THE UNIVERSITY OF CALGARY

An Optimal Model-Matching Approach to Multirate FIR Filter Bank Design

by

Saskia Ratzlaff

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER

ENGINEERING

CALGARY, ALBERTA

AUGUST, 1995

© Saskia Ratzlaff 1995

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "An Optimal Model-Matching Approach to Multirate FIR Filter Bank Design" submitted by Saskia Ratzlaff in partial fulfillment of the requirements for the degree of Master of Science.

Supervisor, Dr. T. Chen
Dept. of Electrical and Computer Engineering

Co-Supervisor, Dr. S. A. Norman
Dept. of Electrical and Computer Engineering

Dr. B. Petersen
Dept. of Electrical and Computer Engineering

Dr. J. Pieper
Department of Mechanical Engineering

Date: Aug. 31, 1995

ABSTRACT

A new model-matching approach to Quadrature Mirror Filter (QMF) bank design is presented. Matching the QMF bank to be designed with an ideal system forms an error system. The optimization involves FIR synthesis filters designed to minimize the reconstruction error between the QMF bank and the ideal system. Norms are used to quantify the reconstruction error. The convex nature of this problem guarantees global optimality of the resulting minimum.

While originally intended for QMF bank design, this model-matching approach is easily extended to other multirate systems, such as A/D hybrid QMF banks and trans-multiplexers. Furthermore, constraints such as linear-phase filters, cosine-modulated filters, and alias-free systems are also accommodated by the model-matching approach. In addition to the cost introduced by these constraints, tradeoffs between reconstruction properties and filter length or system delay are examined.

ACKNOWLEDGEMENTS

I am deeply grateful to my supervisors Dr. Chen and Dr. Norman both for their financial support and excellent guidance. Their faith in my abilities was an enormous boost to my confidence. While giving me enough independence to explore my own ideas, they were always available to offer support, suggestions, and intuition.

I would also like to express my appreciation for the financial support provided by the Department of Electrical and Computer Engineering. Financial support from the Natural Sciences and Engineering Research Council of Canada, made available by Dr. Chen and Dr. Norman, is also gratefully acknowledged.

I am also indebted to my friends, Saeed Aghtar, Anurag Handa, Mohsen Hosseinian, Shavantha Kularatna, Nadra Rafee, Jason Sokolosky, and Krista Doyle, who helped me to know what I mean, write what I mean, and finally to mean what I write. I'm especially grateful to Niall Sexton who painstakingly read my entire thesis and provided many helpful suggestions.

No acknowledgement would be complete without thanking my three most treasured supporters: my mother, Ed, and of course Peter. My mother's unfailing support and persistence continues to be a great encouragement. Ed's patient listening skills were also instrumental in completing this thesis. Finally, I would like to thank Peter, who knows my dreams even before I do.

TABLE OF CONTENTS

APPROVAL PAGE ii

ABSTRACT iii

ACKNOWLEDGEMENTS iv

TABLE OF CONTENTS v

LIST OF TABLES viii

LIST OF FIGURES x

LIST OF SYMBOLS xiii

1. INTRODUCTION 1

1.1 Subband Coding 1

1.2 Quadrature Mirror Filter Banks 2

1.3 Organization of This Thesis 6

2. MULTIRATE SYSTEMS 8

2.1 Multirate Building Blocks 8

2.2 Polyphase Representation 11

2.3 Traditional QMF Designs 12

2.3.1 Perfect Reconstruction 12

2.3.2 Near Perfect Reconstruction 14

3. FIR $\mathcal{H}_\infty$ MODEL-MATCHING METHOD 20

3.1 Developing the Error System 20

3.2 FIR Problem Formulation 23

3.2.1 Ellipsoid Algorithm 25
3.2.2 Calculating Subgradient 27
3.3 Other Performance Measures 29

4. MODEL-MATCHING APPLICATIONS **33**
4.1 Simple QMF 33
4.2 Fast A/D Converters 36
4.3 Transmultiplexers 41
4.4 Filter Length Tradeoffs 46
4.4.1 QMF Banks 47
4.4.2 Fast A/D Converters 49
4.4.3 Overall System Delay 50

5. ADDING CONSTRAINTS **53**
5.1 Linear-Phase Synthesis Filters 54
5.2 Cosine-Modulated Synthesis Filters 59
5.3 Low Delay Systems 63
5.4 Alias-Free Systems 69

6. APPLICATIONS OF THE $\mathcal{H}_2$ NORM **78**
6.1 Sensitivity to Quantization Noise 78
6.2 Overall Performance Measure 82
6.2.1 Justification 83
6.2.2 Optimization 84
6.2.3 Design Example 85

7. CONCLUSIONS **94**
7.1 Summary 94

7.2 Future Work 96

APPENDIX **97**

A. MATLAB CODE FOR CALCULATING SUBGRADIENTS **97**

A.1 Ellipsoid Algorithm 97

A.2 Subgradient: Unconstrained 98

A.3 Subgradient: Linear-Phase Constraint 101

A.3.1 Even Length Synthesis Filters 101

A.3.2 Odd Length Synthesis Filters 104

A.4 Subgradient: Cosine-Modulated Constraint 108

A.5 Subgradient: Alias-Free Constraint 111

A.5.1 Unconstrained 111

A.5.2 Linear-Phase, Even Length 115

A.5.3 Linear-Phase, Odd Length 118

REFERENCES **123**

LIST OF TABLES

2.1 Comparison of Various NPR Design Methods 18

4.1 Coefficients of Filters for Simple QMF Example 34

4.2 Performance Measures for the Simple QMF Example. 36

4.3 Coefficients of 24-tap Synthesis Filters for A/D Example 40

4.4 Coefficients of 8-tap Filters for Transmultiplexer 45

4.5 8-tap Synthesis Filter Coefficients. 52

5.1 Number of Variables to be Optimized for Various Constraints. 54

5.2 Four Types of Real Coefficient Linear-Phase FIR Filters 54

5.3 Coefficients of 24-tap Linear-Phase Synthesis Filters 56

5.4 Cost of Linear-Phase Synthesis Filters 56

5.5 Performance Measures for the Linear-Phase Example. 58

5.6 Analysis and Synthesis Prototype Filters. 62

5.7 Performance Measures for the Cosine-Modulated Example. 62

5.8 Coefficients of 8-tap Analysis Filters 66

5.9 Coefficients of 8-tap Analysis Filters for Delay = 1 66

5.10 Comparison of $\mathcal{H}_\infty$ Norm 68

5.11 Comparison of SNR_r 68

5.12 Performance Measures for Low-Delay Constraint. 70

5.13 Coefficients of $p(n)$ for Alias-Free System 72

5.14 Effect of Alias-Free Condition 72

5.15 Performance Measures for Alias-Free Constraint. 76

6.1 Coefficients of 24-tap Synthesis Filters for $\mathcal{H}_2$ Performance Measure . . 86

6.2 Comparison of Performance Measures for the Simple QMF Example, Using $\mathcal{H}_2$ Norm and $\mathcal{H}_\infty$ Norm. 87

6.3 8-tap Synthesis Filter Coefficients Using $\mathcal{H}_2$ Norm. 91

LIST OF FIGURES

1.1 Splitting a Signal into Two Subbands. 2

1.2 The Quadrature Mirror Filter Bank. 3

1.3 Traditional Choice of Analysis Filters. 3

1.4 Roles of Decimator and Expander. 4

2.1 The M-Fold Decimator. 8

2.2 The M-Fold Decimator in the Frequency Domain. 9

2.3 The L-Fold Expander. 10

2.4 The L-Fold Expander in the Frequency Domain. 10

3.1 Time-Varying Error System. 21

3.2 Time-Invariant Error System. 22

4.1 Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1. . 35

4.2 Input Ramp (dashed) and Reconstruction Error (solid) versus n. 36

4.3 Magnitude (dB) Distortions for $T(z)$ and $A(z)$ ((a) and (c)) versus ω/π. Phase (°) Distortions for $T(z)$ and $A(z)$ ((b) and (d)) versus ω/π. . . 37

4.4 An Analog/Digital Hybrid QMF Bank. 38

4.5 Error System for Fast A/D Converter. 38

4.6 Time-Invariant Error System for Fast A/D Conversion. 39

4.7 2-Channel Transmultiplexer. 41

4.8 Transmultiplexer Error System. 44

4.9 Transmultiplexer: Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1. 46

4.10 Reconstruction Errors for $\hat{x}_0(n)$ (solid) and $\hat{x}_1(n)$ (dotted) for Input Ramps versus n. 47

4.11 J_{trunc} (dashed) and J_{opt} (solid) versus n_{tap} for Simple QMF Example. . . 48

4.12 J_{trunc} (dashed) and J_{opt} (solid) versus n_{tap} for Fast A/D Converter. . . . 50

4.13 J_{opt} versus Delay for 8-tap (solid), 10-tap (dashed), and 12-tap (dotted) Synthesis Filters. 51

5.1 Linear-Phase Constraint: Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1. 57

5.2 Linear-Phase Constraint: Magnitude (dB) Distortions for $T(z)$ and $A(z)$ ((a) and (c)) versus ω/π. Phase (°) Distortions for $T(z)$ and $A(z)$ ((b) and (d)) versus ω/π. 58

5.3 Linear-Phase Constraint: Input Ramp (dashed) and Reconstruction Error (solid) versus n. 59

5.4 Cosine-Modulated Constraint: Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1. 63

5.5 Cosine-Modulated Constraint: Magnitude (dB) Distortions for $T(z)$ and $A(z)$ ((a) and (c)) versus ω/π. Phase (°) Distortions for $T(z)$ and $A(z)$ ((b) and (d)) versus ω/π. 64

5.6 Cosine-Modulated Constraint: Reconstruction Error versus n. 65

5.7 Low-Delay System: Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1. 67

5.8 Low-Delay System: Magnitude (dB) Distortions for $T(z)$ and $A(z)$ ((a) and (c)) versus ω/π. Phase (°) Distortions for $T(z)$ and $A(z)$ ((b) and (d)) versus ω/π. 69

5.9 Low-Delay System: Reconstruction Error versus n. 70

5.10 Alias-Free Constraint: Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1. 73

5.11 Alias-Free Constraint: Magnitude (dB) Distortions for $T(z)$ and $A(z)$ ((a) and (c)) versus ω/π. Phase (°) Distortions for $T(z)$ and $A(z)$ ((b) and (d)) versus ω/π. 74

5.12 Alias-Free Constraint: Input Ramp (dashed) and Reconstruction Error (solid) versus n. 76

6.1 QMF Bank with Quantizers in the Subbands. 79

6.2 Noise Input to the Synthesis Bank. 79

6.3 Polyphase Representation of Noise Model. 80

6.4 LTI Representation of Noise Model. 80

6.5 Equivalent LTI Representation of Noise Model. 81

6.6 $\|R\|_2^2$ versus $\|W - RE\|_\infty$ for Unconstrained Setup (solid) and Alias-Free Setup (dashed). 83

6.7 $\mathcal{H}_2$ Norm: Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1. 87

6.8 $\mathcal{H}_2$ Performance Measure: Magnitude (dB) Distortions for $T(z)$ and $A(z)$ ((a) and (c)) versus ω/π. Phase (°) Distortions for $T(z)$ and $A(z)$ ((b) and (d)) versus ω/π. 88

6.9 $\mathcal{H}_2$ Performance Measure: Input Ramp (dash) and Reconstruction Error (solid) versus n. 89

6.10 $\mathcal{H}_2$ Performance Measure: J_{opt} versus Delay for 8-tap (solid), 10-tap (dashed), and 12-tap (dotted) Synthesis Filters. 90

LIST OF SYMBOLS

Symbol	Definition
$A(z)$	Alias component
A/D	Analog to Digital
D	Decimator
D*	Expander
$\delta(n)$	Discrete-time unit impulse
$E(z)$	Polyphase representation of analysis filters
FDM	Frequency Division Multiplex
FIR	Finite Impulse Response
$f_k(n)$	Impulse response of kth synthesis filter
$F_k(z)$	kth synthesis filter
$h_k(n)$	Impulse response of kth synthesis filter
$H_k(z)$	kth analysis filter
h_{tap}	Number of taps in analysis filters
IIR	Infinite Impulse Response
J	Performance measure
J_{opt}	Optimal performance measure
J_{trunc}	Performance measure for truncated impulse response
LTI	Linear Time-Invariant

m	Overall system delay
MIMO	Multiple-Input, Multiple-Output
SISO	Single-input, Single-output
NPR	Near Perfect Reconstruction
n_{tap}	Number of taps in synthesis filters
ω	Frequency in radians
PR	Perfect Reconstruction
QMF	Quadrature Mirror Filter
RMS	Root Mean Square
$R(z)$	Polyphase representation of synthesis filters
$\Re(z)$	Real part of complex number z
SNR_r	Signal-to-Noise Ratio for input ramp
TDM	Time Division Multiplex
$T(z)$	Transfer function component
$T_d(z)$	Ideal transfer function
$\mathbf{U}$	Unit time delay
$\mathbf{U}^*$	Unit time advance
$x(n)$	Discrete-time input signal
$\hat{x}(n)$	Discrete-time reconstructed signal
$x_k(n)$	kth discrete-time subband signal

$\|x\|_2$	The ℓ_2 norm of the signal x
$\|G\|_2$	The $\mathcal{H}_2$ norm of the transfer function G
$\|G\|_\infty$	The $\mathcal{H}_\infty$ norm of the transfer function G
$\|G\|_F$	The Frobenius norm of the matrix G
$< \cdot, \cdot >$	Inner product

CHAPTER 1

INTRODUCTION

The need for compressing and coding speech, image, and video signals for transmission and storage has led to the current interest in Quadrature Mirror Filter (QMF) systems. QMF systems are a means of realizing filter banks. Filter banks exploit the presence of dominant energy bands in signals by splitting the signal into subbands. Thus, the subband signal with more energy can be encoded using more bits, whereas the subband signal with less energy can be encoded using fewer bits. The net result is a reduction in the data rate.

1.1 Subband Coding

Subband coding has been used in both speech [1], [2] and image signal processing [3]. In these systems, the various frequency bands of a signal are processed independently by splitting the signal into consecutive subbands. In Figure 1.1 the input signal is split into two frequency bands using a lowpass filter H_0 and highpass filter H_1.

The key concepts in subband coding are decimation and efficient bit allocation. An illustrative example of this was given in [4]: Consider a signal $x(n)$ sampled at 10 kHz, requiring 8 bits per sample. Consequently, the data rate is 80 kbits/s. Furthermore, assume that the subband signals $x_0(n)$ and $x_1(n)$ can be encoded using 8 and 4 bits per sample, respectively. After decimation, the new data rate works out to $40 + 20 = 60$ kbits/s, corresponding to compression by a factor of 4/3.

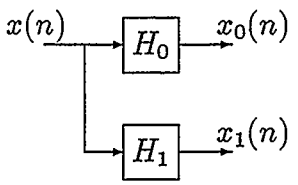

Figure 1.1 Splitting a Signal into Two Subbands.

Successful subband coding requires some *a priori* knowledge about the energy distribution of the input signal. Often, this is not an unrealistic assumption for both speech and image processing [4]. Reconstruction of the original signal is complicated by the fact that bandsplitting and decimation result in aliasing due to non-ideal filters. Subband coding is an example of a multirate system, which exploits the varying sampling requirements throughout the system.

1.2 Quadrature Mirror Filter Banks

A QMF bank is a type of multirate filter bank. QMF banks exploit the fact that multirate filter banks do not need to meet the requirements of the Nyquist sampling theorem on a subband-by-subband basis [5]. Instead, QMF banks use the analysis filters to effectively bandlimit the subband signals and subsequently reduce the sampling rate. Figure 1.2 shows the QMF bank.

The name "Quadrature Mirror Filter" bank is a result of the traditional choice of analysis filters. In the earliest known QMF banks, the frequency-response of $H_1(e^{j\omega})$ was the *mirror-image* of the frequency-response of $H_0(e^{j\omega})$, with respect to a *quarter* of the sampling frequency ($\omega = 2\pi/4$) [6]. Figure 1.3 shows this traditional choice

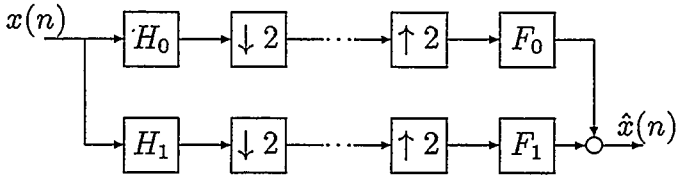

Figure 1.2 The Quadrature Mirror Filter Bank.

of analysis filters. Although the term "maximally decimated analysis-synthesis fil-

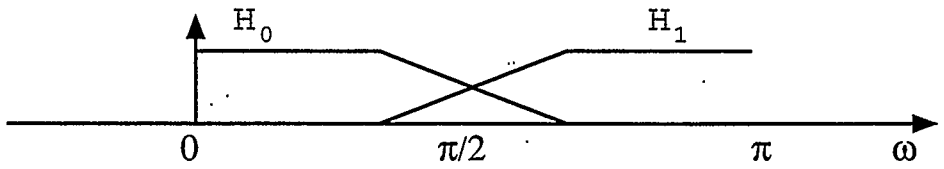

Figure 1.3 Traditional Choice of Analysis Filters.

ter banks" is more general, "QMF" is often used for general M-channel maximally-decimated analysis-synthesis filter banks [7]. For simplicity only, the 2-channel QMF case will be considered.

As shown in Figure 1.2, the 2-channel QMF bank is made up of four filters, two decimators, and two expanders. The implication in the time-domain of downsampling, and subsequently upsampling a signal is illustrated in Figure 1.4. The role of the two-fold decimator is to reduce the sampling rate by retaining only every other sample. On the other hand, the role of the two-fold expander is to increase the sampling rate. This is accomplished by inserting zeros between samples.

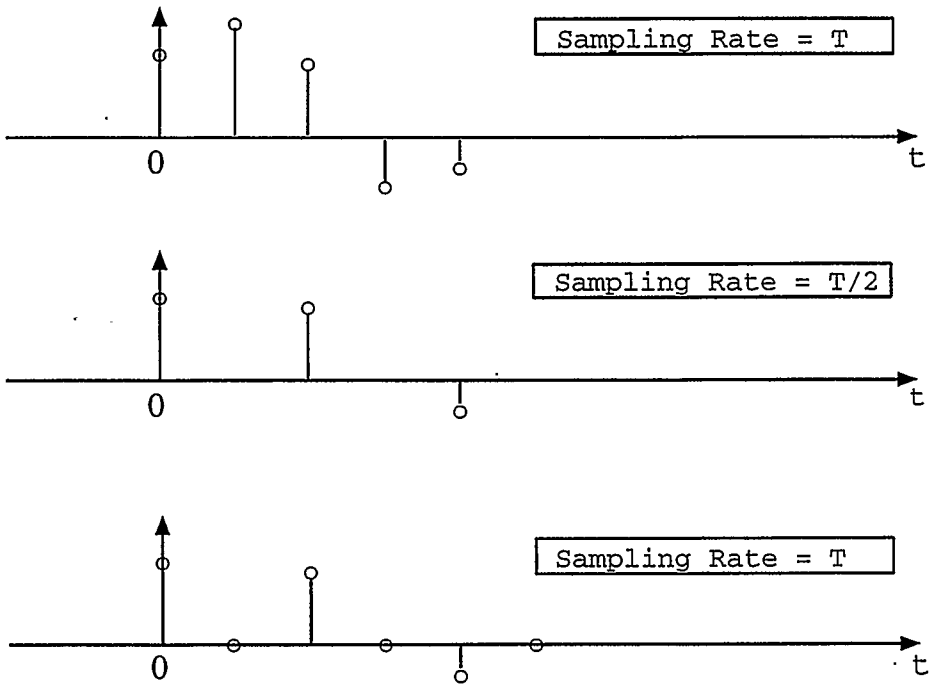

Figure 1.4 Roles of Decimator and Expander.

In a QMF bank, the input signal $x(n)$ is first split into two subbands using the analysis filters, H_0 and H_1, and then the subband signals are downsampled. In speech compression and transmission applications, for example, the subband signals are encoded and transmitted [8]. To recover the original signal, the decoded subband signals are first upsampled and then recombined using the synthesis filters, F_0 and F_1. Any errors due to the coding and decoding processes are neglected in order to simplify the design problem. Ideally, the reconstructed signal is merely a delayed version of the input signal. This ideal situation is called perfect reconstruction. Otherwise, the reconstructed signal differs from the input for three reasons: Aliasing, Magnitude Distortion, and Phase Distortion.

Traditionally, QMF bank design involves designing one or both of the analysis filters, and then defining the synthesis filters in terms of the analysis filters. In some cases only the first analysis filter, H_0, is designable, while the second analysis filter, H_1, as well as the two synthesis filters, F_0 and F_1, are defined in terms of H_0. This is the case for Johnston's filters [9]. In other cases, all four filters are defined in terms of a single prototype filter [8], [10], [11]. However in most cases, while the analysis filters are both designable, the synthesis filters are chosen to cancel aliasing, and are therefore defined in terms of H_0 and H_1 [12], [13], [14]. In terms of design goals, the design techniques also vary widely, although the commonality among the current methods appears to be the specification of frequency-domain constraints on the filters [13], [14], [11], in addition to a possible time-domain constraint [12], [15] on the filters. This is in contrast with both the method proposed in [16] and the proposed FIR model-matching design, which take into account the ideal system (a delay).

Recently, a new approach which decouples the design of the analysis and synthesis banks, was suggested [16]. This approach assumes that the analysis filters (IIR or FIR) have been previously designed. The objective is then to design the synthesis filters in order to minimize the three main contributions to non-perfect reconstruction. This method takes the ideal transfer function into account, and derives an LTI error system. In order to quantify the error between the designed system and the ideal system, the $\mathcal{H}_\infty$ norm is used.

The resulting $\mathcal{H}_\infty$ problem is solved analytically using two Riccati equations. This is the case in [16]; however, the FIR model-matching method requires numerical

optimization. The convex nature of the problem guarantees that the global minimum will be found. Hence, performance limits of the desired setup can be determined. One of the main advantages of decoupling the design of the analysis filters from that of the synthesis filters is increased flexibility; there are no restrictions on the analysis filters. The proposed FIR model-matching method is easily modified to include specifications like linear-phase synthesis filters, cosine-modulated filters, or an alias-free system. Another advantage of the FIR model-matching technique is that as with the methods in [15] and [10], low-delay systems are easily handled.

1.3 Organization of This Thesis

Chapter 2 reviews some multirate fundamentals such as decimators, expanders, and the polyphase representation. Moreover it provides a detailed survey of the traditional QMF design methods. The drawbacks and limitations of the various methods are discussed, thereby providing the motivation for a more general approach to QMF design.

Developing an LTI error system is key to the proposed multirate filter bank design method. Chapter 3 describes the proposed FIR $\mathcal{H}_\infty$ model-matching method in detail. In [16] an LTI error system was derived for the 2-channel QMF bank. After a brief summary of this approach, the FIR problem formulation is described in detail. The performance measures used to determine filter bank system performance are also discussed.

Originally the FIR model-matching method was intended for QMF systems. However, this method is easily extended to other multirate filter bank model-matching

problems. This idea is developed in Chapter 4, where several applications are discussed; namely, QMF banks, fast A/D converters implemented using filter banks, and transmultiplexers. Chapter 4 also illustrates the ease with which the effects of various variables can be studied. For a given setup, the effects of increasing filter length and increasing system delay are easily observed.

One of the main advantages of the FIR model-matching method is the ease with which constraints are handled. Chapter 5 explores this feature. Four kinds of constraints are discussed: linear-phase filters, cosine-modulated filters, low-delay systems, and alias-free systems. In general, imposing constraints on a system degrades the cost function. The FIR model-matching method facilitates studying such tradeoffs because the analysis filters and system delay can be held constant, while various constraints are added to the design problem.

Chapter 6 deals with the $\mathcal{H}_2$ norm and its applications to multirate filter design. There appear to be two main applications for this norm. First, it can be used as a performance measure of the overall system. Second, it can be used as a measure of sensitivity to quantization noise. After an interpretation of the $\mathcal{H}_2$ norm on the time-varying error system, a design example using the $\mathcal{H}_2$ norm as a performance measure is presented. Quantization noise is also discussed in Chapter 6. In addition to justifying the use of the $\mathcal{H}_2$ norm as a measure of sensitivity to quantization noise, a design example is given which illustrates the tradeoff between high quality reconstruction and low sensitivity to quantization noise.

