;
Naw = 0.5*Na;
if round(Nw) - Nw > 0,
   n = (N+1)/2;
   a = [h0(n); 2*h0(n+1:end)];
   g = 0:M:M*(n-1);
else
   n = N/2;
   a = 2*h0(n+1:end);
   g = (0.5:1:0.5*(N-1))*M;
end
g = g(:);
if round(Naw) - Naw > 0,
   na = (Na+1)/2;
   aa = [ha0(na); 2*ha0(na+1:end)];
   ga = 0:1:(na-1);
   nc = (Nc+1)/2;
   ac = [hc0(nc); 2*hc0(nc+1:end)];
   gc = 0:1:(nc-1);
else
   na = Na/2;
   aa = 2*ha0(na+1:end);
   ga = 0.5:1:0.5*(Na-1);
   nc = Nc/2;
   ac = 2*hc0(nc+1:end);
   gc = 0.5:1:0.5*(Nc-1);
```

```
end
ga = ga(:);
gc = gc(:);
x = [a; aa; ac];
L = length(x);
f = 0:pi/1023:pi;
f = f(:);
H = zeros(1024,1);
F = H;
Fa = H;
Fc = H;
for i =1:1024,
   fw = f(i);
   c = cos(g*fw);
   ca = cos(ga*fw);
   cc = cos(gc*fw);
   F(i) = a'*c;
   Fa(i) = aa'*ca;
   Fc(i) = ac'*cc;
end
H = F.*(Fa-Fc) + Fc;
ff = 0.5*f/pi;
figure(1)
plot(ff,20*log10(abs(F)),'k','LineWidth',1.1)
axis([0 0.5 -20 10])
grid
pause(0.5)
figure(2)
plot(ff,20*log10(abs(Fa)),'-k','LineWidth',1.1)
hold on
plot(ff,20*log10(abs(Fc)),'--k','LineWidth',1.1)
axis([0 0.5 -70 10])
grid
hold off
pause(0.5)
figure(3)
plot(ff,20*log10(abs(H)),'k','LineWidth',1.1)
axis([0 0.5 -80 10])
grid
pause(0.5)
```

```
function basic_eval(xk,N,Na,Nc,M)
L = length(xk);
Nw = 0.5*N;
Naw = 0.5*Na;
if round(Nw) - Nw > 0,
   n = (N+1)/2;
   g = (0:1:n-1)*M;
else
   n = N/2;
```

```
   g = (0.5:1:0.5*(N-1))*M;
end
a = xk(1:n);
g = g(:);
if round(Naw) - Naw > 0,
   na = (Na+1)/2;
   ga = 0:1:(na-1);
   nc = (Nc+1)/2;
   gc = 0:1:(nc-1);
else
   na = Na/2;
   ga = 0.5:1:0.5*(Na-1);
   nc = Nc/2;
   gc = 0.5:1:0.5*(Nc-1);
end
aa = xk(n+1:n+na);
ac = xk(n+na+1:L);
ga = ga(:);
gc = gc(:);
f = 0:pi/1023:pi;
f = f(:);
H = zeros(1024,1);
F = H;
Fa = H;
Fc = H;
for i =1:1024,
   fw = f(i);
   c = cos(g*fw);
   ca = cos(ga*fw);
   cc = cos(gc*fw);
   F(i) = a'*c;
   Fa(i) = aa'*ca;
   Fc(i) = ac'*cc;
end
H = F.*(Fa-Fc) + Fc;
ff = 0.5*f/pi;
figure(1)
plot(ff,20*log10(abs(F)),'k','LineWidth',1.1)
axis([0 0.5 -60 20])
grid
pause(0.5)
figure(2)
plot(ff,20*log10(abs(Fa)),'-k','LineWidth',1.1)
hold on
plot(ff,20*log10(abs(Fc)),'--k','LineWidth',1.1)
hold off
axis([0 0.5 -70 10])
grid
pause(0.5)
figure(3)
plot(ff,20*log10(abs(H)),'k','LineWidth',1.1)
axis([0 0.5 -90 10])
```

```
grid
pause(0.5)
```

```
function [h,ha,hc,er_p,er_a] = final_eval_m(xs,N,Na,Nc,M,fp,fa)

omip = pi*fp;
omia = pi*fa;
L = length(xs);
Nw = 0.5*N;
Naw = 0.5*Na;
if round(Nw) - Nw > 0,
   n = (N+1)/2;
   g = 0:M:M*(n-1);
   a = xs(1:n);
   h = [0.5*flipud(a(2:end)); a(1); 0.5*a(2:end)];
else
   n = N/2;
   g = (0.5:1:0.5*(N-1))*M;
   a = xs(1:n);
   h = [0.5*flipud(a); 0.5*a];
end
g = g(:);
if round(Naw) - Naw > 0,
   na = (Na+1)/2;
   ga = 0:1:(na-1);
   aa = xs(n+1:n+na);
   ha = [0.5*flipud(aa(2:end)); aa(1); 0.5*aa(2:end)];
   nc = (Nc+1)/2;
   gc = 0:1:(nc-1);
   ac = xs(n+na+1:L);
   hc = [0.5*flipud(ac(2:end)); ac(1); 0.5*ac(2:end)];
else
   na = Na/2;
   ga = 0.5:1:0.5*(Na-1);
   aa = xs(n+1:n+na);
   ha = [0.5*flipud(aa); 0.5*aa];
   nc = Nc/2;
   gc = 0.5:1:0.5*(Nc-1);
   ac = xs(n+na+1:L);
   hc = [0.5*flipud(ac); 0.5*ac];
end
ga = ga(:);
gc = gc(:);
Mp = round(1024*fp/(fp+1-fa));
Ma = 1024 - Mp;
ffp = 0:omip/(Mp-1):omip;
ffa = omia:(pi-omia)/(Ma-1):pi;
Hp = zeros(Mp,1);
for i = 1:Mp,
    fw = ffp(i);
```

```
    c = cos(g*fw);
    ca = cos(ga*fw);
    cc = cos(gc*fw);
    zw = ac'*cc;
    Hp(i) = (a'*c)*(aa'*ca - zw) + zw;
end
Hpp = 20*log10(abs(Hp));
er_p = max(abs(Hpp(1:end-1)));
Ha = zeros(Ma,1);
for i = 1:Ma,
    fw = ffa(i);
    c = cos(g*fw);
    ca = cos(ga*fw);
    cc = cos(gc*fw);
    zw = ac'*cc;
    Ha(i) = (a'*c)*(aa'*ca - zw) + zw;
end
er_a = -max(20*log10(abs(Ha(2:end))));
```