Finally, in Chapter 7 a summary of the work is presented along with some suggestions for future work.

CHAPTER 2

MULTIRATE SYSTEMS

Many systems and fields, such as communications, speech processing, image compression, antenna, analog voice privacy, spectrum analysis, and radar have been beneficiaries of multirate digital filters and filter banks [4]. Multirate systems reduce sampling rates at various points within a system, thereby improving the efficiency of processing signals. The polyphase representation, not only reduces the total number of multiplications and additions per unit time, but also ensures that each of the filters is continually operative [7]. The sampling rate is altered using decimators and expanders.

2.1 Multirate Building Blocks

The two most fundamental building blocks in multirate systems are the decimator and the expander, also referred to as downsampler and upsampler, respectively [17]. The M-fold decimator, shown in Figure 2.1, reduces the sampling rate by taking an

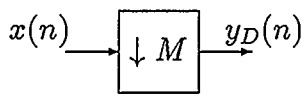

Figure 2.1 The M-Fold Decimator.

input signal and discarding every sample of $x(n)$, except those occurring at multiples

of M:

$$y_D(n) = x(Mn), \tag{2.1}$$

where M is an integer.

The Fourier transform of equation (2.1) is given by [17]

$$Y_D(e^{j\omega}) = \frac{1}{M} \sum_{i=0}^{M-1} X(e^{j(\frac{\omega}{M} - \frac{2\pi i}{M})}).$$

Thus, decimation results in a "stretching" by a factor M in the frequency domain, as illustrated in Figure 2.2. One of the consequences of decimation is that aliasing

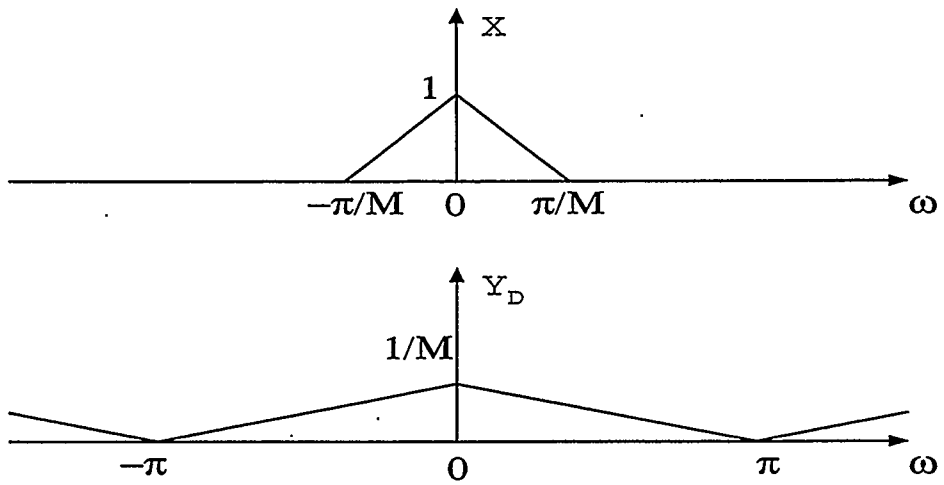

Figure 2.2 The M-Fold Decimator in the Frequency Domain.

occurs if the input signal is not sufficiently bandlimited. Aliasing can be prevented by ensuring that the input signal is of the form

$$X(e^{j\omega}) = 0, \quad \omega_N \leq |\omega| \leq \pi,$$

and $2\pi/M \geq 2\omega_N$, where ω_N is the cutoff frequency.

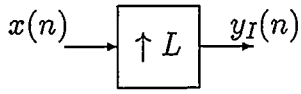

Figure 2.3 The L-Fold Expander.

The role of the L-fold expander, shown in Figure 2.3, is to increase the sampling rate by an integer factor L. This is accomplished by inserting zeros:

$$y_I(n) = \begin{cases} x(\frac{n}{L}) & n = 0, \pm L, \pm 2L, \ldots \\ 0 & \text{otherwise.} \end{cases}$$

The effect of the expander in the frequency domain is a "compression" of the signal. This idea is shown in Figure 2.4. Clearly, the number of copies of the original signal

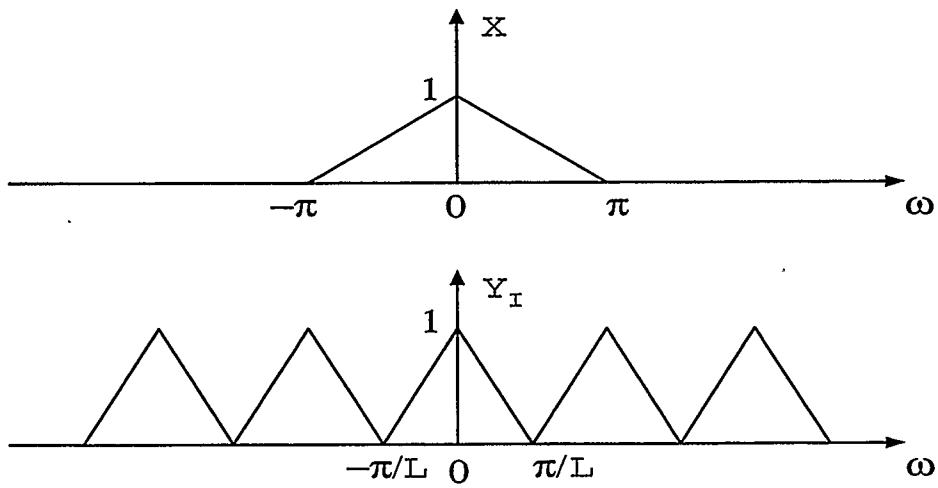

Figure 2.4 The L-Fold Expander in the Frequency Domain.

has increased. The extra copies are called images, and are easily removed by a lowpass filter.

2.2 Polyphase Representation

Introduced in the mid 1970's [18], the polyphase representation was an important development for multirate signal processing; not only does it greatly simplify theoretical results, but it also allows for computationally efficient filters and filter banks (both single and multirate) [4].

The polyphase representation groups the coefficients of a filter. The simplest case would be to separate the odd and even numbered coefficients. This idea is generalized as [7],

$$\begin{aligned} H(z) &= \textstyle\sum_{n=-\infty}^{\infty} h(nM)z^{-nM} \\ &+ z^{-1}\textstyle\sum_{n=-\infty}^{\infty} h(nM+1)z^{-nM} \\ &\vdots \\ &+ z^{-(M-1)}\textstyle\sum_{n=-\infty}^{\infty} h(nM+M-1)z^{-nM}, \end{aligned}$$

which can be rewritten as

$$H(z) = \sum_{l=0}^{M-1} z^{-l} E_l(z^M), \tag{2.2}$$

where

$$E_l(z) = \sum_{n=-\infty}^{\infty} e_l(n) z^{-n},$$

$$e_l(n) = h(Mn+l), \quad 0 \le l \le M-1.$$

Equation (2.2) is referred to as Type 1 polyphase representation [7]. Similarly, the Type 2 polyphase representation is defined as

$$H(z) = \sum_{l=0}^{M-1} z^{-(M-1-l)} R_l(z^M). \tag{2.3}$$

The polyphase representations of the analysis and synthesis bank in the QMF

setup follow from (2.2) and (2.3). Specifically, for the 2-channel analysis bank,

$$H_k(z) = E_{k0}(z^2) + z^{-1}E_{k1}(z^2) \quad k = 0, 1,$$

which can be rewritten as,

$$\begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix} = \begin{bmatrix} E_{00}(z^2) & E_{01}(z^2) \\ E_{10}(z^2) & E_{11}(z^2) \end{bmatrix} \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix} = E(z^2) \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix}.$$

Similarly, using the Type 2 polyphase representation for the synthesis bank yields [7]

$$\begin{bmatrix} F_0(z) & F_1(z) \end{bmatrix} = \begin{bmatrix} z^{-1} & 1 \end{bmatrix} \begin{bmatrix} R_{00}(z^2) & R_{10}(z^2) \\ R_{01}(z^2) & R_{11}(z^2) \end{bmatrix} = \begin{bmatrix} z^{-1} & 1 \end{bmatrix} R(z^2). \tag{2.4}$$

2.3 Traditional QMF Designs

QMF designs can be categorized into Perfect Reconstruction (PR) and Near Perfect Reconstruction (NPR) designs. PR involves an analytical solution developed by placing restrictions on the analysis and synthesis filters. NPR, on the other hand, generally involves optimization to minimize the factors contributing to non-perfect reconstruction. In both PR and NPR designs, coding/decoding errors and filter roundoff noise are neglected.

2.3.1 Perfect Reconstruction

PR implies that the reconstructed signal $\hat{x}(n)$ is a scaled and delayed version of the input signal $x(n)$. It is relatively easy to obtain PR QMF banks, but the problem

is that restrictions are placed on the system; moreover, in some cases the filters are trivial and not useful.

One of the simplest PR examples is as follows. The analysis filters are related by

$$H_1(z) = H_0(-z), \tag{2.5}$$

and the synthesis filters are chosen to be

$$F_0(z) = H_0(z), \quad F_1(z) = -H_1(z). \tag{2.6}$$

This choice of synthesis filters eliminates aliasing. Thus, the system becomes LTI with transfer function, $T(z)$. Recalling that the polyphase representation for H_0 would be

$$H_0(z) = E_{00}(z^2) + z^{-1}E_{01}(z^2),$$

it can be shown that [7]

$$T(z) = 2z^{-1}E_{00}(z^2)E_{01}(z^2). \tag{2.7}$$

Equation (2.7) holds for any QMF bank for which the filters are related by (2.5) and (2.6). The implication for an FIR H_0 is that in order to obtain PR, $E_{00}(z)$ and $E_{01}(z)$ must be delays. If $E_{00}(z) = c_0 z^{-n_0}$ and $E_{01}(z) = c_1 z^{-n_1}$, then

$$H_0(z) = c_0 z^{-2n_0} + c_1 z^{-(2n_1+1)}, \quad H_1(z) = c_0 z^{-2n_0} - c_1 z^{-(2n_1+1)}.$$

Unfortunately, these filters cannot have sharp cutoffs or good stopband attenuations. Alternatively, choosing $E_{01}(z) = \frac{1}{E_{00}(z)}$ results in PR, but now the filters are IIR and may be unstable.

Another, more general, necessary and sufficient condition for PR of a 2-channel QMF bank is [7],

$$E(z)R(z) = cz^{-m_0}\begin{bmatrix} \mathbf{0} & \mathbf{I}_{2-r} \\ z^{-1}\mathbf{I}_r & \mathbf{0} \end{bmatrix}, \tag{2.8}$$

where r is an integer ($0 \leq r \leq 1$), m_0 is an integer, and c is a non-zero constant. The delay of the overall system is $m = 2m_0 + r + 1$. The result of (2.8) is a condition on the determinant:

$$\det R(z) \det E(z) = c_0 z^{-k_0}, \quad c_0 \neq 0, \quad k_0 = \text{integer}. \tag{2.9}$$

All (both FIR and IIR) PR QMF systems must satisfy (2.9). Thus, for an FIR QMF system,

$$\det E(z) = \alpha z^{-K}, \quad \alpha \neq 0, \quad K = \text{integer},$$

and a similar requirement holds for $R(z)$.

The PR condition is very restrictive; all QMF banks must satisfy (2.9). Moreover, in some cases, additional constraints are placed on the filters. At the expense of the PR requirement, NPR methods have been suggested to relax the restrictions on the filters.

2.3.2 Near Perfect Reconstruction

The NPR methods can basically be broken down into three sections, according to which filters are designable:

- Design only one analysis filter
- Design analysis filters
- Design analysis and synthesis filters.

The earliest NPR methods involved minimizing an error criterion composed of a weighted sum of the passband (E_r) and the stopband (E_{sb}) energies of the analysis

filter H_0,

$$J = E_r + \alpha E_{sb}. \tag{2.10}$$

The remaining filters were completely determined by H_0:

$$\begin{aligned} H_1(z) &= H_0(-z), \\ F_0(z) &= H_0(z), \\ F_1(z) &= -H_0(-z). \end{aligned}$$

This was the approach used in [13] and [19]. The approaches in [13] and [19] differed only in the optimization step. The Hooke and Jeeves optimization was used in [13]. This particular routine requires careful choice of the initial starting point, and may require manual intervention to avoid becoming trapped in a local minimum. The approach in [19] is an improvement over the technique in [13] because the optimization requires no manual intervention and will converge to the true minimum. Reference [13] reformulates (2.10) into a quadratic time-domain cost function subject to a quadratic constraint, the solution of which is well known. The optimization involves searching for a minimum eigenvalue.

Another approach is to reconfigure the design problem into a quadratic program with some constraints. Then the design problem becomes minimizing $\Phi(y)$ subject to some constraints, where

$$\Phi(y) = \frac{1}{2} y^T Q y + p^T y + d,$$

and y is a vector containing the impulse response coefficients of $h_1(n)$. This is the cost function used in [10] and [14]. In [10] linear constraints are used. Initially, $H_0(z)$ is independently designed, after which $H_1(z)$ is optimized according to some frequency considerations, such as passband ripple and stopband energy. The design in [10] is based on the first design presented in [14]. The improvement offered by [10] is that

low-delay systems can be accommodated. The synthesis filters are chosen as

$$\begin{aligned} F_0(z) &= 2H_1(-z), \\ F_1(z) &= -2H_0(-z), \end{aligned}$$

to cancel aliasing. Furthermore, a pure-delay constraint is imposed on the system:

$$H_0(z)H_1(z) - H_1(z)H_0(-z) = z^{-2k+1}. \tag{2.11}$$

Having designed $H_0(z)$, its coefficients are used as known variables in (2.11), leaving a set of linear constraint equations for designing $H_1(z)$. One drawback of this method is that the length of $H_1(z)$ must be greater than the length of $H_0(z)$ ($N_1 > N_0$). The reason for this is that for equal length filters, the degrees of freedom in designing $H_1(z)$ reduce to zero.

The Lagrange-Newton method discussed in [14] addresses this problem. The Lagrange-Newton method optimizes H_0 and H_1 simultaneously. In this case the design problem becomes minimizing $\Phi(y)$ subject to a set of nonlinear constraints.

A more general method, which designs all four filters at once is introduced in [15], and further elaborated in [12] and [20]. This approach combines the reconstruction error of the QMF bank and various frequency error terms into a single cost function. The reconstruction error involves several matrices related to the analysis filters, synthesis filters, and the desired system delay. The Frobenius norm is used to quantify the reconstruction error. The cost function used in [15] is

$$J = w_1\varepsilon_r + \sum_{j=2} w_j\varepsilon_j,$$

where ε_r is strictly associated with the reconstruction error, ε_j are specific frequency errors associated with the analysis filters, such as passband ripple and stopband

ripple, and ω_j are the weighting factors. This method is based on Powell's algorithm. The key to the approach in [15] is that the analysis filters are the parameters of the optimization process. For a given set of analysis filters, the synthesis filters are optimized to minimize the cost function J. If the minimum cost function is too large, a new set of analysis filters is determined. This process is repeated until some acceptable cost function has been reached. The method in [15] is quite flexible in terms of the length of the filters, types of filters, and system delay. One other noteworthy point regarding this method is that it is possible to use it to design cosine-modulated filters.

Low implementation cost has made using cosine-modulated filters in QMF banks an attractive option. These QMF systems, first proposed in [21] and later expanded on in [22], are referred to as pseudo-QMF systems. In pseudo-QMF systems, the analysis and synthesis filters are cosine-modulated versions of a prototype filter.

Various techniques have been suggested for cosine-modulated filter banks, but these apply strictly to cosine-modulated filter banks. In [23] the lowpass prototype filter is found using the optimization technique developed by Johnston, as described in [19]. In [8] the prototype filter $P(z)$ is a linear-phase spectral factor of a 2M-th band filter, $G(z)$:

$$G(z) = z^{-(N-1)}P(z)P(z^{-1}),$$

where N is the number of taps in the prototype filter. The amplitude and phase distortions are completely eliminated, leaving only aliasing, which can be minimized by finding solutions with high stopband attenuation.

In [11] the prototype filter is also a spectral factor of a 2M-th band filter. However, in this case, the prototype does not have linear phase. Furthermore, the magnitude response of the overall transfer function (neglecting aliasing) has dips and bumps around $\omega = 0$ and $\omega = \pi$. This may be undesirable in applications like image processing. A key assumption of the method in [11] is that the filters belonging to nonadjacent channels do not significantly overlap. Nevertheless, the advantage of this method is that no optimization is required.

Table 2.1 shows a comparison of the various methods.

Reference	Restrictions on Filters	Linear Phase	Cosine-Modulated	Low-Delay System
[13], [19]	$F_0(z) = H_0(z)$ $F_1(z) = -H_1(z)$ $H_1(z) = H_0(z)$	only	no	no
[12], [15], [20]		optional	optional	optional
[10]	$F_0(z) = 2H_1(-z)$ $F_1(z) = -2H_0(-z)$ $N_1 > N_0$	only	no	optional
[8], [23]	equal length $F_k(z) = z^{-(N-1)}H_k(z^{-1})$	prototype: linear-phase	only	no
[11]	equal length $F_k(z) = z^{-(N-1)}H_k(z^{-1})$ non-adjacent filters: no overlap	prototype: non-linear-phase	only	no

Table 2.1 Comparison of Various NPR Design Methods

The method introduced in [15] appears to be the most flexible. One restriction shared by all the methods mentioned in Table 2.1 is that all the filters must be FIR.

Along with the flexibility in specifying constraints, the proposed FIR model-matching method does not have any restrictions on the analysis filters. In terms of flexibility, the proposed method is quite similar to the time-domain approach presented in [15]; the main difference being the ability to study tradeoffs. In [15] the analysis filters are variable in the optimization process, however in the FIR model-matching method the analysis filters are held constant. For this reason the FIR model-matching method provides an effective tool to study the effects of adding constraints to the filters (linear-phase, cosine-modulated) or to the system (alias-free, low-delay).

CHAPTER 3

FIR $\mathcal{H}_\infty$ MODEL-MATCHING METHOD

The idea of using model-matching in QMF bank design was first proposed in [16]. This idea forms the basis of the FIR $\mathcal{H}_\infty$ model-matching method. Optimization must be used in order to obtain the FIR synthesis filters. The convex nature of the problem ensures that there is only one minimum. Consequently, the model-matching setup is an ideal choice for studying performance limits. Matching the QMF bank to an ideal delay results in a time-varying system, which can be transformed into an LTI error system.

3.1 Developing the Error System

Ideally, the reconstructed signal $\hat{x}(n)$ out of the QMF bank is a delayed version of the input signal $x(n)$:

$$\hat{x}(n) = x(n-m).$$

In the z-domain:

$$\hat{X}(z) = z^{-m}X(z),$$

and consequently, the ideal system is linear time-invariant (LTI) with transfer function

$$T_d(z) = z^{-m}.$$

As a means of quantifying the difference between the reconstructed signal $\hat{x}(n)$ and the input $x(n)$, an error system was proposed in [16]. This error system is shown in Figure 3.1, the reconstruction error being $e(n) = x(n-m) - \hat{x}(n)$.

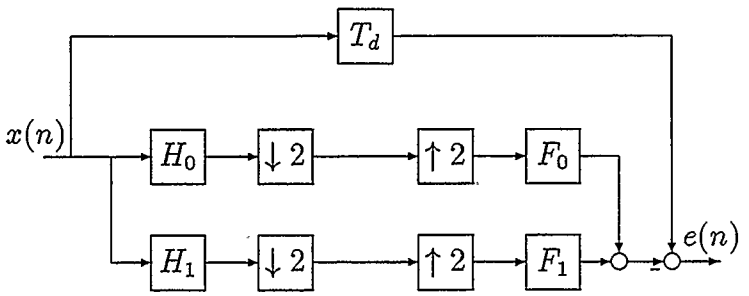

Figure 3.1 Time-Varying Error System.

As a measure of performance, the ℓ_2-induced norm from $x(n)$ to $e(n)$ is used,

$$J := \sup_{\|x\|_2 \neq 0} \frac{\|e\|_2}{\|x\|_2}, \tag{3.1}$$

so that J is the worst-case error-to-signal ratio for all possible finite energy input signals. Since this is a worst-case norm, the nature of the norm guarantees that if J is small, then the error will be uniformly small over all inputs, $x(n)$. Furthermore, it can be shown [16] that the optimal J will be in the range [0,1], where $J_{opt} = 0$ implies perfect reconstruction and $J_{opt} = 1$ occurs when $F_0 = F_1 = 0$. By comparing the QMF bank to an ideal time delay system, the synthesis filters can be designed for optimal reconstruction for a given choice of analysis and delay.

The method in [16] takes advantage of the fact that the ℓ_2-induced norm of a system is equivalent to the $\mathcal{H}_\infty$ norm of its transfer function, $G(z)$,

$$\sup_{\|x\|_2=1} \|e\|_2 = \|G\|_\infty = \sup_{0 \leq \omega \leq 2\pi} \|G(e^{j\omega})\|. \tag{3.2}$$

Equation (3.2) applies only to stable, causal, LTI systems. For such systems which are single-input single-output (SISO), the interpretation of the ℓ_2-induced norm is straightforward: The ℓ_2-induced norm is the peak gain of the magnitude response of

the transfer function, over all frequencies. Due to the upsamplers and downsamplers, the error system in Figure 3.1 is, in general, not LTI. However, the system is periodically time-varying. This property may be exploited in order to achieve an LTI system. An equivalent LTI system is derived in [16] using polyphase representations and blocking. Given a scalar signal $x(n)$, its 2-fold blocked version $\mathbf{x}(n)$ is given by [24]

$$\mathbf{x}(n) = \begin{bmatrix} x(2n) \\ x(2n-1) \end{bmatrix}.$$

The end result is the LTI error system shown in Figure 3.2. Since the polyphase

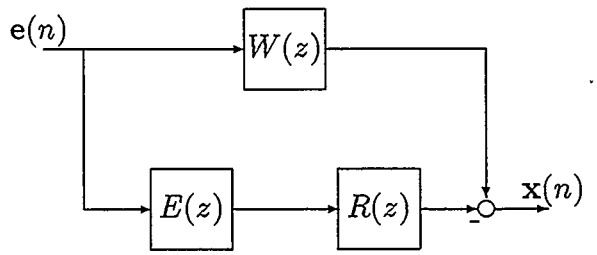

Figure 3.2 Time-Invariant Error System.

representation and blocking preserve the ℓ_2 norm, the ℓ_2-induced norm of the periodically time-varying system in Figure 3.1 equals the ℓ_2-induced norm of the LTI system in Figure 3.2.

$E(z)$ and $R(z)$ are the 2×2 transfer matrices corresponding to the polyphase representations of the analysis and synthesis filters, respectively, as defined in Section 2.2. $W(z)$ is related to the delay m, as shown below,

$$W(z) = \begin{cases} z^{-d} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \text{if } m = 2d+1 \\ z^{-d} \begin{bmatrix} 0 & z \\ 1 & 0 \end{bmatrix} & \text{if } m = 2d. \end{cases}$$

Given a delay m, and a set of analysis filters, E and W are completely defined. Therefore, the only quantity remaining to be designed is a stable and causal R to minimize the performance measure J,

$$J = \inf_{R(z)} \|W - RE\|_\infty.$$

This is an $\mathcal{H}_\infty$-model-matching problem. The solution in [16] is based on solving Riccati equations. The drawbacks of this solution are 1) it always yields IIR synthesis filters; 2) it does not allow for the specification of any constraints, such as a linear phase requirement or an alias-free specification. The proposed FIR $\mathcal{H}_\infty$ model-matching method attempts to address these issues. This new method offers more flexibility in terms of the number of taps in the synthesis filters, filter type (linear-phase, cosine-modulated), as well as constraints on the system such as an alias-free requirement.

3.2 FIR Problem Formulation

The basic setup of the FIR $\mathcal{H}_\infty$ model-matching method is the same as in [16], although in this case, R is related to FIR synthesis filters. The synthesis filters, F_0 and F_1, are related to R by the polyphase representation, as given in (2.4:

$$\begin{bmatrix} F_0(z) & F_1(z) \end{bmatrix} = \begin{bmatrix} z^{-1} & 1 \end{bmatrix} R(z^2).$$

This leads to a very simple relationship which can then in turn be used to translate constraints on the synthesis filters into constraints on R. One important consequence of this relationship is that the number of taps in the synthesis filters must always be even. Odd length synthesis filters are still possible by simply constraining the last coefficient of the synthesis filters to be zero.

Defining an FIR R as,

$$R(z) = \begin{bmatrix} r_{11}(0) & r_{12}(0) \\ r_{21}(0) & r_{22}(0) \end{bmatrix} + \cdots + \begin{bmatrix} r_{11}(\frac{n_{tap}}{2}-1) & r_{12}(\frac{n_{tap}}{2}-1) \\ r_{21}(\frac{n_{tap}}{2}-1) & r_{22}(\frac{n_{tap}}{2}-1) \end{bmatrix} z^{-(\frac{n_{tap}}{2}-1)}. \quad (3.3)$$

Then the synthesis filters, F_0 and F_1, are also FIR and are given by,

$$F_0(z) = r_{21}(0)+r_{11}(0)z^{-1}+r_{21}(1)z^{-2}+r_{11}(1)z^{-3}+\cdots+r_{11}(\frac{n_{tap}}{2}-1)z^{-(n_{tap}-1)} \quad (3.4)$$

and

$$F_1(z) = r_{22}(0)+r_{12}(0)z^{-1}+r_{22}(1)z^{-2}+r_{12}(1)z^{-3}+\cdots+r_{12}(\frac{n_{tap}}{2}-1)z^{-(n_{tap}-1)}. \quad (3.5)$$

Another observation is that the number of taps in R is actually half the number of taps in F_0 and F_1. Since R is a 2×2 transfer matrix, the number of variables to be optimized is

$$N = \frac{n_{tap}}{2} \times 4.$$

To summarize, the initial FIR design problem is as follows:

> For a given set of analysis filters, H_0 and H_1, and delay, m, design the n_{tap}-tap synthesis filters, F_0 and F_1, of the form in (3.4) and (3.5), by minimizing (3.1).

The initial problem is then converted to the equivalent $\mathcal{H}_\infty$ model-matching optimization problem:

For W and E as before, design R, of the form in (3.3),

by minimizing $\|W - RE\|_\infty$.

This is a finite-dimensional convex optimization problem, which guarantees there to be only one minimum. As a result, a local optimization method can be used. The ellipsoid algorithm is a suitable choice for the optimization. Although it is slow compared to some other algorithms, three main reasons for choosing the ellipsoid algorithm are: First, the algorithm can find the minimum to a guaranteed accuracy; second, it is easily modified to include convex constraints; and third, the ellipsoid algorithm works for non-differentiable functions.

3.2.1 Ellipsoid Algorithm

The ellipsoid algorithm will determine the minimum of a convex function, $\phi(x)$, to a specified accuracy. The basic philosophy behind the ellipsoid algorithm is to generate a decreasing sequence of ellipsoids, which are guaranteed to contain a minimizing point [25]. Centered at x_k, the ellipsoid is specified with positive definite matrix A_k. More explicitly, an ellipsoid $\mathbf{E}$ is given by

$$\mathbf{E} = \left\{ x \,\middle|\, (x - x_k)^T A_k^{-1} (x - x_k) \leq 1 \right\}.$$

The ellipsoid algorithm uses a subgradient, g_k, of the function to be minimized to determine a half-space that is guaranteed to contain the minimizers. Next, a minimum volume ellipsoid, $\mathbf{E_{k+1}}$, is determined, which contains the half ellipsoid of the previous iteration, $\mathbf{E_k}$. This process is then repeated, until the minimizers x_k and corresponding minimum of $\phi(x_k)$ have been found to the desired accuracy.

The ellipsoid algorithm is guaranteed to converge to the global minimum, provided that the initial ellipsoid, $\mathbf{E_1}$, contains a minimizing point. Often $\mathbf{E_1}$ is chosen to be a sufficiently large ball with $A_1 = R^2\mathbf{I}$, where R refers to the radius, and $\mathbf{I}$ is the $N \times N$ identity matrix. The radius of the initial sphere depends on the size of the coefficients, x_k. Furthermore, the initial guess for the coefficients is arbitrary, although a "good" guess will result in faster convergence. More explicitly, the basic ellipsoid algorithm is as follows:

1. Enter stopping criterion, initial guess of minimizers, and initial sphere.
2. Compute any subgradient of $\phi(x_k)$ at the current guess of minimizers.
3. Update minimizers.
4. Calculate new minimum volume ellipsoid based on half-space determined by subgradient.
5. Evaluate stopping criterion at new minimizers.
6. If stopping criterion is met stop; otherwise go to Step 2.

A more detailed description is given in [25].

For the design problem at hand, the convex function ϕ is the performance measure J. Moreover, the vector x used in the minimization is composed of the coefficients in the FIR $R(z)$. The initial guess, x_1, for the minimizers is the truncated result obtained from the IIR method in [16]. The stopping criterion is composed of two parts. The first part is the accuracy to which the minimum should be determined. The second stopping criterion exploits the fact that the minimum must be in [0,1].

Hence, the user may specify a tolerable minimum, after which the algorithm may stop. When the ellipsoid algorithm is invoked, the user must specify an error tolerance ε_{tol} as well as a tolerable minimum J_{tol}.

3.2.2 Calculating Subgradient

The ellipsoid algorithm requires calculation of the subgradient. The subgradient of the $\mathcal{H}_\infty$ norm of a multi-input, multi-output (MIMO) function, G (in this case $G = W - RE$) is as follows [25]. First, G is evaluated at the current guess for the minimizers to yield G_c. Next, the $\mathcal{H}_\infty$ norm of G_c is calculated. For stable G_c, the $\mathcal{H}_\infty$ norm of G_c is defined as the maximum singular value over all frequencies:

$$\|G_c\|_\infty = \sup_\omega \{\sigma_{max}[G_c(j\omega)]\}.$$

The objective of this calculation is to find the frequency, ω_c, at which the $\mathcal{H}_\infty$ norm is achieved. Next, the singular value decomposition must be calculated,

$$G_c(j\omega_c) = U\Sigma V^*.$$

Choosing u_c to be the first column of U, v_c to be the first column of V^*, and defining the linear functional

$$\phi^{u,v,\omega}(G) = \Re[u^* G(j\omega) v],$$

where $\Re(z)$ is defined as the real part of a complex number z. Then a subgradient at G_c is given by the linear functional

$$\phi^{sg} = \phi^{u_c,v_c,\omega_c}.$$

In the case of FIR synthesis filters, the problem involves a finite-dimensional subspace. Specifically, we have a convex functional ϕ, in this case the $\mathcal{H}_\infty$ norm, acting

on a finite set of transfer matrices $G_0, G_1, \ldots, G_N$. To determine the subgradient, consider the convex function [25],

$$\varphi(x) = \phi(G_0 + x_1 G_1 + \cdots + x_N G_N) \tag{3.6}$$

Where N refers to the number of variables ($N = \frac{n_{tap}}{2} \times 4$). The subgradient of (3.6) is then given by,

$$g = \begin{bmatrix} \phi^{sg}(G_1) \\ \vdots \\ \phi^{sg}(G_N) \end{bmatrix}.$$

For the FIR optimization at hand,

$$\|W - RE\|_\infty,$$

or more explicitly,

$$\|W - \left(\begin{bmatrix} r_{11}(0) & r_{12}(0) \\ r_{21}(0) & r_{22}(0) \end{bmatrix} + \cdots + \begin{bmatrix} r_{11}(\frac{n_{tap}}{2}-1) & r_{12}(\frac{n_{tap}}{2}-1) \\ r_{21}(\frac{n_{tap}}{2}-1) & r_{22}(\frac{n_{tap}}{2}-1) \end{bmatrix} z^{-(\frac{n_{tap}}{2}-1)} \right) E\|_\infty.$$

This must then be modified in order to get it into the format of (3.6):

$$\|W - \left(r_{11}(0) \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + r_{12}(0) \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \cdots + r_{22}(\frac{n_{tap}}{2}-1) \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} z^{-(\frac{n_{tap}}{2}-1)} \right) E\|_\infty$$

Finally, substituting

$$z = \frac{1 + j\omega_0}{1 - j\omega_0},$$

yields an equation of the form in (3.6), where

$$
\begin{aligned}
G_0 &= W\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right) \\
G_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right) \\
G_2 &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right) \\
G_3 &= \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right) \\
G_4 &= \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right) \\
&\vdots \\
G_N &= \begin{bmatrix} 0 & 0 \\ 0 & \left(\tfrac{1+j\omega_c}{1-j\omega_c}\right)^{-(\frac{n_{tap}}{2}-1)} \end{bmatrix} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right).
\end{aligned}
$$

To summarize, the FIR $\mathcal{H}_\infty$ model-matching method requires:

- Number of taps in the synthesis filters
- Delay
- Bound, ε_{tol}, on the distance to the minimum
- Maximum tolerable cost function, J_{tol}.

3.3 Other Performance Measures

The ℓ_2-induced norm is only one of many ways to determine the performance of a filter bank system. One simple and common metric is to observe the response to a ramp input [15], [14]. Since the $\mathcal{H}_\infty$ norm is a worst-case norm over all possible

inputs, J_{opt} will provide an upper bound on the energy ratio of the error to the ramp input.

Another measure of the performance of the overall system is to look at the amplitude and phase distortions [11]. The reconstructed signal, $\hat{x}(n)$, is given by [7],

$$\begin{aligned}\hat{X}(z) &= \frac{1}{2}[H_0(z)F_0(z) + H_1(z)F_1(z)]X(z) + \frac{1}{2}[H_0(-z)F_0(z) + H_1(-z)F_1(z)]X(-z) \\ &= T(z)X(z) + A(z)X(-z). \end{aligned} \tag{3.7}$$

The second term, $A(z)$, takes into account aliasing due to the decimators and imaging due to the expanders, and is referred to as the alias component. Clearly, this term should be small. Furthermore, if aliasing is cancelled (i.e., $A(z) = 0$), the system becomes LTI with transfer function $T(z)$. Ideally, $T(z)$ has unity magnitude response and a linear phase response. The "distortion plots" show the magnitude and phase response of $T(z)$ and $A(z)$, and give an indication of the performance of QMF filter bank.

Yet another approach to verifying a particular design is to use the signal-to-noise ratio (SNR_r) [12], [15]. The SNR_r is defined for a ramp input,

$$x_{ramp}(n) = \begin{cases} n, & 0 \le n \le 99 \\ 0, & \text{otherwise,} \end{cases}$$

and is calculated using

$$\text{SNR}_r = 20 \times \log\left(\frac{\|x_{ramp}\|_2}{\|e\|_2}\right),$$

where the error is defined as the difference between the delayed input $x_{ramp}(n-m)$ and the reconstructed signal $\hat{x}(n)$. The $\mathcal{H}_\infty$ norm provides a worst case signal-to-noise ratio for all possible input signals since,

$$J = \sup_x \frac{\|e\|_2}{\|x\|_2} = \left(\inf_x \frac{\|x\|_2}{\|e\|_2}\right)^{-1}.$$

Thus, if the system is optimal according to the $\mathcal{H}_\infty$ norm,

$$\mathrm{SNR}_r \geq -20 \times \log J_{opt}.$$

The reconstruction error ε_r is used in [12], [15], [20], and is defined as

$$\varepsilon_r = \|\mathbf{AS} - \mathbf{B}\|_F^2,$$

where $\mathbf{A}$ and $\mathbf{S}$ are matrices related to the analysis and synthesis filters, respectively. Although the definition in [12], [15], and [20], is for the M-channel case, for simplicity, the following discussion will be restricted to the 2-channel case. The system filters are all assumed to have length N, equal to an integer multiple, L, of the decimation factor (in this case 2). Clearly, arbitrary filter lengths are still possible by setting the appropriate coefficients to zero.

The derivation of this performance metric is based on dividing the QMF system into three subsystems: the analysis section; the down/up sampler; and the synthesis section. Using inner products and matrix vector products, the input and output relations are defined for each subsystem. After a lengthy derivation, [15] derived a complete set of time domain conditions for PR, which in terms of a matrix equation is given by $\mathbf{AS} = \mathbf{B}$. $\mathbf{A}$ and $\mathbf{S}$ are defined as

$$\mathbf{A} = \begin{bmatrix} \mathbf{P}_0^T & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{P}_1^T & \mathbf{P}_0^T & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{P}_2^T & \mathbf{P}_1^T & \mathbf{P}_0^T & & \vdots \\ \vdots & \vdots & \vdots & \ddots & \\ \mathbf{P}_{L-1}^T & \mathbf{P}_{L-2}^T & \mathbf{P}_{L-3}^T & \cdots & \mathbf{P}_0^T \\ \mathbf{0} & \mathbf{P}_{L-1}^T & \mathbf{P}_{L-2}^T & \cdots & \mathbf{P}_1^T \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{L-1} & \mathbf{P}_{L-2}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{L-1}^T \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} \mathbf{Q}_0 \\ \mathbf{Q}_1 \\ \vdots \\ \mathbf{Q}_{L-2} \\ \mathbf{Q}_{L-1} \end{bmatrix},$$

where $\mathbf{P}_i$ and $\mathbf{Q}_i$ are the 2×2 submatrices of $\mathbf{P}$ and $\mathbf{Q}$:

$$\mathbf{P} = \begin{bmatrix} h_0(0) & h_0(1) & \dots & h_0(N-1) \\ h_1(0) & h_1(1) & \dots & h_1(N-1) \end{bmatrix},$$

$$\mathbf{Q} = \begin{bmatrix} f_0(0) & f_0(1) & \dots & f_0(N-1) \\ f_1(0) & f_1(1) & \dots & f_1(N-1) \end{bmatrix}.$$

The matrix $\mathbf{B}$ is related to the delay,

$$\mathbf{B} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \mathbf{J}_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

where $\mathbf{J}_2$ is the 2×2 exchange matrix, given by,

$$\mathbf{J}_2 = \frac{1}{2} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

To quantify the difference between $\mathbf{AS}$ and $\mathbf{B}$, [15] uses the Frobenius norm, defined as

$$\|X\|_F^2 = \sum_i \sum_j |x_{ij}|^2.$$

Since the $\mathcal{H}_\infty$ norm is a worst-case norm, it provides an upper bound on the other performance measures, and consequently, it is a sufficient measure of system performance. However, the more conventional performance measures provide a means for comparing the $\mathcal{H}_\infty$ model-matching method to the results in the literature.

CHAPTER 4

MODEL-MATCHING APPLICATIONS

Although the FIR $\mathcal{H}_\infty$ model-matching method was originally intended for QMF bank design, it is easily extended to other setups. In order to demonstrate the flexibility of this model-matching method, three applications are presented:

- Simple QMF Setup
- A/D Converter
- Transmultiplexer.

4.1 Simple QMF

The example described in this section forms the basis of comparison in Chapter 5, which illustrates the effect of adding constraints. The delay, m, is chosen to be 21 samples and the analysis filters are FIR linear-phase filters of order 19, obtained using a Hamming window. For simplicity (this is not a necessary condition), $H_1(z)$ is selected to be the mirror image of $H_0(z)$, $H_1(z) = H_0(-z)$. The corresponding coefficients of the analysis filters and optimized synthesis filters are shown in Table 4.1. The magnitude plots of the analysis and 24-tap synthesis filters are shown in Figure 4.1.

According to the IIR method in [12], the optimal cost function for this example is $J_{opt} = 0.03$. The corresponding IIR synthesis filters were of order 77. Using the FIR $\mathcal{H}_\infty$ model-matching method, the optimized cost function with the 24-tap synthesis filters is 0.0302. Thus, there is little degradation in terms of the $\mathcal{H}_\infty$ norm.

n	$h_0(n)$	$h_1(n)$	$f_0(n)$	$f_1(n)$
0	0.00188953	0.00188953	-0.00000945	0.00021408
1	0.00276976	-0.00276976	-0.05346902	-0.05306207
2	-0.00529526	-0.00529526	-0.00225164	0.00185754
3	-0.00995577	0.00995577	0.05893342	0.05940328
4	0.01742330	0.01742330	0.00145445	-0.00263743
5	0.02881991	-0.02881991	-0.09873663	-0.09868561
6	-0.04646496	-0.04646496	0.00341369	-0.00348885
7	-0.07642885	0.07642885	0.15513218	0.15479254
8	0.14129435	0.14129435	-0.03700080	0.03703133
9	0.44594798	-0.44594798	-0.25793461	-0.25880044
10	0.44594798	0.44594798	0.20381798	-0.20305760
11	0.14129435	-0.14129435	0.99522530	0.99691096
12	-0.07642885	-0.07642885	0.99673755	-0.99523212
13	-0.04646496	0.04646496	0.20325262	0.20402951
14	0.02881991	0.02881991	-0.25835486	0.25779336
15	0.01742330	-0.01742330	-0.03676981	-0.03708290
16	-0.00995577	-0.00995577	0.15515693	-0.15536993
17	-0.00529526	0.00529526	0.00349532	0.00361137
18	0.00276976	0.00276976	-0.09842892	0.09843233
19	0.00188953	-0.00188953	0.00272826	0.00188061
20	0	0	0.05969376	-0.05904509
21	0	0	-0.00156940	-0.00212076
22	0	0	-0.05281396	0.05336317
23	0	0	-0.00021680	0.00000983

Table 4.1 Coefficients of Filters for Simple QMF Example

As a more conventional measure of performance [15], a ramp input signal was used, and the corresponding error was calculated for the 24-tap synthesis filters. Figure 4.2 confirms that the $\mathcal{H}_\infty$ norm is indeed a worst-case norm since the ratio of the 2-norm of the error to the 2-norm of the input ramp is 0.0202, which is less than the bound established by the $\mathcal{H}_\infty$ norm. Figure 4.3 shows the distortion plots, as defined in Section 3.3, for the 24-tap case. As expected, $T(z)$ is approximately unity with

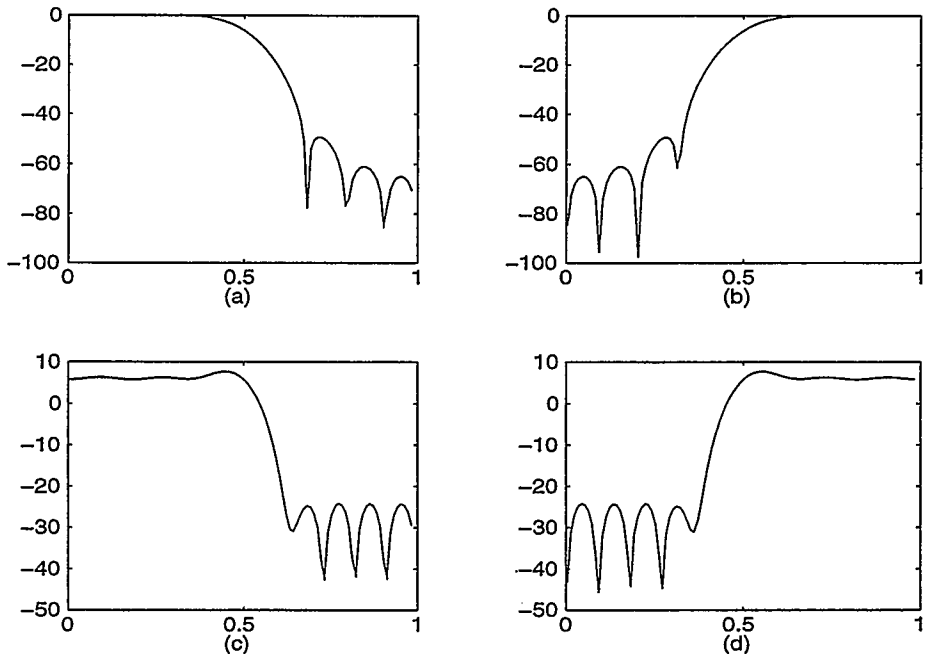

Figure 4.1 Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1.

linear-phase, and $A(z)$ is small.

A summary of the various performance measures described in Section 3.3 is given in Table 4.2. The worst-case values of the three factors contributing to non-perfect reconstruction are also given in Table 4.2. The magnitude error is given by

$$\sup_{0\leq\omega\leq 2\pi} \left| |T(e^{j\omega})| - 1 \right|.$$

The phase error is the maximum difference between the ideal system and the phase of $T(e^{j\omega})$, $\phi(\omega)$:

$$\sup_{0\leq\omega\leq 2\pi} \left| +\omega m \frac{180^\circ}{\pi} \right|.$$

The aliasing error is defined as the maximum value the aliasing component $|A(e^{j\omega})|$.

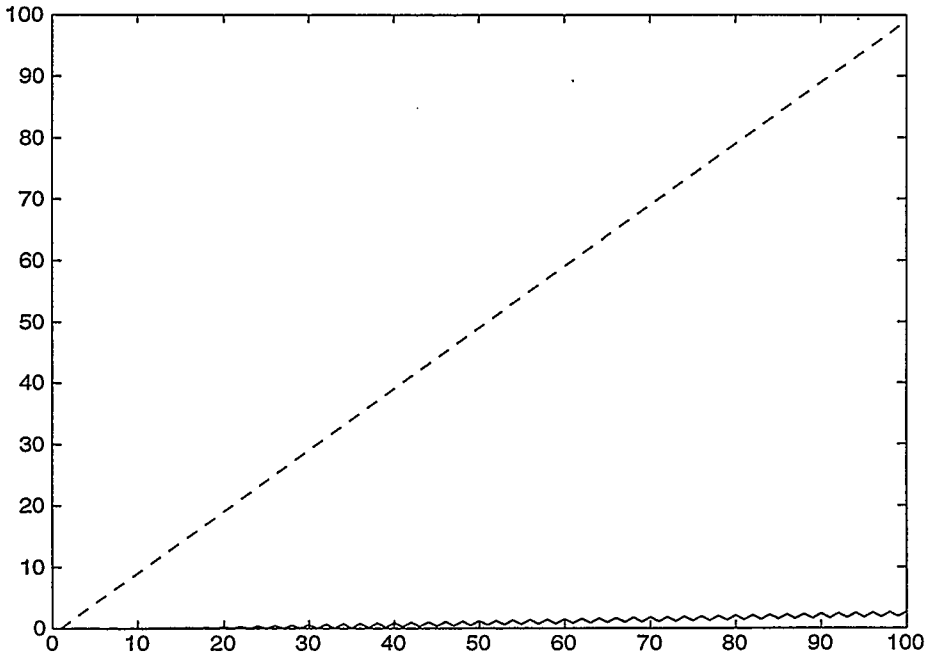

Figure 4.2 Input Ramp (dashed) and Reconstruction Error (solid) versus n.

$\mathcal{H}_\infty$ Norm	SNR_r	$\|\mathbf{AS} - \mathbf{B}\|_F$	Magnitude Error	Phase Error	Aliasing
0.0302	0.020226	0.0017689	0.0302	0.096°	-30.39 dB

Table 4.2 Performance Measures for the Simple QMF Example.

4.2 Fast A/D Converters

One approach to fast A/D conversion is to use multirate filter banks [7]. The basic idea is to perform the A/D conversion at a slow rate on M subband signals, and then upsample these subband signals and recombine them. The net result is a high sampling rate A/D converter operating at an M-fold lower rate. A 2-channel ($M = 2$) example of a fast A/D converter is shown in Figure 4.4.

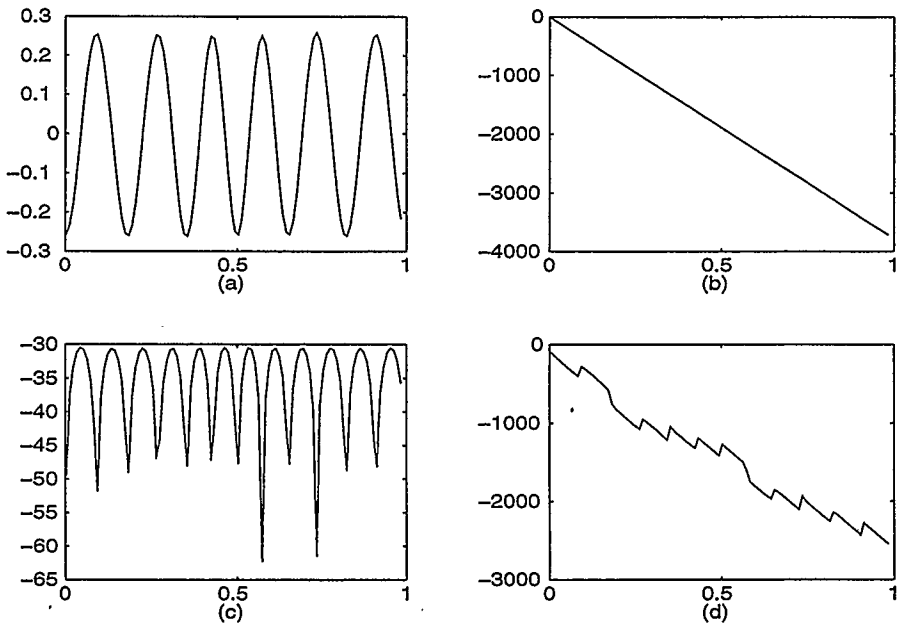

Figure 4.3 Magnitude (dB) Distortions for $T(z)$ and $A(z)$ ((a) and (c)) versus ω/π. Phase (°) Distortions for $T(z)$ and $A(z)$ ((b) and (d)) versus ω/π.

In Figure 4.4 the continuous-time input signal $x_a(t)$ is split into two subband signals using the analog analysis filters $H_0(s)$ and $H_1(s)$. The A/D conversion occurs at half the intended rate. To get the desired rate, the digitized subband signals are upsampled. Finally, the subband signals are recombined through the digital synthesis filters to obtain the digitized version of x_a [7]. Ideally $\hat{x}$ closely resembles x_a digitized at the fast rate. However, due to nonideal filters, as with the traditional QMF setup, aliasing, amplitude distortion, and phase distortion introduce errors.

In [26] the problem is converted into an $\mathcal{H}_\infty$ model-matching problem by reconfiguring the original setup into an error system. In this case the ideal system is the

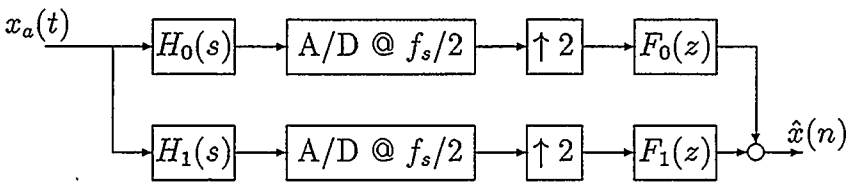

Figure 4.4 An Analog/Digital Hybrid QMF Bank.

fast A/D converter along with some delay. The resulting error system is shown in Figure 4.5. The only purpose of H in Figure 4.5 is to ensure that the input to the

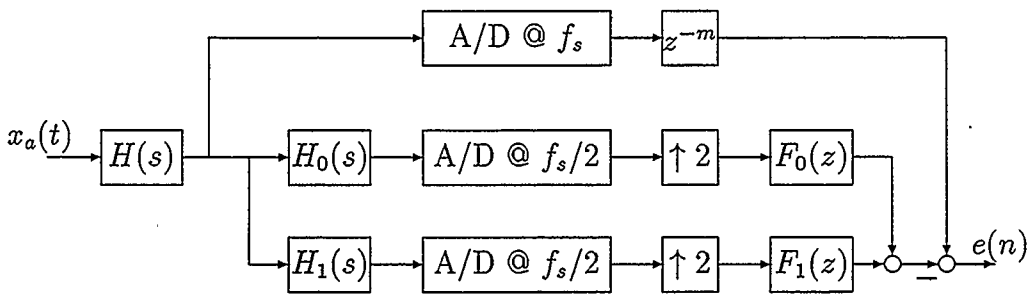

Figure 4.5 Error System for Fast A/D Converter.

filter bank is appropriately bandlimited, thereby removing any restrictions on $x_a(t)$. In fact, H can be thought of as the anti-aliasing filter typically used prior to any A/D conversion. The performance measure was chosen to be

$$J_{A/D} = \sup_{\|x_a\| \neq 0} \frac{\|e\|_2}{\|x_a\|_2},$$

which is similar to the QMF performance measure, except that here x_a is a continuous-time signal.

As with the QMF error system, the original A/D error system is periodically time-varying. In [26] an equivalent discrete-time LTI system, shown in Figure 4.6, was derived. R is still the polyphase representation of the synthesis filters, and W

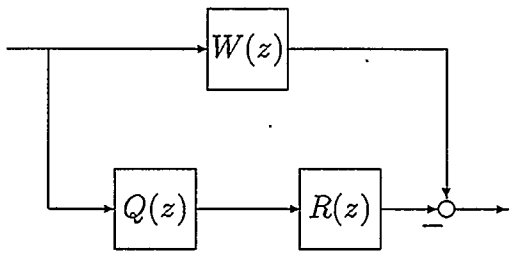

Figure 4.6 Time-Invariant Error System for Fast A/D Conversion.

and Q are related to the continuous time analysis filters. Since the ℓ_2-induced norm of $W - RQ$ is equal to the $\mathcal{H}_\infty$ norm of its transfer matrix, the problem becomes minimizing

$$J = \|W - RQ\|_\infty.$$

This optimization problem was solved in [26] using the MATLAB function *hinfsyn*, resulting in IIR synthesis filters. Although the sizes of W and Q are different from the QMF system, the FIR $\mathcal{H}_\infty$ model-matching method can be applied directly without any modifications. The filters are chosen as in [26]:

$$H_0(s) = \frac{0.145s^2 + 0.0677}{s^3 + 0.3776s^2 + 0.2893s + 0.0677},$$

$$H_1(s) = \frac{0.0066s^5 + 0.0507s^3 + 0.0016s}{s^6 + 0.3754s^5 + 1.7368s^4 + 0.4121s^3 + 0.851s^2 + 0.0901s + 0.1176},$$

and,

$$H(s) = \frac{0.0983s^4 + 0.5012s^2 + 0.4572}{s^6 + 1.9743s^5 + 3.7966s^4 + 3.6605s^3 + 3.3018s^2 + 1.4472s + 0.5755}.$$

In [26] the optimal IIR cost function was determined to be 0.0631, the order of the IIR synthesis filters being 89. However, using the FIR $\mathcal{H}_\infty$ model-matching method, the optimized cost function for 24-tap synthesis filters, shown in Table 4.3 is also 0.0631. Thus, there appears to be no significant degradation in terms of the $\mathcal{H}_\infty$

n	$f_0(n)$	$f_1(n)$
0	-0.01850153	0.03682190
1	0.00521060	0.00667183
2	-0.00544921	0.07949379
3	-0.02265870	-0.04119823
4	0.00710092	0.08052021
5	-0.00417989	0.03333513
6	-0.01202309	0.07882246
7	-0.03432660	-0.18481640
8	0.01192795	0.12840674
9	0.15588569	0.19804506
10	-0.01414087	0.17505728
11	-0.28124527	-0.95462447
12	0.02659146	1.06059673
13	0.87871023	-0.34791308
14	1.16366524	-0.32963357
15	0.51524714	0.19251424
16	-0.12388192	-0.07556957
17	-0.24921665	0.06360486
28	-0.03747594	-0.09789024
19	0.07644620	0.11514995
20	0.00548002	-0.09877461
21	-0.02970953	-0.08034145
22	-0.02688360	0.11413632
23	-0.01840056	-0.02366817

Table 4.3 Coefficients of 24-tap Synthesis Filters for A/D Example

norm. As with the traditional QMF setup, the FIR $\mathcal{H}_\infty$ model-matching method not

only reduces filter order, but also allows for additional constraints like linear-phase or cosine-modulated synthesis filters.

4.3 Transmultiplexers

Transmultiplexers are used to convert time-division multiplexed (TDM) signals to frequency-division multiplexed (FDM) signals, and vice-versa. This conversion is sometimes necessary in digital telephone networks [7]. Figure 4.7 shows the 2-channel transmultiplexer.

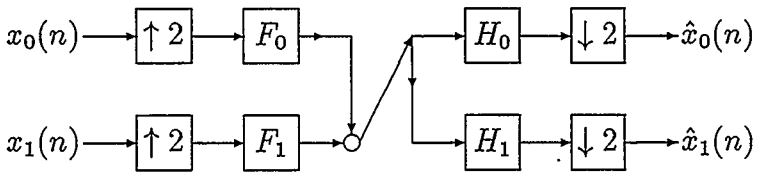

Figure 4.7 2-Channel Transmultiplexer.

The transmultiplexer is similar to the QMF bank, except that the order of the expanders and synthesis filters is interchanged with the decimators and analysis filters [7]. In Figure 4.7 the TDM input signals $x_0(n)$ and $x_1(n)$ are first upsampled. In the frequency-domain, 2-fold upsampling results in 2-fold compression, and consequently, two images. The purpose of the synthesis filters is to retain only one of the images. Ideally the retained copies of $X_0(e^{j2\omega})$ and $X_1(e^{j2\omega})$ do not overlap with each other. Adding the outputs of the synthesis filters results in the FDM format. After transmission, the signal is converted back to the TDM format using the analysis filters and decimation. The analysis filters separate the consecutive regions in the FDM signal,

while the decimators "stretch" the signals into their original format. The design goal is for $\hat{x}_k(n)$ to be scaled and delayed versions of $x_k(n)$. The delays need not be the same for both signals.

If the filters are nonideal, the reconstructed signal $\hat{x}_k(n)$ will generally have a contribution from $x_k(n)$ as well as $x_l(n), l \neq k$. The latter terms are referred to as cross-talk terms. Analogous to aliasing in the QMF bank, cross-talk in the transmultiplexer can be eliminated through careful choice of H_k and F_k. Prior to this technique, proposed by Vetterli [27], the only two options available to eliminate cross-talk were either to design filters with practically non-overlapping frequency responses, or to use guard bands. The problem with using non-overlapping frequency responses is that the required filter order is very high (exceeding 2000) [7]. While guard bands allow larger transition bands in the filters, they do not allow the full channel bandwidth to be used during transmission. The advantage of allowing and eventually eliminating crosstalk through careful choice of filters, is that the resulting filters are more economical than conventional designs.

A duality exists between transmultiplexers and QMF banks. The difference between the two systems is the order of the analysis and synthesis sections. In fact, the problem of designing PR transmultiplexers is the same as designing PR QMF banks. It can be shown [7] that the filters used in a PR transmultiplexer, are simply delayed versions of the filters used in the PR QMF bank.

The problem of designing transmultiplexers is greatly simplified by the LTI nature of the system. As a result, $\hat{X}_k(z)$ is directly related to $X_k(z)$:

$$\hat{\mathbf{X}}(z) = \mathbf{S}(z)\mathbf{X}(z),$$

where,

$$\mathbf{X}(z) = \begin{bmatrix} X_0(z) \\ X_1(x) \end{bmatrix},$$

and similarly,

$$\hat{\mathbf{X}}(z) = \begin{bmatrix} \hat{X}_0(z) \\ \hat{X}_1(x) \end{bmatrix}.$$

The km-th element in $\mathbf{S}$, $S_{km}(z)$, is the 0th polyphase component of $H_k(z)F_m(z)$. Perfect reconstruction implies that $\mathbf{S}$ is a diagonal matrix, so,

$$\begin{aligned} \hat{X}_k(z) &= S_{kk}(z)X_k(z) \\ &= z^{-n_k}X_k(z). \end{aligned}$$

[7] showed that through the use of the noble identities,

$$\mathbf{S}(z) = E(z)\Gamma(z)R(z),$$

where E and R represent the polyphase representations of the analysis and synthesis filters, respectively, and,

$$\begin{aligned} \Gamma(z) &= \begin{bmatrix} 0 & 1 \\ z^{-1}I_{M-1} & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 \\ z^{-1} & 0 \end{bmatrix}, \qquad \text{for } M = 2. \end{aligned}$$

Transforming the system into an error system results in the familiar model-matching problem. The error system in Figure 4.8 is very similar to the original error system used in Figure 3.2. For this reason, the optimization algorithm is unaffected. However, the initial matrices provided to the algorithm are different. The ideal system, $T_d(z)$ is a 2×2 matrix:

$$T_d(z) = \begin{bmatrix} z^{-n_0} & 0 \\ 0 & z^{-n_1} \end{bmatrix}.$$

This is a simpler problem than the QMF and A/D problems, since the original error system is LTI. The optimization only involves matching the transmultiplexer to T_d.

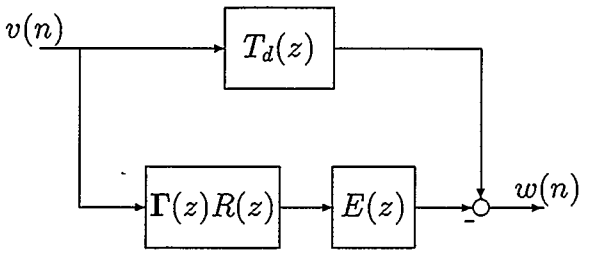

Figure 4.8 Transmultiplexer Error System.

To quantify the reconstruction error, the worst-case $\mathcal{H}_\infty$ norm can once again be used,

$$\begin{aligned} J &= \|T_d - S\|_\infty \\ &= \|T_d - E\Gamma R\|_\infty. \end{aligned}$$

The transmultiplexer differs from the previous two applications in that the synthesis filters must be specified, and the analysis filters are to be designed. The objective is to minimize J over all stable and causal $E(z)$. While this minor change does not affect the ellipsoid algorithm, it does affect the relationship between the optimal coefficients and the resulting filters:

$$\begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix} = E(z^2) \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix}.$$

This leads to the following simple relationship: Defining an FIR $E(z)$ as

$$E(z) = \begin{bmatrix} e_{11}(0) & e_{12}(0) \\ e_{21}(0) & e_{22}(0) \end{bmatrix} + \cdots + \begin{bmatrix} e_{11}(\frac{h_{tap}}{2}-1) & e_{12}(\frac{h_{tap}}{2}-1) \\ e_{21}(\frac{h_{tap}}{2}-1) & e_{22}(\frac{h_{tap}}{2}-1) \end{bmatrix} z^{-(\frac{h_{tap}}{2}-1)}, \quad (4.1)$$

where h_{tap} is an even integer. Then the analysis filters are given by:

$$H_0(z) = e_{11}(0) + e_{12}(0)z^{-1} + e_{11}(1)z^{-2} + e_{12}(1)z^{-3} + \cdots + e_{12}(\frac{h_{tap}}{2}-1)z^{-(h_{tap}-1)} \quad (4.2)$$

and

$$H_1(z) = e_{21}(0) + e_{22}(0)z^{-1} + e_{21}(1)z^{-2} + e_{22}(1)z^{-3} + \cdots + e_{22}(\frac{h_{tap}}{2}-1)z^{-(h_{tap}-1)}. \quad (4.3)$$

Thus, the only modifications are in the setup and in the derivation of the analysis filters from the optimized coefficients. As with the previous two applications, adding constraints to the filters and system is once again straightforward.

As an example, Johnston's 8-tap synthesis filters, as described in [19], are used. The delays for both reconstructed signals $\hat{x}_0(n)$ and $\hat{x}_1(n)$ are chosen to be 5 samples. For the transmultiplexer the analysis filter coefficients are optimized. Table 4.4 shows the coefficients of the four filters. Figure 4.9 shows the magnitude responses

n	$h_0(n)$	$h_1(n)$	$f_0(n)$	$f_1(n)$
0	-0.13569486	0.13645406	0.00938715	0.00938715
1	0.00035990	-0.00042239	0.07065183	-0.07065183
2	0.00035375	0.00145444	0.06942827	0.06942827
3	0.97647595	0.97357027	-0.48998080	0.48998080
4	-0.97286858	0.97588113	0.48998080	0.48998080
5	0.00108502	-0.00054262	-0.06942827	0.06942827
6	0.00030450	-0.00002523	-0.07065183	-0.07065183
7	0.13725006	0.13591819	-0.00938715	0.00938715

Table 4.4 Coefficients of 8-tap Filters for Transmultiplexer

of the four filters shown in Table 4.4. For this combination of filters, the optimized performance measure given by the $\mathcal{H}_\infty$ norm of the error system is 0.161. To verify the performance of the system a ramp input is applied at both inputs. The resulting reconstruction errors for $\hat{x}_0(n)$ and $\hat{x}_1(n)$ are shown in Figure 4.10. Figure 4.10 confirms the reconstructed signals $\hat{x}_0(n)$ and $\hat{x}_1(n)$ resemble the input ramps for $x_0(n)$ and $x_1(n)$. Although the error is higher than in the previous examples, and the filters used may not be practical, the ability to design and determine performance

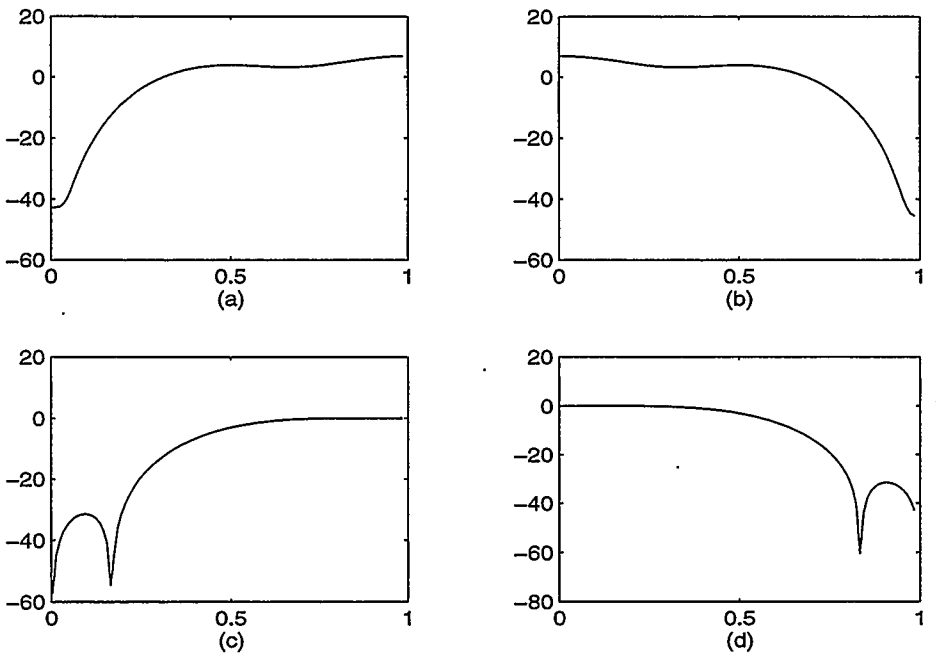

Figure 4.9 Transmultiplexer: Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1.

limitations of transmultiplexers is clearly illustrated.

4.4 Filter Length Tradeoffs

One of the advantages of the FIR $\mathcal{H}_\infty$ model-matching method is the ability to study tradeoffs. Synthesis filter length is one important variable which can be studied. The cost associated with using FIR synthesis filters instead of IIR synthesis filters is easily observed. Often IIR filters may be approximated by truncating the length of their infinite-length impulse responses. Truncation relies on the assumption that the coefficients of the impulse response are sufficiently small beyond a certain point so

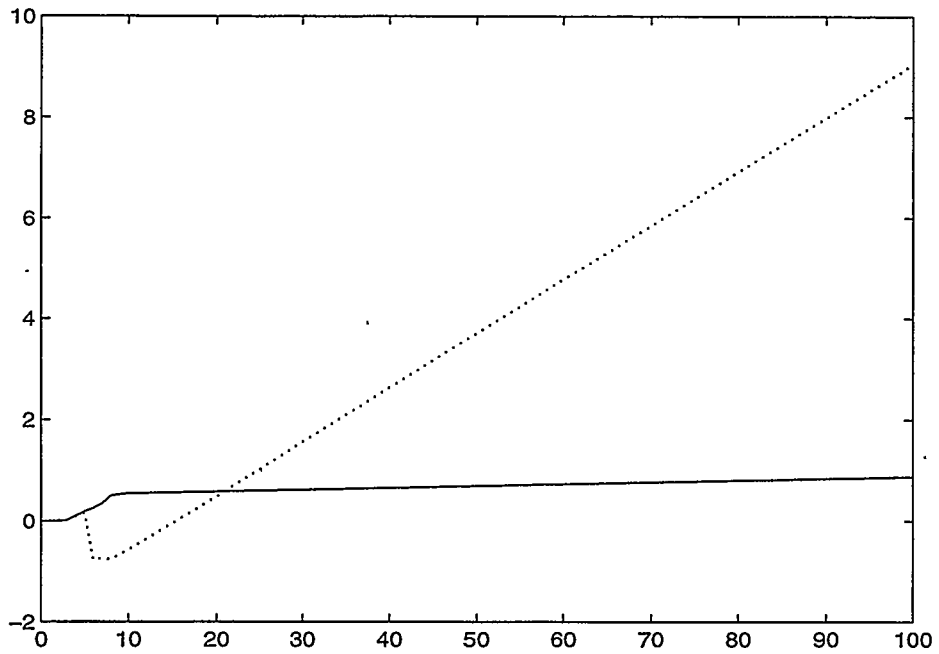

Figure 4.10 Reconstruction Errors for $\hat{x}_0(n)$ (solid) and $\hat{x}_1(n)$ (dotted) for Input Ramps versus n.

that these coefficients may be neglected.

4.4.1 QMF Banks

In [16] the synthesis filters are IIR. Although the order of the synthesis filters is quite high, the impulse responses of these filters can be truncated while still maintaining a comparable cost function, J_{trunc}. In fact, the FIR $\mathcal{H}_\infty$ model-matching method cannot substantially improve upon truncation, provided that the truncation is reasonable. However, the FIR $\mathcal{H}_\infty$ model-matching method will further reduce the cost function, in the case where truncation degrades the cost function significantly.

Figure 4.11 shows the improvement offered by optimization, as opposed to truncation for the example in Section 4.1. In this case $J_{opt} = 0.03$ for the IIR filter.

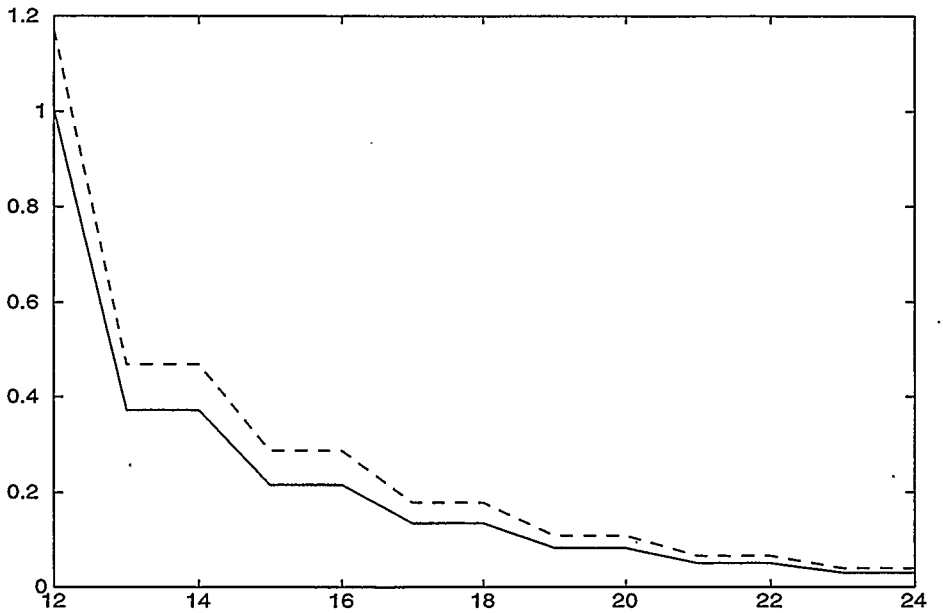

Figure 4.11 J_{trunc} (dashed) and J_{opt} (solid) versus n_{tap} for Simple QMF Example.

Clearly, the FIR $\mathcal{H}_\infty$ model-matching method does improve the cost function obtained by truncation to some extent. However, when truncation results in an impractical cost function ($J_{opt} \geq 1$), the FIR $\mathcal{H}_\infty$ model-matching method also returns an impractical solution ($J_{opt} = 1$). This means that for the chosen combination of analysis filters and delay, any reconstruction of the input signal is impossible, since in the absence of any synthesis filters ($F_0 = 0$ and $F_1 = 0$), $J_{opt} = 1$.

Truncation relies on the fact that the impulse responses of the synthesis filters

die out sufficiently fast. The FIR $\mathcal{H}_\infty$ model-matching method supports the intuitive notion that truncation is a viable option. Furthermore, it quantifies this idea since the ellipsoid algorithm yields a cost function to a specified accuracy, ε, as explained in Section 3.2.1. The cost functions in Figure 4.11 are guaranteed to be within 0.002 of the minimum cost function achievable. Hence, the FIR $\mathcal{H}_\infty$ model-matching method not only confirms the validity of a certain choice of taps, but also confirms that when truncation yields an impractical solution, no practical solution exists for that number of taps.

A second observation regarding Figure 4.11 is the staircase appearance of the cost function, J_{opt}, with respect to the number of taps. The reason for this becomes apparent upon inspection of the impulse responses of the synthesis filters shown in Table 4.1. For that particular example, every other coefficient of the impulse response is small compared to its neighbouring coefficients. Consequently, the cost function is virtually unaffected when increasing the number of taps from n_{tap} to $n_{tap}+1$, for odd n_{tap}.

4.4.2 Fast A/D Converters

A similar comparison can be made for the fast A/D setup. A plot demonstrating the improvements offered by optimization over truncation is shown in Figure 4.12. As expected, J_{opt} is a non-increasing function of the number of taps, and approaches the performance limit established by the IIR filters, as the number of taps increases.

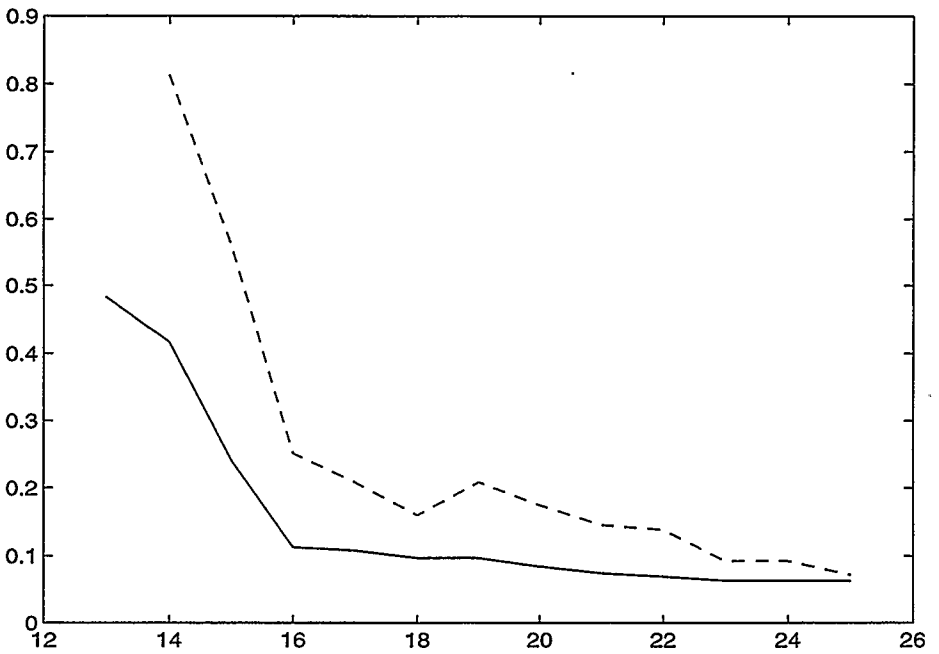

Figure 4.12 J_{trunc} (dashed) and J_{opt} (solid) versus n_{tap} for Fast A/D Converter.

4.4.3 Overall System Delay

Another tradeoff easily examined using the FIR $\mathcal{H}_\infty$ model-matching method is the effect of filter length on the overall system delay. In [16] it was shown that the cost function, J_{opt} is a non-increasing function of the delay. For a given set of analysis filters, one can obtain arbitrarily good reconstruction provided that a sufficiently large time delay can be tolerated. However, this result holds only for IIR synthesis filters. As shown in Figure 4.13, for FIR filters there exists some optimal delay for a given number of taps.

H_0 was designed using the Hamming window approach, with cutoff frequency $\pi/2$ and order 9. For simplicity $H_1(z) = H_0(-z)$. Table 4.5 shows the coefficients of the

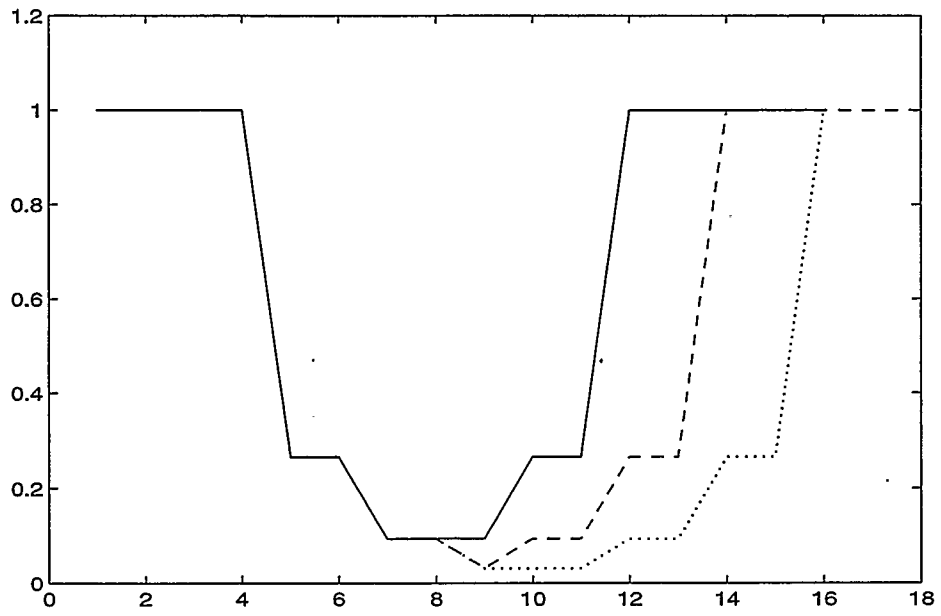

Figure 4.13 J_{opt} versus Delay for 8-tap (solid), 10-tap (dashed), and 12-tap (dotted) Synthesis Filters.

synthesis filters for the 8-tap synthesis filters at delays of 5 and 10 samples, with $J_{opt} \approx 0.265$. The symmetry in Figure 4.13 can be attributed to the linear-phase analysis filters and the fact that the analysis filters are defined to be the mirror image of each other. The optimization introduces some numerical errors which prevent establishing a relationship between the coefficients of the synthesis filters. However, there appears to be a symmetrical and anti-symmetrical relation between the coefficients of synthesis filters for the 5 and 11 sample delay systems, respectively. This is confirmed in Section 6.2.3, where the $\mathcal{H}_2$ norm is used as a performance measure. A more thorough explanation of the symmetry in Figure 4.13 is given in Section 6.2.3.

Delay = 5		Delay = 11	
$f_0(n)$	$f_1(n)$	$f_0(n)$	$f_1(n)$
1.15068079	-1.15468188	0.04212865	-0.04778514
1.18304658	1.19546258	0.00233335	-0.00409643
0.11174422	-0.11284383	-0.00418642	0.00219647
-0.24140364	-0.23457580	0.00450442	-0.00331415
0.00019052	-0.00096658	-0.22041207	0.22256163
0.05549798	0.05666283	0.11184196	0.11221503
-0.00577406	-0.00297160	1.10785217	-1.11218112
-0.04780754	-0.04689106	1.15231486	1.10785217

Table 4.5 8-tap Synthesis Filter Coefficients.

CHAPTER 5

ADDING CONSTRAINTS

One of the main advantages of the proposed FIR $\mathcal{H}_\infty$ model-matching method is the ease with which constraints on the synthesis filters, and on the overall system are handled. In this chapter four constraints are examined:

- Linear-Phase Synthesis Filters
- Cosine-Modulated Synthesis Filters
- Low-Delay Systems
- Alias-Free Systems.

The first two specifications relate to the synthesis filters directly, and the last two specifications apply to the overall system. Although any combination of the above constraints is possible, for the purposes of comparison, each constraint is handled individually.

Although constraints may be added to the ellipsoid algorithm, a simpler approach is to reformulate the problem into a new unconstrained optimization problem. In general, adding constraints results in faster convergence since the number of independent variables is reduced. This idea is shown in Table 5.1, for a system with h_{tap}-tap analysis filters and n_{tap}-tap synthesis filters.

Type	Number of Variables
Simple	$2 \times n_{tap}$
Linear-Phase	n_{tap}
Cosine-Modulated	n_{tap}
Alias-Free	$n_{tap} - h_{tap} + 1$

Table 5.1 Number of Variables to be Optimized for Various Constraints.

5.1 Linear-Phase Synthesis Filters

The setup for linear-phase synthesis filters is quite similar to the general setup. The only difference is, that for linear-phase synthesis filters, there is a relationship between the coefficients of the respective synthesis filters. The linear-phase requirement is implemented using the relationships shown in Table 5.2 [7].

Type	Symmetry	Parity of n_{tap}	Special Features
1	$h(n) = h(n_{tap} - 1 - n)$	n_{tap} odd	
2	$h(n) = h(n_{tap} - 1 - n)$	n_{tap} even	Zero at $\omega = \pi$
3	$h(n) = -h(n_{tap} - 1 - n)$	n_{tap} odd	Zeros at $\omega = 0$ and π
4	$h(n) = -h(n_{tap} - 1 - n)$	n_{tap} even	Zero at $\omega = \pi$

Table 5.2 Four Types of Real Coefficient Linear-Phase FIR Filters

With the linear-phase FIR filters in Table 5.2, half of the coefficients of a filter are defined in terms of the other coefficients. This reduces the number of variables to be optimized over by a factor of two. Constraining the synthesis filters to be linear-phase is therefore handled not by using a constrained ellipsoid algorithm, but by redefining R. For example, if $n_{tap} = 6$, $f_0(n) = f_0(5 - n)$ and $f_1(n) = -f_1(5 - n)$, then R is

defined as,

$$R(z) = \begin{bmatrix} r_{11}(0) & r_{12}(0) \\ r_{21}(0) & r_{22}(0) \end{bmatrix} + \begin{bmatrix} r_{11}(1) & r_{12}(1) \\ r_{11}(1) & -r_{12}(1) \end{bmatrix} z^{-1} + \begin{bmatrix} r_{21}(0) & -r_{22}(0) \\ r_{11}(0) & -r_{12}(0) \end{bmatrix} z^{-2},$$

which after gathering like terms becomes,

$$\begin{aligned} R(z) &= r_{11}(0) \begin{bmatrix} 1 & 0 \\ z^{-2} & 0 \end{bmatrix} + r_{12}(0) \begin{bmatrix} 0 & 1 \\ 0 & -z^{-2} \end{bmatrix} + r_{21}(0) \begin{bmatrix} z^{-2} & 0 \\ 1 & 0 \end{bmatrix} \\ &+ r_{22}(0) \begin{bmatrix} 0 & -z^{-2} \\ 0 & 1 \end{bmatrix} + r_{11}(1) \begin{bmatrix} z^{-1} & 0 \\ z^{-1} & 0 \end{bmatrix} + r_{12}(1) \begin{bmatrix} 0 & z^{-1} \\ 0 & -z^{-1} \end{bmatrix}. \end{aligned}$$

Clearly, R now has 6 coefficients to be optimized, which is half the number of coefficients in the same setup without a linear-phase constraint on the synthesis filters.

The subgradient calculation for the $\mathcal{H}_\infty$ norm is the same as in Section 3.2.2, except that in this case the transfer matrices of (3.6), $G_1, G_2, \ldots, G_6$, become

$$\begin{aligned} G_1 &= \begin{bmatrix} 1 & 0 \\ \left(\frac{1+j\omega_c}{1-j\omega_c}\right)^{-2} & 0 \end{bmatrix} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right) \\ G_2 &= \begin{bmatrix} 0 & 1 \\ 0 & -\left(\frac{1+j\omega_c}{1-j\omega_c}\right)^{-2} \end{bmatrix} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right) \\ &\vdots \\ G_6 &= \begin{bmatrix} 0 & \left(\frac{1+j\omega_c}{1-j\omega_c}\right)^{-1} \\ 0 & -\left(\frac{1+j\omega_c}{1-j\omega_c}\right)^{-1} \end{bmatrix} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right). \end{aligned}$$

The linear-phase requirement is fairly restrictive. Consequently, practical ($J_{opt} < 1$) filter banks are difficult to find for for arbitrary analysis filters. Finding practical solutions is an iterative procedure involving both the delay and the number of taps in the filters. For the analysis filters used in Section 4.1, a linear-phase system was found for 24-tap synthesis filters. The coefficients of the optimized filters are shown in Table 5.3. Since F_0 and F_1 are linear-phase, with symmetric and antisymmetric impulse responses, respectively, only the first half of the coefficients are given.

n	$f_0(n)$	$f_1(n)$
0	0.00011976	0.00025757
1	-0.05238635	-0.05228220
2	-0.00163162	0.00202955
3	0.05989666	0.05969398
4	0.00235090	-0.00239896
5	-0.09838709	-0.09870367
6	0.00364273	-0.00394522
7	0.15513133	0.15535685
8	-0.03737916	0.03725379
9	-0.25837940	-0.25801941
10	0.20308650	-0.20297840
11	0.99603549	0.99600354

Table 5.3 Coefficients of 24-tap Linear-Phase Synthesis Filters

Furthermore, the magnitude responses of the analysis and synthesis filters for this example are shown in Figure 5.1.

Table 5.4 compares the IIR [16] and unconstrained FIR $\mathcal{H}_\infty$ model-matching results with the resulting linear-phase synthesis filters, the FIR synthesis filters all having the same length ($n_{tap} = 24$).

Method	J_{opt}
IIR [16]	0.03
IIR [16] (truncated)	0.03935
FIR $\mathcal{H}_\infty$ (unconstrained)	0.030248
FIR $\mathcal{H}_\infty$ (linear-phase)	0.030193

Table 5.4 Cost of Linear-Phase Synthesis Filters

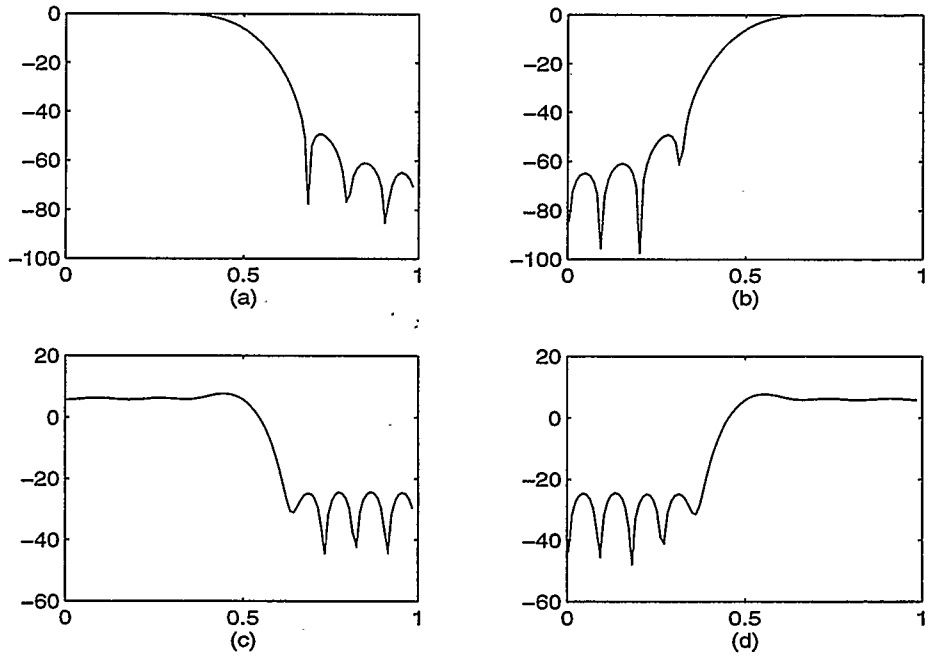

Figure 5.1 Linear-Phase Constraint: Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1.

The unconstrained FIR $\mathcal{H}_\infty$ model-matching method resulted in "nearly" linear-phase filters, while the constrained FIR $\mathcal{H}_\infty$ model-matching method resulted in exactly linear-phase filters. This suggests that the optimal solution is indeed linear-phase, and the linear-phase constraint serves to eliminate any numerical errors which prevent the unconstrained method from achieving exactly linear-phase synthesis filters.

To confirm the validity of this choice of synthesis filters, the performance measures from Chapter 3 are used. Figure 5.2 shows the distortion plots for this setup. As expected, $T(z)$ is nearly unity with linear phase, while $A(z)$ is small. The input ramp and subsequent error are plotted in Figure 5.3. Table 5.5 gives a summary

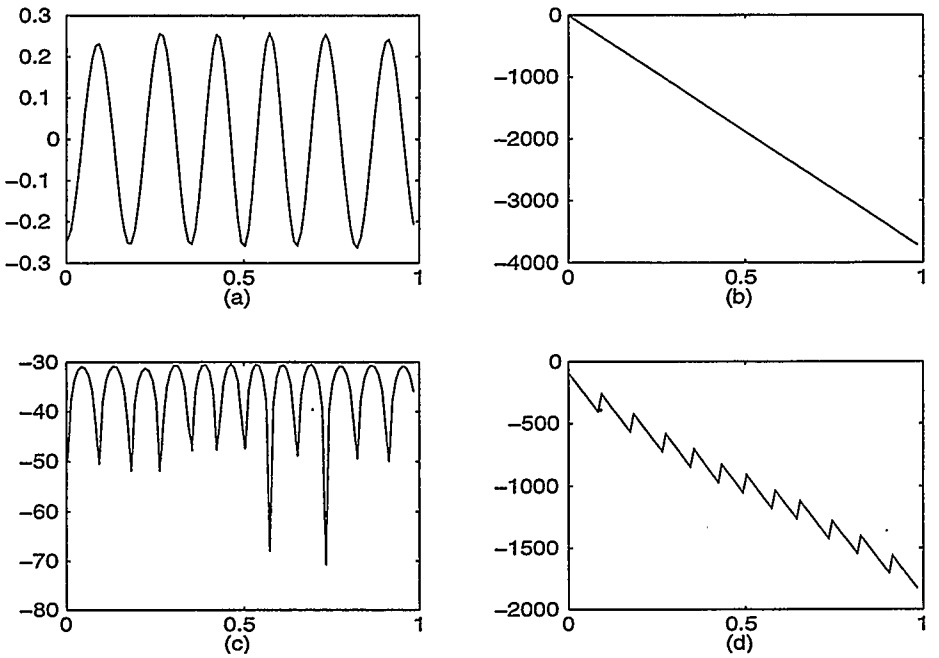

Figure 5.2 Linear-Phase Constraint: Magnitude (dB) Distortions for $T(z)$ and $A(z)$ ((a) and (c)) versus ω/π. Phase (°) Distortions for $T(z)$ and $A(z)$ ((b) and (d)) versus ω/π.

of the various performance measures, and confirms that the $\mathcal{H}_\infty$ norm is indeed a worst-case norm.

$\mathcal{H}_\infty$ Norm	SNR_r	$\|\mathbf{AS}-\mathbf{B}\|_F$	Magnitude Error	Phase Error	Aliasing
0.0302	0.020226	0.0017119	0.02996	9.1×10^{-13} °	-30.36 dB

Table 5.5 Performance Measures for the Linear-Phase Example.

Another constraint which may be handled similarly to the linear-phase constraint

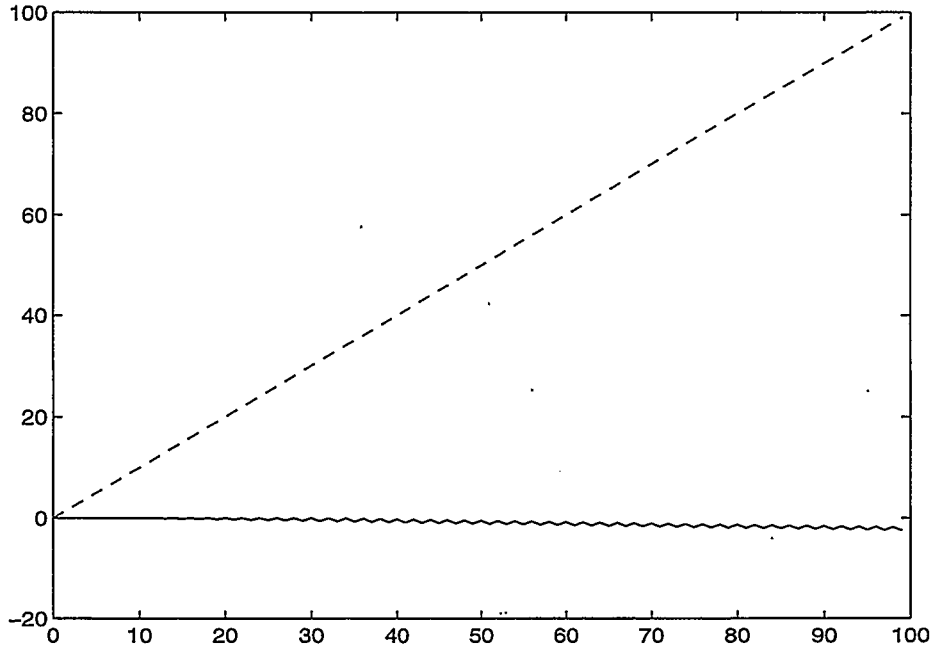

Figure 5.3 Linear-Phase Constraint: Input Ramp (dashed) and Reconstruction Error (solid) versus n.

is the cosine-modulated constraint.

5.2 Cosine-Modulated Synthesis Filters

Filters derived from a single prototype filter by cosine-modulation are called cosine-modulated filters. Two noteworthy points regarding these filters are [7]:

- The implementation complexity of the filter banks is the cost of the prototype filter plus the modulation cost.
- The optimization involves only the coefficients of the prototype filter.

The additional improvements offered by the FIR $\mathcal{H}_\infty$ model-matching method still hold. In particular, low-delay systems, linear-phase synthesis prototype filters, and/or any of the other constraints are still possible. Unlike with many other methods, [8], [11], [21], [22], and [23], it may be possible to have synthesis filters with lower order than the analysis filters, depending on the chosen initial setup, thereby reducing the overall complexity.

The most general format of a cosine-modulated filter is [20],

$$g_k(n) = p(n)\cos\left(\frac{\pi}{2}(n-\theta)(k+\phi)+\gamma\right), \quad 0 \le k \le 1, \tag{5.1}$$

where θ, ϕ, and γ are predefined constants, and $p(n)$ is the baseband prototype filter. Most methods define both the analysis and synthesis filters in terms of the same prototype filter. In [20] this is not the case; the analysis filters are given by,

$$h_k(n) = h_b(n)\cos\left(\frac{\pi(n-n_a)}{2}(k+\frac{1}{2})\right), \quad 0 \le k \le 1, \tag{5.2}$$

and similarly for the synthesis filters,

$$f_k(n) = f_b(n)\cos\left(\frac{\pi(n-n_s)}{2}(k+\frac{1}{2})\right), \quad 0 \le k \le 1, \tag{5.3}$$

where n_a and n_s are the analysis and synthesis shift factors, which may or may not be integers [20]. The technique in [20] is an extension of the approach presented in [15] with modulation constraints placed on the analysis and synthesis filters.

Since one of the advantages of the FIR $\mathcal{H}_\infty$ model-matching method is that the design of the analysis and synthesis filters is decoupled, (5.2) and (5.3) are used as the basis for the filters.

The implication for the ellipsoid algorithm is that instead of optimizing the coefficients of the synthesis filters, only the coefficients of $f_b(n)$ need to be optimized. The

synthesis filters can subsequently be derived from $f_b(n)$ according to (5.3). The number of coefficients to optimize is halved since both of the synthesis filters are derived from a single prototype filter. Each coefficient of $f_0(n)$ and $f_1(n)$ is then determined by a factor dependent only on the position, n, of the coefficient. The subgradient calculation for the $\mathcal{H}_\infty$ norm is once again the same as in Section 3.2.2. However, the transfer matrices of (3.6), $G_1, G_2, \ldots, G_{n_{tap}}$, for an even value of n_{tap} become

$$
\begin{aligned}
G_1 &= \begin{bmatrix} 0 & 0 \\ C_0(0) & C_1(0) \end{bmatrix} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right) \\
G_2 &= \begin{bmatrix} C_0(1) & C_1(1) \\ 0 & 0 \end{bmatrix} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right) \\
G_3 &= \begin{bmatrix} 0 & 0 \\ C_0(2) & C_1(2) \end{bmatrix} \left(\tfrac{1+j\omega_c}{1-j\omega_c}\right)^{-1} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right) \\
G_4 &= \begin{bmatrix} C_0(3) & C_1(3) \\ 0 & 0 \end{bmatrix} \left(\tfrac{1+j\omega_c}{1-j\omega_c}\right)^{-1} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right) \\
&\vdots \\
G_{n_{tap}} &= \begin{bmatrix} C_0(n_{tap}-1) & C_1(n_{tap}-1) \\ 0 & 0 \end{bmatrix} \left(\tfrac{1+j\omega_c}{1-j\omega_c}\right)^{-(\frac{n_{tap}}{2}-1)} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right),
\end{aligned}
$$

where,

$$C_k(n) = \cos\left(\frac{\pi(n-n_s)}{2}(k+\frac{1}{2})\right).$$

A similar set of transfer matrices can be derived for odd n_{tap}, except in this case,

$$G_{n_{tap}} = \begin{bmatrix} 0 & 0 \\ C_0(n_{tap}-1) & C_1(n_{tap}-1) \end{bmatrix} \left(\tfrac{1+j\omega_c}{1-j\omega_c}\right)^{-(\frac{n_{tap}+1}{2}-1)} E\left(\tfrac{1+j\omega_c}{1-j\omega_c}\right).$$

As with the linear-phase constraint, finding a suitable setup is an iterative procedure. For the following example the analysis filters are defined in terms of a baseband lowpass filter with cutoff frequency $\pi/4$, as in (5.2), with $n_a = -1.5$. The length of all four filters is chosen to be 4 taps. The system delay m is 3, and $n_s = 4.5$. The

coefficients of the analysis prototype and the optimized synthesis prototype filters are given in Table 5.6. The magnitude plots for the analysis and synthesis filters are shown in Figure 5.4

n	$h_b(n)$	$f_b(n)$
0	0.03857900	0.18185295
1	0.46142099	2.16746129
2	0.46142099	2.16932577
3	0.03857900	0.00439477

Table 5.6 Analysis and Synthesis Prototype Filters.

Although this example may not be of practical interest, it illustrates the ability to design cosine-modulated synthesis filters. The corresponding distortion plots are shown in Figure 5.5. Figure 5.6 shows a plot of the error for a ramp input. In comparison to the input the corresponding error is very small.

Table 5.7 is a summary of the various performance measures.

$\mathcal{H}_\infty$ Norm	SNR_r	$\|\mathbf{AS}-\mathbf{B}\|_F$	Magnitude Error	Phase Error	Aliasing
0.00225	0.0014	5.2E-06	0.0015	0.064°	-56.66 dB

Table 5.7 Performance Measures for the Cosine-Modulated Example.

One of the features of the method in [20] and the FIR $\mathcal{H}_\infty$ model-matching method is that the ability to specify low-delay systems is preserved.

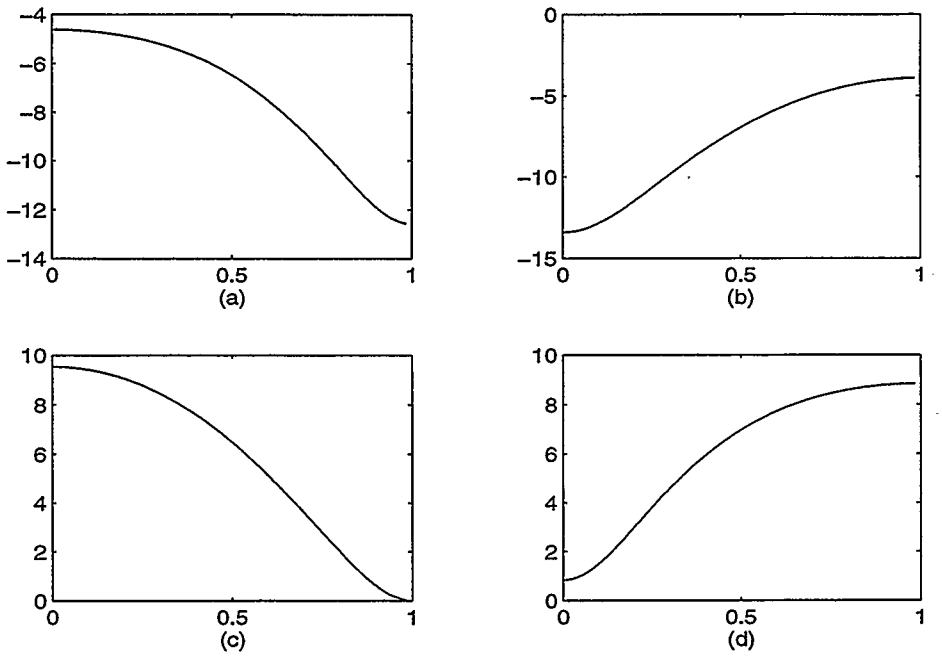

Figure 5.4 Cosine-Modulated Constraint: Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1.

5.3 Low Delay Systems

In [12] a technique for designing low delay analysis-synthesis systems was proposed. A system is said to have low reconstruction delay, m, when

$$m < n_{tap} - 1,$$

where n_{tap} is the length of the filters. Systems with low delay are particularly useful for tree-structured subband coders [20]. The method in [12] is based on a time domain approach, and like the FIR $\mathcal{H}_\infty$ model-matching method, has more flexibility in designing the time delay of the overall system than conventional methods. Specifically, the approach reduces a reconstruction error to zero and minimizes a frequency error

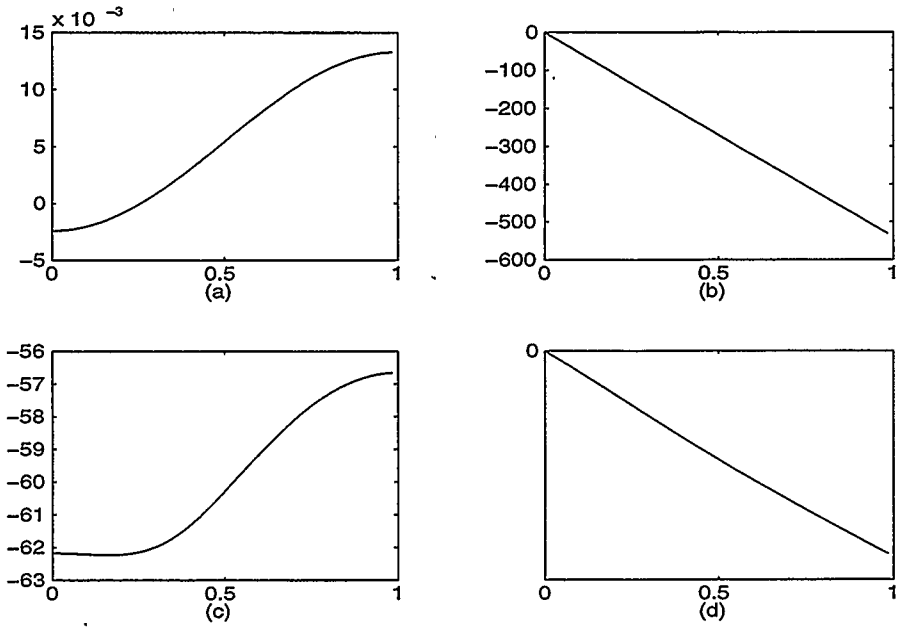

Figure 5.5 Cosine-Modulated Constraint: Magnitude (dB) Distortions for $T(z)$ and $A(z)$ ((a) and (c)) versus ω/π. Phase (°) Distortions for $T(z)$ and $A(z)$ ((b) and (d)) versus ω/π.

term containing the passband and stopband errors of the system filters [12]. A more thorough description of this method was given in Section 2.3.2.

The method in [12] differs from the FIR $\mathcal{H}_\infty$ model-matching method in three ways. The most significant difference lies in the performance specifications. The $\mathcal{H}_\infty$ norm is a worst-case norm over all possible finite-energy input signals, while the Frobenius norm, used by [12], minimizes the error for an impulse input. Another difference between the two methods is that in [12], the analysis filters are restricted to be FIR. Finally, in contrast to the FIR $\mathcal{H}_\infty$ model-matching method which assumes that the analysis filters have already been designed, in [12] the analysis filters are

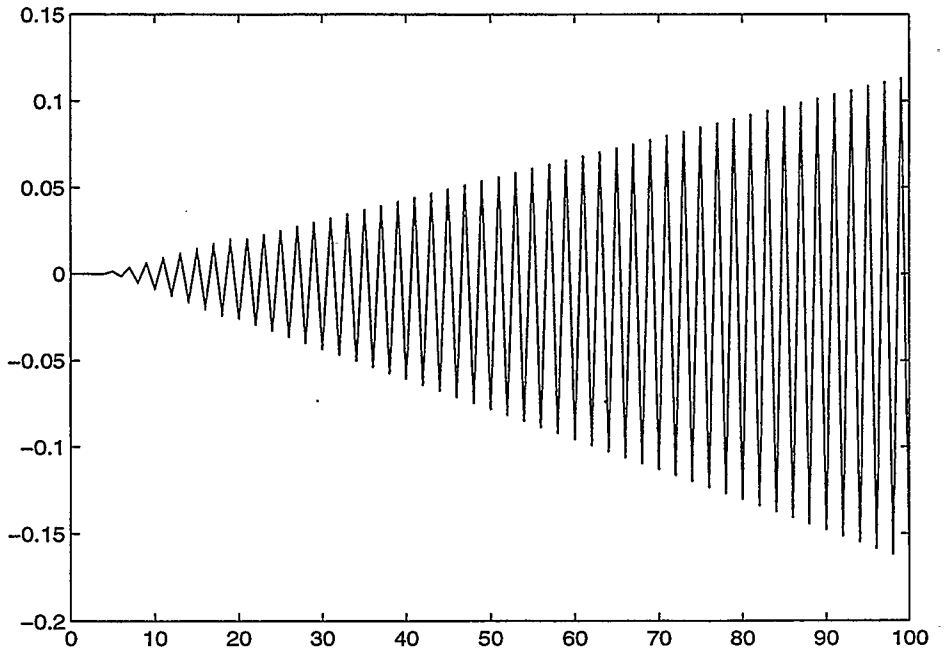

Figure 5.6 Cosine-Modulated Constraint: Reconstruction Error versus n.

variables. Consequently, using the FIR $\mathcal{H}_\infty$ model-matching method, the designer has complete control over the analysis filters. The synthesis filters in [12] are directly related to the analysis filters:

$$\begin{aligned} F_0(z) &= H_1(-z) \\ F_1(z) &= -H_0(-z). \end{aligned}$$

Although this choice of analysis filters guarantees the elimination of aliasing, by relaxing this condition it may be possible to improve the reconstruction properties of the system.

Low-delay systems are easily accommodated by the FIR $\mathcal{H}_\infty$ model-matching method. No changes to the ellipsoid algorithm are required since delay is one of the variables specified by the user. However, finding a system with a useful J_{opt} ($J_{opt} < 1$)

involves some trial and error. To avoid the trial and error process, the analysis filters given in [12] are used throughout this section. The coefficients for the analysis filters are shown in Table 5.8. The delay chosen for this particular setup is one sample. The

n	$h_0(n)$	$h_1(n)$
0	3.89565E-01	4.15186E-01
1	6.32632E-01	-6.09244E-01
2	1.40179E-01	1.06619E-01
3	-2.12770E-01	1.65608E-01
4	4.41944E-03	-4.31897E-03
5	8.06863E-02	-3.19333E-02
6	-2.10077E-02	7.99625E-03
7	-9.78011E-03	3.76170E-03

Table 5.8 Coefficients of 8-tap Analysis Filters

corresponding coefficients of the optimized synthesis filters are shown in Tables 5.9. Figure 5.7 shows the magnitude responses of the analysis and synthesis filters.

n	$f_0(n)$	$f_1(n)$
0	0.83037185	-0.77912986
1	1.21848778	1.26526377
2	0.20413244	-0.28981432
3	-0.34457732	-0.43941428
4	-0.00840111	-0.00798339
5	0.07127725	0.16996261
6	0.01621581	0.04168687
7	-0.01003252	-0.02346107

Table 5.9 Coefficients of 8-tap Analysis Filters for Delay = 1

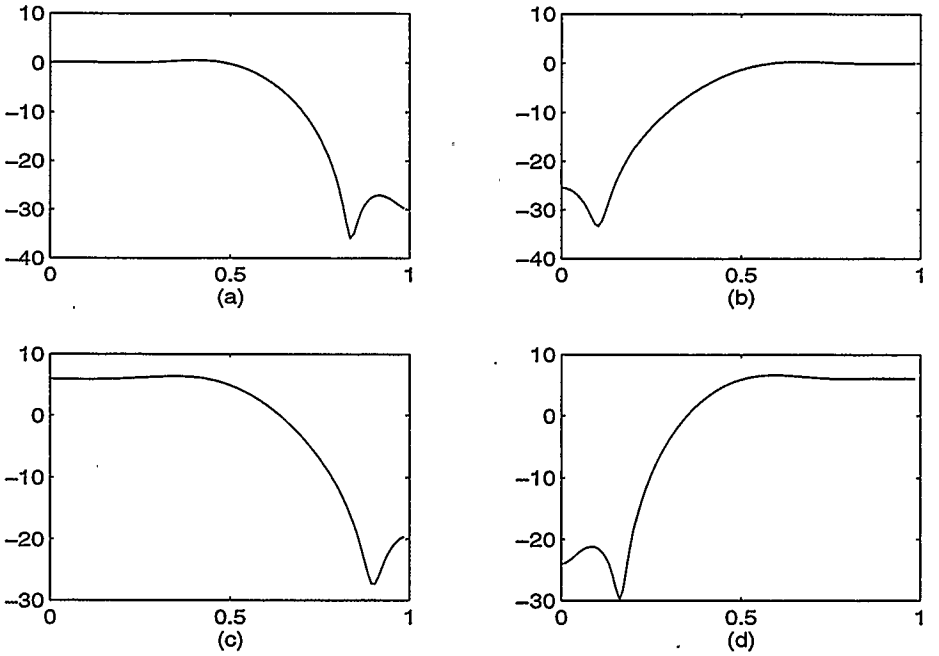

Figure 5.7 Low-Delay System: Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1.

Table 5.10 shows the comparison of the cost functions using the $\mathcal{H}_\infty$ norm for [12] and the FIR $\mathcal{H}_\infty$ model-matching method. In each case the analysis filters from [12] are used, so the difference between the two methods is solely due to the synthesis filters.

For 8-tap synthesis filters, Table 5.10 indicates that according to the $\mathcal{H}_\infty$ norm, the FIR $\mathcal{H}_\infty$ model-matching method yields better results. Furthermore, the table suggests that using fewer taps still results in similar performance in terms of the $\mathcal{H}_\infty$ norm.

Table 5.11 compares the signal-to-reconstruction-noise ratios for the two different

	$\mathcal{H}_\infty$ Norm
Low Delay Method [12]	0.0146
FIR $\mathcal{H}_\infty$ Method (8 taps)	0.0001
FIR $\mathcal{H}_\infty$ Method (7 taps)	0.0155
FIR $\mathcal{H}_\infty$ Method (6 taps)	0.0156
FIR $\mathcal{H}_\infty$ Method (5 taps)	0.1144

Table 5.10 Comparison of $\mathcal{H}_\infty$ Norm

methods. Table 5.11 suggests that the FIR $\mathcal{H}_\infty$ model-matching method outperforms

	SNR_r
Low Delay Method [12]	41.720 dB
IIR Method	304.928 dB
FIR $\mathcal{H}_\infty$ Method (8 taps)	68.475 dB
FIR $\mathcal{H}_\infty$ Method (7 taps)	44.354 dB
FIR $\mathcal{H}_\infty$ Method (6 taps)	41.967 dB
FIR $\mathcal{H}_\infty$ Method (5 taps)	31.34 dB

Table 5.11 Comparison of SNR_r

the method proposed in [12] with respect to SNR_r. The improvement is of the order of 20 dB for the eight tap filters. Furthermore, Table 5.11 suggests that the number of taps in the synthesis filters can be reduced to six, without adverse effects on the SNR_r.

The other measure of performance, given by the distortion plots, is shown in Figure 5.8 for the 8-tap example. The error due to a ramp input is shown in Figure 5.9.

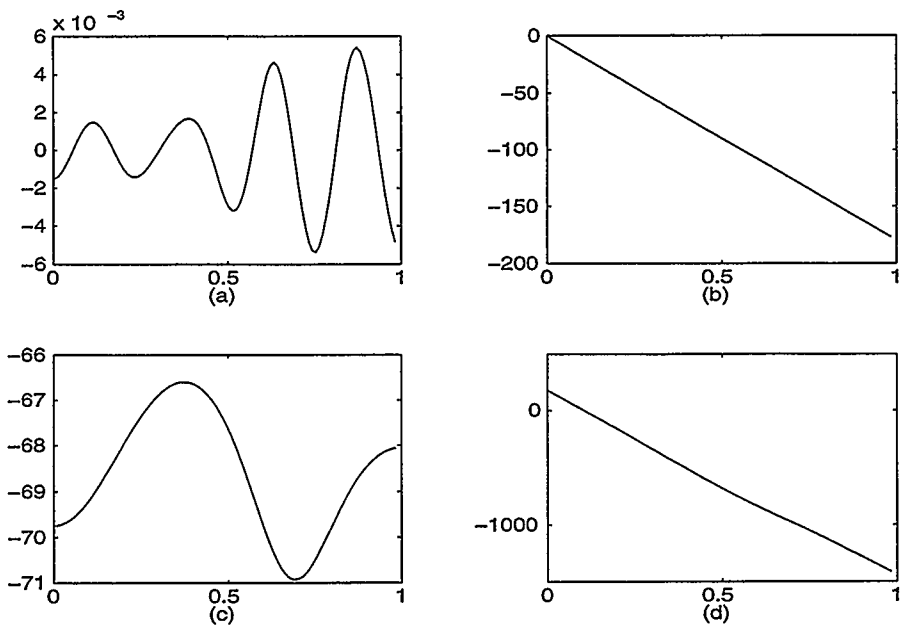

Figure 5.8 Low-Delay System: Magnitude (dB) Distortions for $T(z)$ and $A(z)$ ((a) and (c)) versus ω/π. Phase (°) Distortions for $T(z)$ and $A(z)$ ((b) and (d)) versus ω/π.

Table 5.12 summarizes the various performance measures for the FIR $\mathcal{H}_\infty$ model-matching designs. Clearly the performance degrades as the number of taps decreases, indicating a tradeoff between the number of taps and performance. However, this degradation may be acceptable for some applications.

5.4 Alias-Free Systems

Like the linear-phase and cosine-modulated constraints, the alias-free constraint is handled, not as a constrained optimization problem, but rather by formulating a new unconstrained optimization problem. For the system to be alias-free, the following

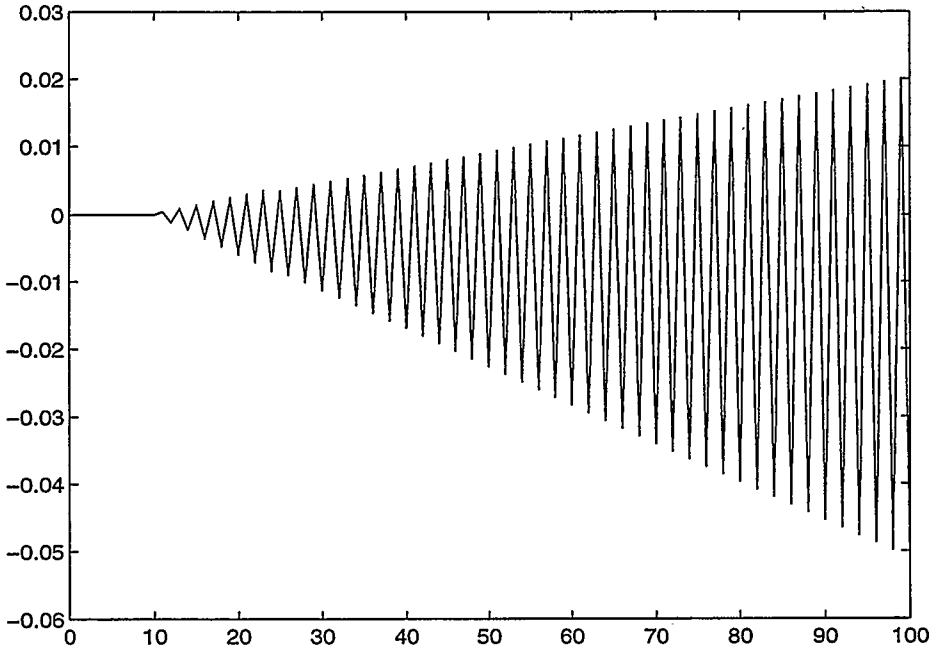

Figure 5.9 Low-Delay System: Reconstruction Error versus n.

Taps	$\mathcal{H}_\infty$ Norm	SNR_r	$\|\mathbf{AS}-\mathbf{B}\|_F$	Magnitude Error	Phase Error	Aliasing
8	0.00083	0.00038	6.53E-07	6.22E-04	0.036°	-66.60 dB
7	0.0155	0.006	2.88E-04	0.0122	0.668°	-38.17 dB
6	0.0156	0.008	0.001	0.0306	1.79°	-35.91 dB
5	0.1144	0.027	0.015	0.0584	3.36°	-20.25 dB

Table 5.12 Performance Measures for Low-Delay Constraint.

condition must hold [7]:

$$H_0(-z)F_0(z) + H_1(-z)F_1(z) = 0. \tag{5.4}$$

The following method will only yield FIR synthesis filters for FIR analysis filters. However, IIR analysis filters may be approximated by truncating their impulse re-

sponses appropriately. Solving (5.4) for $F_1(z)$ gives

$$F_1(z) = -\frac{H_0(-z)F_0(z)}{H_1(-z)},$$

which is in general IIR, unless $F_0(z)$ has the same zeros as $H_1(-z)$. As a result,

$$F_0(z) = H_1(-z)P(z),$$

and consequently,

$$F_1(z) = -H_0(-z)P(z).$$

Having defined the synthesis filters in terms of the analysis filters, the optimization involves optimizing over an FIR $P(z)$. The number of taps in $p(n)$, p_{tap}, is $n_{tap} - h_{tap} + 1$, where n_{tap} and h_{tap} refer to the number of taps in the synthesis and analysis filters, respectively.

The alias-free constraint has an impact on the transfer matrices in (3.6):

$$G_k = \begin{cases} \begin{bmatrix} H_1^{even}(e^{j\omega_c}) & H_0^{even}(e^{j\omega_c}) \\ H_1^{odd}(e^{j\omega_c}) & H_0^{odd}(e^{j\omega_c}) \end{bmatrix} \left(\frac{1+j\omega_c}{1-j\omega_c}\right)^{-(\frac{k-1}{2})} E\left(\frac{1+j\omega_c}{1-j\omega_c}\right) & \text{if } k = \text{odd} \\ \begin{bmatrix} H_1^{odd}(e^{j\omega_c})\left(\frac{1+j\omega_c}{1-j\omega_c}\right)^{-(\frac{k}{2}-1)} & H_0^{odd}(e^{j\omega_c})\left(\frac{1+j\omega_c}{1-j\omega_c}\right)^{-(\frac{k}{2}-1)} \\ H_1^{even}(e^{j\omega_c})\left(\frac{1+j\omega_c}{1-j\omega_c}\right)^{-\frac{k}{2}} & H_0^{even}(e^{j\omega_c})\left(\frac{1+j\omega_c}{1-j\omega_c}\right)^{-\frac{k}{2}} \end{bmatrix} E\left(\frac{1+j\omega_c}{1-j\omega_c}\right) & \text{if } k = \text{even.} \end{cases}$$

where, G_k, $(k = 1, 2, \ldots, p_{tap})$, and

$$H_k^{even}(e^{j\omega_c}) = \sum_{l=0}^{\frac{h_{tap}}{2}-1} h_k(2l)\left(\frac{1+j\omega_c}{1-j\omega_c}\right)^{-l} \quad 0 \le k \le 1, \tag{5.5}$$

and

$$H_k^{odd}(e^{j\omega_c}) = \sum_{l=0}^{\frac{h_{tap}}{2}-1} h_k(2l+1)\left(\frac{1+j\omega_c}{1-j\omega_c}\right)^{-l} \quad 0 \le k \le 1. \tag{5.6}$$

Equations (5.5) and (5.6) are for even h_{tap}. For odd h_{tap}, H_k^{even} and H_k^{odd} are defined similarly, except the upper limit of the summations in (5.5) and (5.6) changes to $\frac{h_{tap}-1}{2}$.

The analysis filters ($h_{tap} = 20$, delay $= 21$) from Section 4.1 were used to observe the effect of adding the alias-free constraint. Once again, the FIR synthesis filters all have 24 taps. The optimized coefficients of $p(n)$ are given in Table 5.13. The

n	$p(n)$
0	1.00000000
1	0.00290967
2	-0.07541587
3	-0.00145432
4	0.12913010

Table 5.13 Coefficients of $p(n)$ for Alias-Free System

magnitude responses of the analysis and synthesis filters are given in Figure 5.10.

The restrictive nature of the alias-free condition is shown in Table 5.14, where the cost function increases dramatically upon the addition of the alias-free constraint.

Method	J_{opt}
IIR [16]	0.0300
IIR [16] (truncated)	0.0394
FIR $\mathcal{H}_\infty$ (unconstrained)	0.0302
FIR $\mathcal{H}_\infty$ (alias-free)	0.2786

Table 5.14 Effect of Alias-Free Condition

The resulting distortion plots for the alias-free system are shown in Figure 5.11. The price paid for the alias-free condition is clearly illustrated. In comparison to

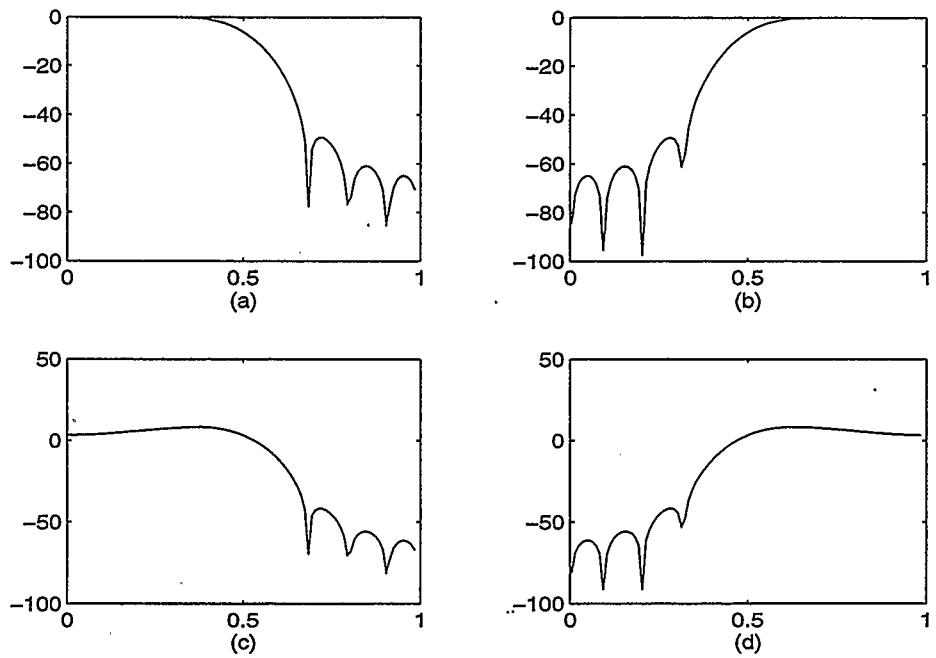

Figure 5.10 Alias-Free Constraint: Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1.

Figure 4.3, $T(z)$ is worse, but the aliasing is now negligible.

Since the alias-free system is LTI, according to (3.7), it can be described by the transfer function $T(z)$. Therefore, the remaining magnitude and phase distortion errors can now be directly related to the $\mathcal{H}_\infty$ norm. Thus, the reconstructed signal $\hat{X}(z)$ can be expressed in terms of input signal $X(z)$:

$$\hat{X}(z) = T(z)X(z),$$

and in terms of the frequency response,

$$\hat{X}(e^{j\omega}) = |T(e^{j\omega})|\angle\phi(\omega)X(e^{j\omega}).$$

Ideally, the transfer function $T(z)$ is a delay. Consequently, the magnitude error at

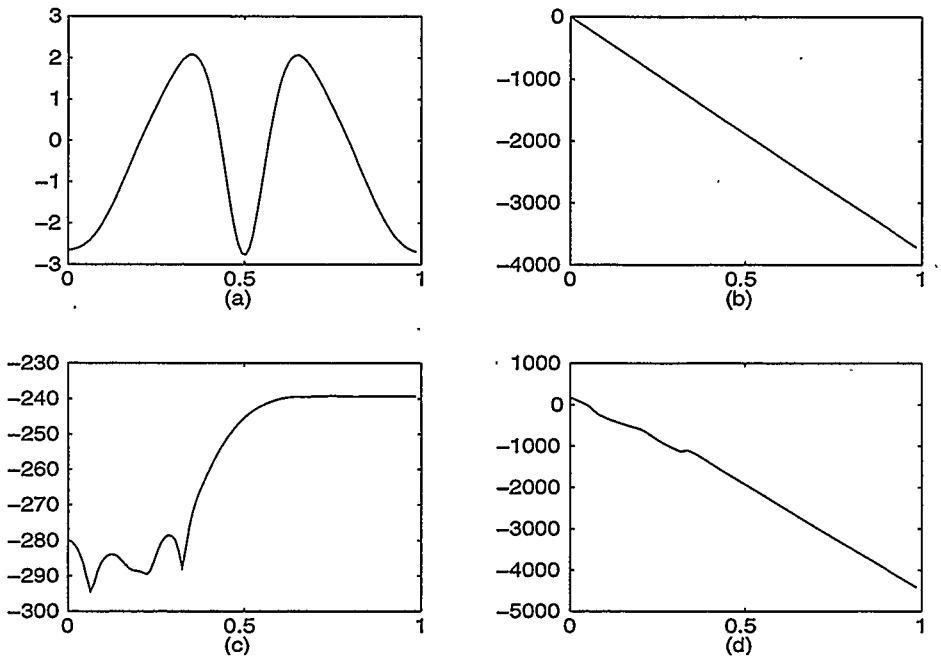

Figure 5.11 Alias-Free Constraint: Magnitude (dB) Distortions for $T(z)$ and $A(z)$ ((a) and (c)) versus ω/π. Phase (°) Distortions for $T(z)$ and $A(z)$ ((b) and (d)) versus ω/π.

frequency ω is defined as,

$$M(\omega) = \Big||T(e^{j\omega})| - 1\Big|.$$

The maximum magnitude error is then

$$M_{max} = \sup_{0\leq\omega\leq 2\pi} \Big||T(e^{j\omega})| - 1\Big|.$$

Similarly for the phase error, define

$$\Phi(\omega) = \phi(\omega) - (-\omega m),$$

and,

$$\Phi_{max} = \sup_{0\leq\omega\leq 2\pi} |\phi(\omega) + \omega m|.$$

For this LTI system the $\mathcal{H}_\infty$ norm performance is,

$$\begin{aligned} J_{opt} &= \|T(z) - z^{-m}\|_\infty \\ &= \max_\omega \left| |T(e^{j\omega})| e^{j\phi(\omega)} - e^{-j\omega m} \right| \\ &= \max_\omega \left| |T(e^{j\omega})| e^{j[\phi(\omega)+\omega m]} - 1 \right| \\ &= \max_\omega \left| |T(e^{j\omega})| e^{j\Phi(\omega)} - 1 \right|. \end{aligned} \tag{5.7}$$

Relating the magnitude and phase errors to the $\mathcal{H}_\infty$ norm is now straightforward. Equation (5.7) implies that at any frequency ω,

$$\left| |T(e^{j\omega})| e^{j\Phi(\omega)} - 1 \right| \leq J_{opt},$$

which means that the complex number $|T(e^{j\omega})| e^{j\Phi(\omega)}$ lies in the disc centered at 1, with radius J_{opt}. Thus,

$$M(\omega) \leq J_{opt},$$

and

$$|\Phi(\omega)| \leq \sin^{-1}(J_{opt}),$$

and consequently,

$$M_{max} \leq J_{opt},$$

and

$$\Phi_{max} \leq \sin^{-1}(J_{opt}).$$

In other words, the $\mathcal{H}_\infty$ norm of the error system provides an upper bound on the maximum magnitude and phase errors for alias-free systems.

As a further measure of performance, the ramp input and subsequent error are plotted in Figure 5.12. Clearly the alias-free condition imposed on the system severely

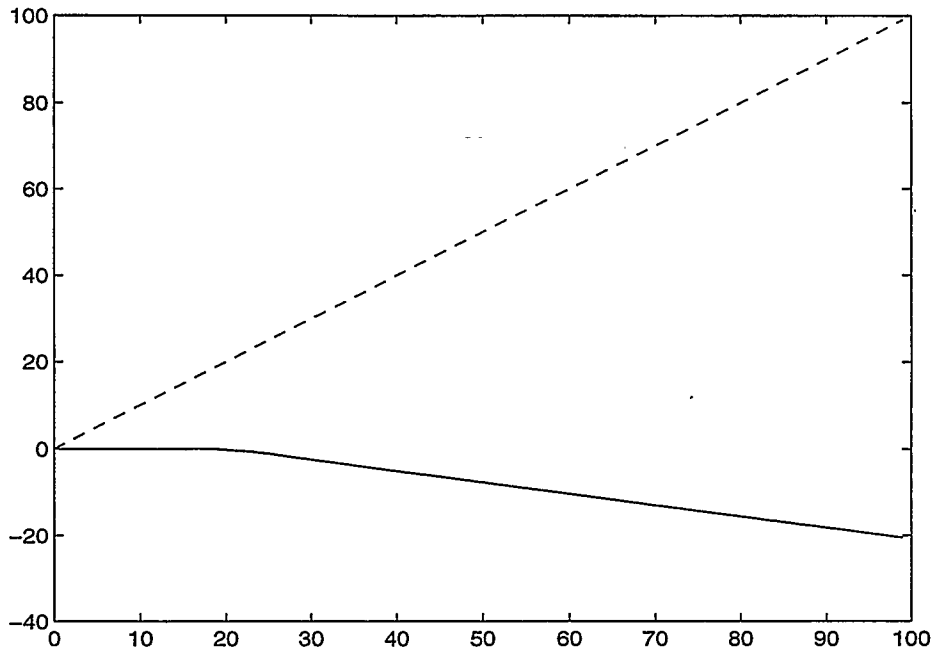

Figure 5.12 Alias-Free Constraint: Input Ramp (dashed) and Reconstruction Error (solid) versus n.

degrades the performance of the system. Table 5.15 shows a summary of the performance measures. Although, according to Figure 5.11 and Table 5.15 there is still some aliasing; however, this can be solely attributed to numerical errors. Comparing

$\mathcal{H}_\infty$ Norm	SNR_r	$\|\mathbf{AS}-\mathbf{B}\|_F$	Magnitude Error	Phase Error	Aliasing
0.27864	0.1845	0.07214	0.2723	1.29°	-239 dB

Table 5.15 Performance Measures for Alias-Free Constraint.

Table 5.15 to Table 4.2 illustrates the cost introduced by adding the alias-free con-

straint. In this case, the degradation in reconstruction outweighs the benefits gained by the alias-free constraint.

The FIR $\mathcal{H}_\infty$ model-matching method is a useful tool for determining this type of tradeoff, since the designer has direct control over all the variables. Thus, the effects of the various constraints are easily isolated.

CHAPTER 6

APPLICATIONS OF THE $\mathcal{H}_2$ NORM

The $\mathcal{H}_\infty$ norm of a system corresponds to a worst-case bound over all inputs. On the other hand, the $\mathcal{H}_2$ norm is a measure of the size of an output signal, given a specific input. By the Parseval theorem, the $\mathcal{H}_2$ norm of a single-input single-output (SISO) LTI system can be interpreted as the ℓ_2 norm of its impulse response [25]:

$$\|G\|_2 = \|g\|_2.$$

A similar interpretation holds for the $\mathcal{H}_2$ norm of a MIMO LTI system. If G is an n_w-input, n_z-output system, then its impulse response $g(n)$ is also n_z by n_w. The $\mathcal{H}_2$ norm is given by [28]

$$\|G\|_2 = \left(\sum_{i=1}^{n_z} \sum_{k=1}^{n_w} \|g_{ik}\|_2^2 \right)^{\frac{1}{2}}. \tag{6.1}$$

6.1 Sensitivity to Quantization Noise

There are three types of quantization effects [7]: Multiplier (or Coefficient) Quantization, Signal Quantization, and Subband Quantization. The latter tends to dominate the total noise [7], and is the focus of this section. Subband quantization can be modeled by adding a noise source, $q_i(n)$, to each channel, as shown in Figure 6.1. As in [7], each $q_i(n)$ is assumed to be wide sense stationary (WSS) and white, and uniformly distributed with zero mean and variance, σ^2. Without loss of generality, σ can be assumed to be unity. Furthermore, the noise sources are assumed to be uncorrelated.

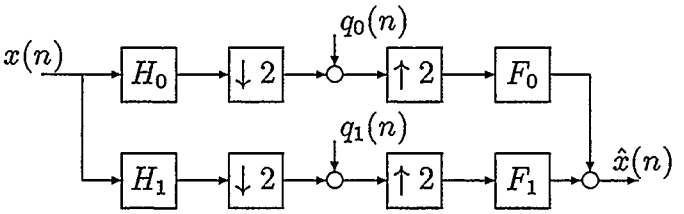

Figure 6.1 QMF Bank with Quantizers in the Subbands.

The $\mathcal{H}_2$ norm of an LTI system is the RMS value of the output when the inputs are driven by independent, zero-mean, unit-variance white noise sources [25]. This makes the $\mathcal{H}_2$ norm of R, the polyphase representation of the synthesis filters, a natural choice for quantifying the effects of subband quantization noise. This turns out to be a reasonable choice despite the expanders. The initial setup of the synthesis bank is shown in Figure 6.2. Since this system is time-varying, the use of the $\mathcal{H}_2$ norm of R

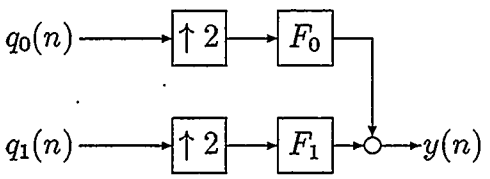

Figure 6.2 Noise Input to the Synthesis Bank.

requires some justification. Even when the inputs are white noise, the mean square ($E[y^2(n)]$) of the output $y(n)$ in Figure 6.2 is not independent of n. However, the quantity

$$J_2 := E[y^2(2k)] + E[y^2(2k-1)],$$

assuming that the system has reached steady-state, is independent of n, due to the periodicity of the system.

By taking the polyphase representation, Figure 6.2 becomes Figure 6.3, where,

$$q(n) = \begin{bmatrix} q_0(n) \\ q_1(n) \end{bmatrix},$$

U is the unit time delay, and $\mathbf{D}^*$ is the expander. The system in Figure 6.3 is still

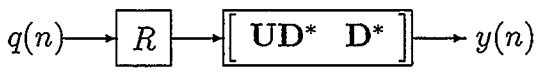

Figure 6.3 Polyphase Representation of Noise Model.

time-varying with periodicity 2, but it can be made time-invariant by blocking $y(n)$ [24]. This is shown in Figure 6.4, where,

$$\underline{y}(n) = \begin{bmatrix} y(2n) \\ y(2n-1) \end{bmatrix}.$$

The Noble Identities [7] must be used to show that Figure 6.4 is equivalent to

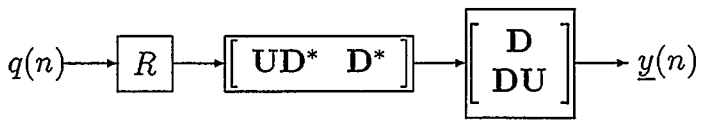

Figure 6.4 LTI Representation of Noise Model.

Figure 6.5.

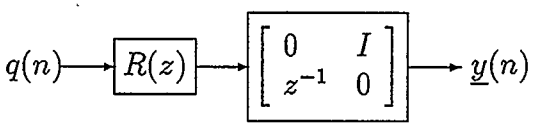

Figure 6.5 Equivalent LTI Representation of Noise Model.

The transfer function of the system shown in Figure 6.5 is

$$\hat{t}(z) = \begin{bmatrix} 0 & 1 \\ z^{-1} & 0 \end{bmatrix} R(z),$$

which in terms of the $\mathcal{H}_2$ norm implies

$$\|\hat{t}\|_2^2 = \|R\|_2^2.$$

By definition,

$$\|\hat{t}\|_2^2 = trace\left\{E[\underline{y}(n)\underline{y}'(n)]\right\},$$

which after substituting for $\underline{y}(n)$ and $\hat{t}$ becomes,

$$\|R\|_2^2 = E[y^2(2n)] + E[y^2(2n-1)] = J_2.$$

Thus, $\|R\|_2^2$ is a reasonable measure of the system's sensitivity to quantization noise.

In order to accommodate this measure, it is included in the cost function, J_{opt}, along with a weighting factor, α,

$$J = \|W - RE\|_\infty + \alpha\|R\|_2^2.$$

When the cost function for the ellipsoid algorithm is modified, the calculation of the subgradient must also be modified accordingly. The subgradient calculation for the $\mathcal{H}_2$ norm is straightforward since the $\mathcal{H}_2$ cost function is a differentiable function.

Since,

$$\varphi := \alpha \|R\|_2^2 = \alpha \left[r_{11}^2(0) + r_{12}^2(0) + \cdots + r_{22}^2(N) \right],$$

then the subgradient of φ is simply

$$g = \begin{bmatrix} \frac{\partial \varphi}{\partial r_{11}(0)} \\ \vdots \\ \frac{\partial \varphi}{\partial r_{22}(N)} \end{bmatrix},$$

which can be added to the subgradient previously calculated for $\mathcal{H}_\infty$.

By varying α the tradeoff between the $\mathcal{H}_\infty$ and $\mathcal{H}_2$ norms can be studied. This is illustrated in Figure 6.6 for the examples in Sections 4.1 and 5.4. Clearly a linear tradeoff exists. To get even a modest improvement in noise rejection (from 2 to 1.5), one must accept a 4 or 5-fold increase in reconstruction error. This appears to be the case for all systems studied, suggesting that, in these cases, it is not possible to significantly decrease a system's sensitivity to quantization noise without severe degradation in terms of the reconstruction error.

Another use for the $\mathcal{H}_2$ norm is in studying overall performance.

6.2 Overall Performance Measure

Using the $\mathcal{H}_2$ norm as a performance measure for the transmultiplexer is straightforward. In that case, the performance measure relates directly to the original LTI error setup G: the error energy is minimized for impulse inputs. The original QMF error system is time varying, which complicates matters slightly. However, the periodicity of 2 may be exploited in order to justify the choice of the $\mathcal{H}_2$ norm of the equivalent LTI error system, $W - RE$, as a performance measure.

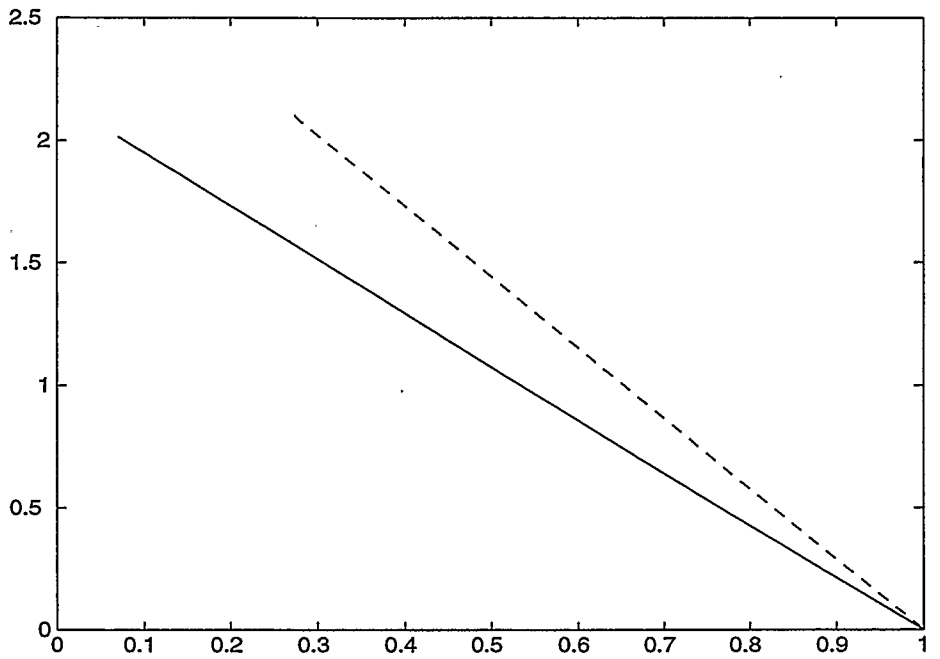

Figure 6.6 $\|R\|_2^2$ versus $\|W - RE\|_\infty$ for Unconstrained Setup (solid) and Alias-Free Setup (dashed).

6.2.1 Justification

A time-varying system does not have a unique impulse response. Instead, $g_\tau(n)$ is defined to be a shifted version of the response to the input $\delta(n-\tau)$. For a periodically time-varying system with periodicity 2,

$$\|g_0\|_2 = \|g_{2\tau}\|_2 \quad \|g_1\|_2 = \|g_{2\tau+1}\|_2, \tag{6.2}$$

where τ is an integer.

Define the following performance measure on the periodically time-varying error system,

$$J_{PTV} = \|g_0\|_2^2 + \|g_1\|_2^2.$$

Furthermore, J is defined as the performance measure for the equivalent LTI system (shown in Figure 3.2) $\underline{G}$,

$$J = \|\underline{G}\|_2^2.$$

The objective is to show that the performance measures defined for the time-varying and equivalent LTI systems are equal.

According to (6.1), the $\mathcal{H}_2$ norm for the 2×2 LTI error system $\underline{G}$ is

$$\|\underline{G}\|_2^2 = \|\underline{g}_{11}\|_2^2 + \|\underline{g}_{12}\|_2^2 + \|\underline{g}_{21}\|_2^2 + \|\underline{g}_{22}\|_2^2,$$

where, as in [16],

$$\begin{aligned} \underline{G}_{11} &= \mathbf{D}\mathbf{U}^*\underline{G}\mathbf{D}^* \\ \underline{G}_{12} &= \mathbf{D}\underline{G}\mathbf{D}^* \\ \underline{G}_{21} &= \mathbf{D}\mathbf{U}^*\underline{G}\mathbf{U}^*\mathbf{D}^* \\ \underline{G}_{22} &= \mathbf{D}\underline{G}\mathbf{U}^*\mathbf{D}^*. \end{aligned}$$

The operators $\mathbf{D}$, $\mathbf{D}^*$, and $\mathbf{U}^*$ refer to the decimator, expander, and unit time advance, respectively. In terms of the original system G, the impulse responses of the equivalent time-invariant error system $\underline{g}_{ik}(n)$ are given by,

$$\begin{aligned} \underline{g}_{11}(n) &= g_0(2n-1) \\ \underline{g}_{12}(n) &= g_0(2n) \\ \underline{g}_{21}(n) &= g_{-1}(2n-1) \\ \underline{g}_{22}(n) &= g_{-1}(2n). \end{aligned} \tag{6.3}$$

Using (6.2) and (6.3),

$$\begin{aligned} J &= \|\underline{G}\|_2^2 \\ &= \|g_0\|_2^2 + \|g_1\|_2^2 \\ &= J_{PTV} \end{aligned}$$

6.2.2 Optimization

The $\mathcal{H}_2$ problem formulation is the same as the problem formulation described in Section 3.1, except now, instead of using the $\mathcal{H}_\infty$ norm to quantify the error, the $\mathcal{H}_2$ norm is used. Thus, the design problem becomes:

For W and E as defined in Section 3.1, design a stable and causal R by minimizing $\|W - RE\|_2$.

The $\mathcal{H}_2$ norm differs from the $\mathcal{H}_\infty$ norm; in general, the $\mathcal{H}_2$ norm optimization problem can be solved in one iteration. As with the $\mathcal{H}_\infty$ problem formulation, the $\mathcal{H}_2$ problem is of the form in (3.6). Therefore, the $\mathcal{H}_2$ problem formulation is a quadratic problem of the form,

$$f(x) = x^T Ax + Bx + C,$$

which is readily solved. The program *QDES* [29], based on the approach proposed in [30], is used to solve the MIMO $\mathcal{H}_2$ optimization problem.

6.2.3 Design Example

The simple design example from Section 4.1 is used here. The only difference is that the $\mathcal{H}_2$ norm is now used as the performance measure. The resulting synthesis filter coefficients are given in Table 6.1. Although there was no linear-phase requirement, the optimal synthesis filters are clearly linear-phase. The magnitude responses of the analysis and synthesis filters are given in Figure 6.7. To illustrate the performance of the system, Figures 6.8 and 6.9 show the distortion plots and the error plot, respectively. As expected, the desired transfer function $T(z)$ is close to unity, and the aliasing, $A(z)$, is small. The error plot shows that, compared to the input, the error resulting from a ramp is small. A quantitative comparison of the performance measures is given in Table 6.2. The second row in Table 6.2 summarizes the performance results of the same setup, but using the $\mathcal{H}_\infty$ norm as a performance metric, as described in Section 4.1. Clearly, the resulting system is not optimal in

n	$f_0(n)$	$f_1(n)$
0	-0.00019156	0.00019156
1	-0.03424780	-0.03424780
2	-0.00189113	0.00189113
3	0.05745363	0.05745363
4	0.00188819	-0.00188819
5	-0.09705241	-0.09705241
6	0.00362549	-0.00362549
7	0.15402781	0.15402781
8	-0.03722723	0.03722723
9	-0.25744338	-0.25744338
10	0.20373956	-0.20373956
11	0.99543452	0.99543452
12	0.99543452	-0.99543452
13	0.20373956	0.20373956
14	-0.25744338	0.25744338
15	-0.03722723	-0.03722723
16	0.15402781	-0.15402781
17	0.00362549	0.00362549
18	-0.09705241	0.09705241
19	0.00188819	0.00188819
20	0.05745363	-0.05745363
21	-0.00189113	-0.00189113
22	-0.03424780	0.03424780
23	-0.00019156	-0.00019156

Table 6.1 Coefficients of 24-tap Synthesis Filters for $\mathcal{H}_2$ Performance Measure

terms of the $\mathcal{H}_\infty$ norm, since in Section 4.1 the $\mathcal{H}_\infty$ norm was 0.0302, as compared to 0.0475 by using the $\mathcal{H}_2$ norm as a performance measure. However, for a ramp input, the design using the $\mathcal{H}_2$ norm appears preferable, because the SNR_r is lower for this design than the design in Section 4.1.

The design example from Section 4.4 is used to examine the effect of filter length on the performance-versus-delay curves, as shown in Figure 6.10. Since the $\mathcal{H}_2$ per-

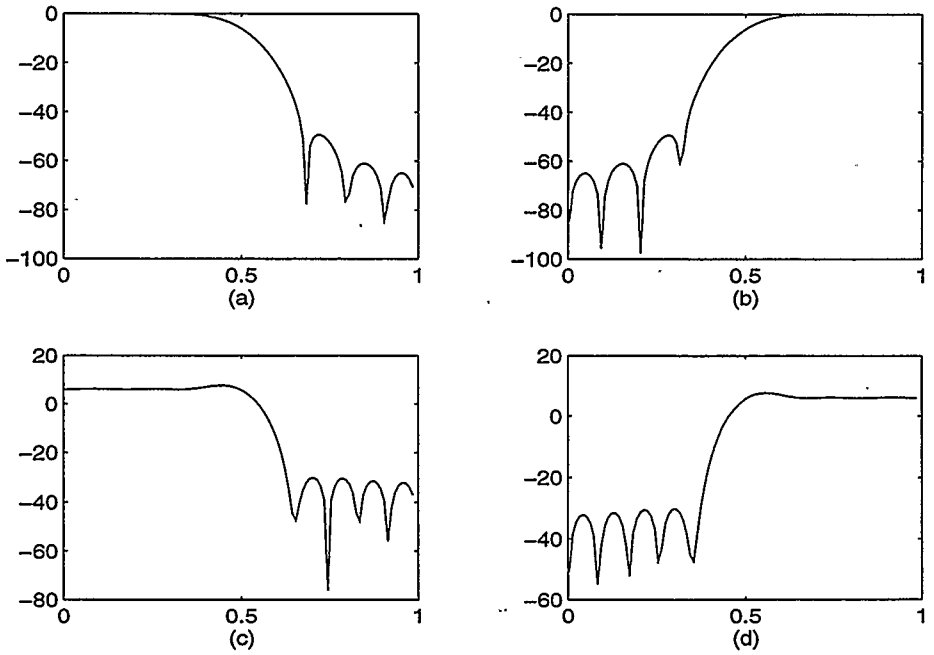

Figure 6.7 $\mathcal{H}_2$ Norm: Magnitude Responses in dB versus ω/π. (a) H_0. (b) H_1. (c) F_0. (d) F_1.

$\mathcal{H}_2$ Norm	$\mathcal{H}_\infty$ Norm	SNR_r	$\|\mathbf{AS}-\mathbf{B}\|_F$	Mag. Error	Phase Error	Aliasing
0.00112	0.0475	7.4E-05	0.00112	0.047	9.1×10^{-13} °	-27.51 dB
0.00177	0.0302	0.020226	0.0017689	0.0302	0.096°	-30.39 dB

Table 6.2 Comparison of Performance Measures for the Simple QMF Example, Using $\mathcal{H}_2$ Norm and $\mathcal{H}_\infty$ Norm.

formance measure can be solved analytically, the symmetry observed in Figure 6.10 is more easily explained than the corresponding plot for the $\mathcal{H}_\infty$ norm in Section 4.4. Table 6.3 shows the coefficients of the synthesis filters for the 8-tap synthesis filters at delays of 5 and 11 samples, with $J_{opt} = 0.0702$. Whereas, in Table 4.5 the rela-

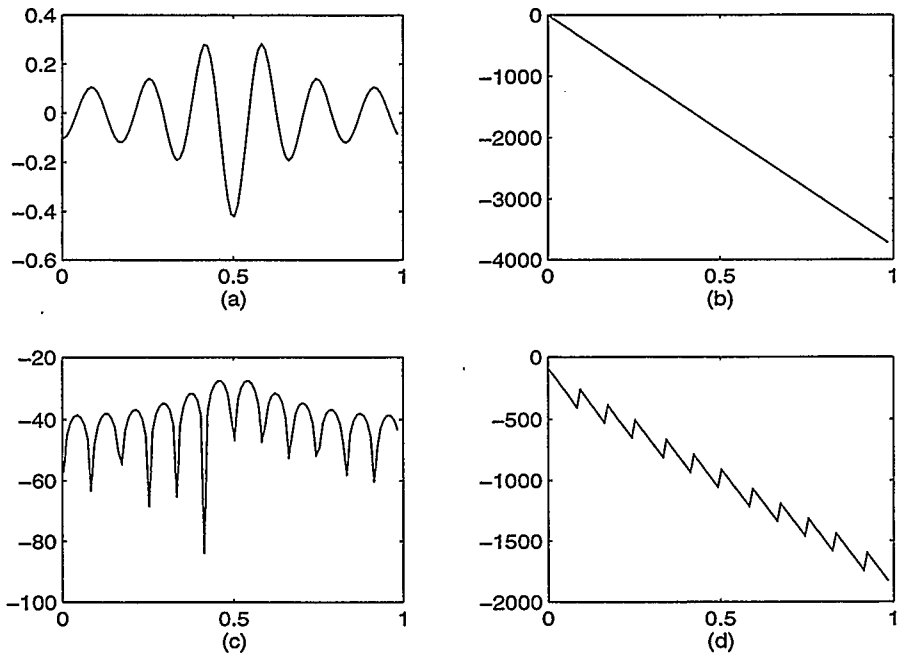

Figure 6.8 $\mathcal{H}_2$ Performance Measure: Magnitude (dB) Distortions for $T(z)$ and $A(z)$ ((a) and (c)) versus ω/π. Phase (°) Distortions for $T(z)$ and $A(z)$ ((b) and (d)) versus ω/π.

tionship between the synthesis filters is somewhat distorted due to numerical errors, in Table 6.3 the relationship is obvious. Clearly, the coefficients of $f_0(n)$ and $f_1(n)$ for delay $= 11$ are symmetrical and anti-symmetrical versions, respectively, of $f_0(n)$ and $f_1(n)$ for delay $= 5$.

The explanation of the symmetries exhibited in Figure 6.10 and Table 4.5 involves the symmetries in the original setup. In particular, the even-length analysis filters are linear-phase and are related by $H_1(z) = H_0(-z)$. The symmetries in the analysis filters are used in the following explanation. To simplify the explanation, the delay is assumed to be odd ($m = 2d + 1$). A similar derivation holds for even delays.

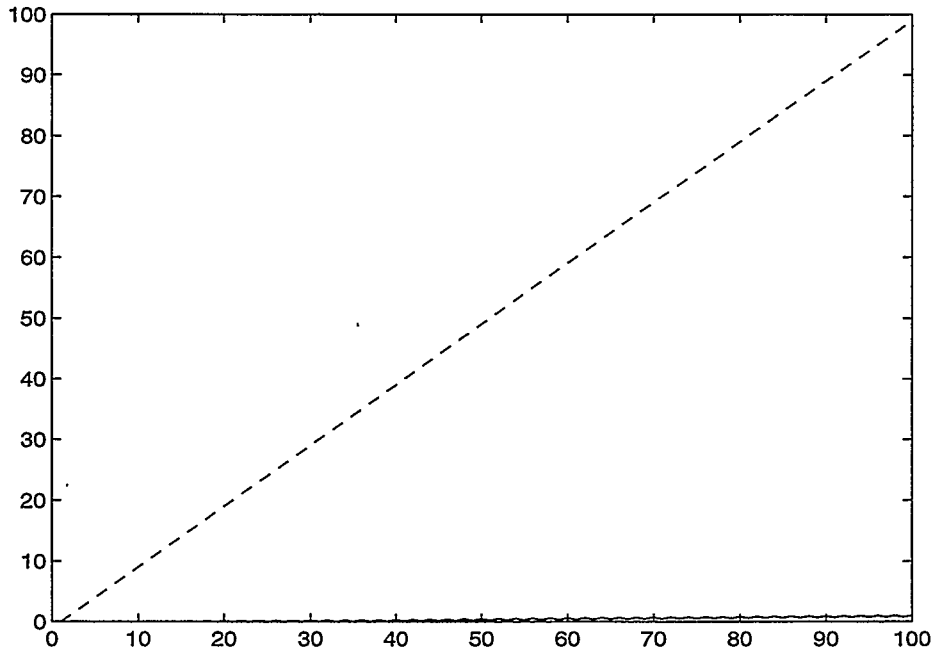

Figure 6.9 $\mathcal{H}_2$ Performance Measure: Input Ramp (dash) and Reconstruction Error (solid) versus n.

The original model-matching problem for delay $m = 2d + 1$ is

$$\begin{bmatrix} G_{11}(z) & G_{12}(z) \\ G_{21}(z) & G_{22}(z) \end{bmatrix} = \begin{bmatrix} z^{-d} & 0 \\ 0 & z^{-d} \end{bmatrix} - \begin{bmatrix} R_{00}(z) & R_{10}(z) \\ R_{01}(z) & R_{11}(z) \end{bmatrix} \begin{bmatrix} E_{00}(z) & E_{01}(z) \\ E_{10}(z) & E_{11}(z) \end{bmatrix}. \quad (6.4)$$

Since the analysis filters are related by $H_1(z) = H_0(-z)$, the polyphase representations of the analysis filters are also related:

$$E_{10}(z) = E_{00}(z),$$

and

$$E_{11}(z) = -E_{01}(z).$$

Using the relationship between the polyphase representations and rewriting (6.4), the

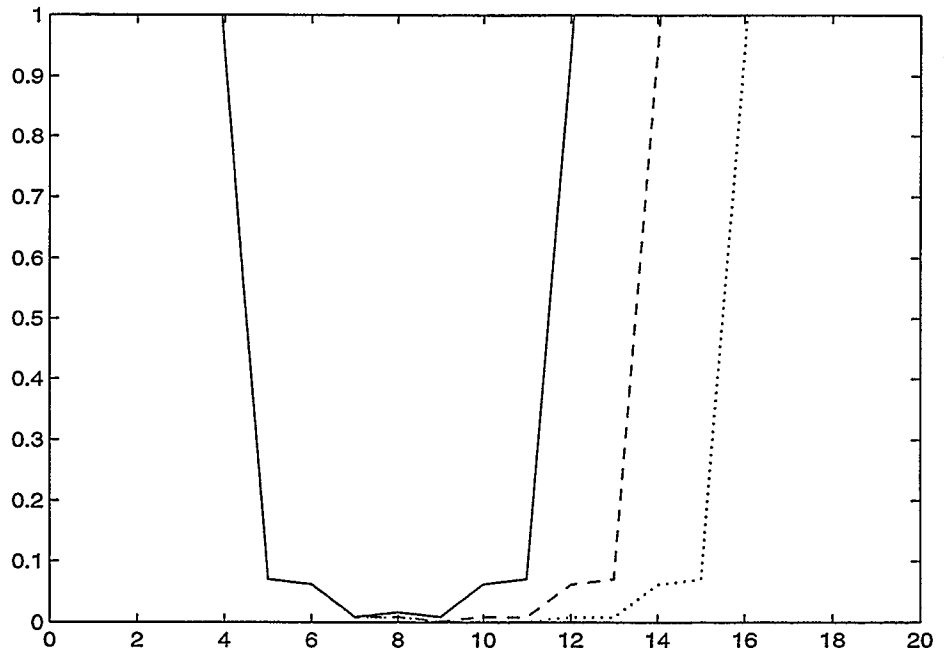

Figure 6.10 $\mathcal{H}_2$ Performance Measure: J_{opt} versus Delay for 8-tap (solid), 10-tap (dashed), and 12-tap (dotted) Synthesis Filters.

right hand side of (6.4) can be simplified to

$$\begin{bmatrix} z^{-d} - (R_{00}(z) + R_{10}(z))\, E_{00}(z) & -(R_{00}(z) - R_{10}(z))\, E_{01}(z) \\ -(R_{01}(z) + R_{11}(z))\, E_{00}(z) & z^{-d} - (R_{01}(z) - R_{11}(z))\, E_{01}(z) \end{bmatrix}.$$

For notational convenience, define

$$R_1(z) = R_{00}(z) + R_{10}(z), \quad R_2(z) = R_{00}(z) - R_{10}(z),$$

$$R_3(z) = R_{01}(z) + R_{11}(z), \quad R_4(z) = R_{01}(z) - R_{11}(z).$$

In general, the cost function for the $\mathcal{H}_2$ optimization is

$$J(d) = \|G_{11}(d)\|_2^2 + \|G_{12}\|_2^2 + \|G_{21}\|_2^2 + \|G_{22}(d)\|_2^2. \tag{6.5}$$

Delay = 5		Delay = 11	
$f_0(n)$	$f_1(n)$	$f_0(n)$	$f_1(n)$
1.04707710	-1.04707710	-0.03174667	0.03174667
1.11273170	1.11273170	0.00113632	0.00113632
0.12608338	-0.12608338	0.10157315	-0.10157315
-0.28351721	-0.28351721	-0.00628066	-0.00628066
-0.00628066	0.00628066	-0.28351721	0.28351721
0.10157315	0.10157315	0.12608338	0.12608338
0.00113632	-0.00113632	1.11273170	-1.11273170
-0.03174667	-0.03174667	1.04707710	1.04707710

Table 6.3 8-tap Synthesis Filter Coefficients Using $\mathcal{H}_2$ Norm.

where

$$\begin{aligned} G_{11}(d,z) &= z^{-d} - R_1(z)E_{00}(z), \\ G_{12}(z) &= -R_2(z)E_{01}(z), \\ G_{21}(z) &= -R_3(z)E_{00}(z), \\ G_{22}(d,z) &= z^{-d} - R_4(z)E_{01}(z). \end{aligned}$$

Similarly for $J(n_E + n_R - d)$, the transfer matrices G_{ij} are defined as

$$\begin{aligned} G_{11}(n_E+n_R-d,z) &= z^{-(n_E+n_R-d)} - R_1(z)E_{00}(z), \\ G_{12}(z) &= -R_2(z)E_{01}(z), \\ G_{21}(z) &= -R_3(z)E_{00}(z), \\ G_{22}(n_E+n_R-d,z) &= z^{-(n_E+n_R-d)} - R_4(z)E_{01}(z), \end{aligned}$$

where n_E and n_R are the orders of the FIR $E(z)$ and the FIR $R(z)$, respectively. To show that $J_{opt}(d) = J_{opt}(n_E + n_R - d)$, the cost functions for these two different delays must have the same form. For both cost functions, $G_{12}(z)$ and $G_{21}(z)$ are independent of d, consequently from an optimization point of view, they must have the same form.

For even-length analysis filters with linear-phase

$$E_{00}(z) = z^{-n_E} E_{01}(z^{-1}).$$

Then $G_{11}(n_E + n_R - d, z)$ can be rewritten:

$$\begin{aligned} G_{11}(n_E + n_R - d, z) &= z^{-(n_E+n_R-d)} - R_1(z) z^{-n_E} E_{01}(z^{-1}) \\ &= z^{-n_E} \left[z^{-(n_R-d)} - R_1(z) E_{01}(z^{-1}) \right]. \end{aligned}$$

The order of the coefficients of $R_1(z)$ can be reversed by replacing $R_1(z)$ with $z^{-n_R} R_1(z^{-1})$: If $R_1(z)$ is given by

$$R_1(z) = r_1(0) + r_1(1) z^{-1} + \cdots + r_1(n_R) z^{-n_R}$$

then

$$z^{-n_R} R_1(z^{-1}) = r_1(n_R) + r_1(n_R - 1) z^{-1} + \cdots + r_1(0) z^{-n_R}. \tag{6.6}$$

Since R_1 is a variable, changing the order of the coefficients does not affect the minimum of $G_{11}(n_E + n_R - d, z)$. However, the interpretation of the minimizers will change, as indicated by (6.6). Accordingly, the optimization is unchanged by replacing $G_{11}(n_E + n_R - d, z)$ with

$$\begin{aligned} & z^{-n_E} \left[z^{-(n_R-d)} - z^{-n_R} R_1(z^{-1}) E_{01}(z^{-1}) \right] \\ = \quad & z^{-(n_E+n_R)} \left[z^d - R_1(z^{-1}) E_{01}(z^{-1}) \right]. \end{aligned} \tag{6.7}$$

Furthermore, replacing z with z^{-1} will not change the optimization outcome. Thus, rewriting the quantity in (6.7) as

$$z^{(n_E+n_R)} \left[z^{-d} - R_1(z) E_{01}(z) \right] \tag{6.8}$$

will not change the optimization problem. Since the time advance $z^{(n_E+n_R)}$ does not affect the optimization, (6.8) has the same format as $G_{22}(d, z)$. Similarly, $G_{22}(n_E +$

$n_R - d, z)$ and $G_{11}(d, z)$ are related. Thus, both optimizations are a summation of the same transfer matrices, except the order of some of the transfer matrices is interchanged:

$$\begin{aligned} J_{opt}(d) &= \min_{R_k} \left(\|G_{11}(d)\|_2^2 + \|G_{12}\|_2^2 + \|G_{21}\|_2^2 + \|G_{22}(d)\|_2^2 \right) \\ &= \min_{R_k} \left(\|G_{22}(d)\|_2^2 + \|G_{12}\|_2^2 + \|G_{21}\|_2^2 + \|G_{11}(d)\|_2^2 \right) \\ &= \min_{R_k} \left(\|G_{11}(n_E + n_R - d)\|_2^2 + \|G_{12}\|_2^2 + \|G_{21}\|_2^2 + \|G_{22}(n_E + n_R - d)\|_2^2 \right) \\ &= J_{opt}(n_E + n_R - d). \end{aligned}$$

The only difference between the two cost functions is the interpretation of R_k. A similar explanation holds for even delays. Thus, the symmetry in Figure 6.10 is a result of choosing even-length analysis filters with linear-phase, related by $H_0(z) = H_1(-z)$.

CHAPTER 7

CONCLUSIONS

7.1 Summary

Model-matching techniques provide a novel solution to the QMF design problem. Many traditional QMF design methods place restrictions on the analysis filters. By reformulating the problem into a convex optimization problem these restrictions are removed. The FIR model-matching method decouples the design of the analysis and synthesis filters by assuming that the analysis filters have been previously designed. Throughout the optimization process the analysis filters remain fixed. For this reason, the FIR model-matching method provides a valuable tool for studying the tradeoffs involved with various requirements, such as reconstruction properties, synthesis filter length, delay, linear-phase synthesis filters, cosine-modulated synthesis filters, or alias-free systems.

This method was first proposed in [16]. In that work the synthesis filters were IIR, and the designer had no control over the size of the filters. Optimizing the coefficients of the synthesis filters, as in the proposed FIR model-matching method, results in greater control over these filters.

Another advantage offered by the FIR model-matching method is that it can be used in conjunction with other design methods. Specifically, low-delay systems are difficult to design, but by using the analysis filters designed by the method in [12], the FIR model-matching method may be able to design synthesis filters resulting in better performance. Yet another advantage is the convex nature of the design

problem, which guarantees that the minimum found by the numerical optimization is the global minimum.

The FIR model-matching method also allows for a weighted cost function involving both the reconstruction performance and the system's sensitivity to quantization noise. The $\mathcal{H}_2$ norm is a useful measure of a system's sensitivity to quantization noise. Another application of the $\mathcal{H}_2$ norm is as a performance measure of the overall system. Unlike the $\mathcal{H}_\infty$ norm, the $\mathcal{H}_2$ norm is not a worst-case norm; however, it may still be useful when the system is designed for standard test signals, or when the input signal can be modeled as a stationary stochastic process.

The model-matching concept is useful not only for QMF design, but also for A/D conversion using hybrid QMF filter banks, as in [26], and transmultiplexers. The FIR model-matching method is easily modified to accommodate these design problems.

To summarize, the main advantages of the model-matching FIR model-matching method are:

- reduced restrictions on analysis filters
- good for studying tradeoffs
- easy to handle constraints on synthesis filters:
  - linear-phase
  - cosine-modulated
- easy to handle constraints on the system:
  - low-delay
  - alias-free
- global optimality guaranteed.

7.2 Future Work

Future research can be categorized into two distinct areas: generalizing the setup and changing the design process. The most obvious example of generalizing the setup is to extend the method to M-channels. Prior to generalizing the setup, the efficiency of the optimizing algorithm must be improved. For large unconstrained problems the ellipsoid algorithm, implemented in MATLAB, is exceedingly slow. Another possible modification to the setup would be to allow arbitrary decimation and interpolation factors.

The investigation of new constraints belongs to the second area of future work. One possible constraint would be a linear-phase requirement for the prototype filter used with the cosine-modulated synthesis filters. Specific requirements of a system could also be exploited. In some cases the distortion around $\omega = 0$ and $\omega = \pi$ is not critical and in these cases a weighted $\mathcal{H}_\infty$ norm may be useful. Lastly, the design process could be altered to include the design of the analysis filters. This change would require some convex frequency domain constraints on the analysis filters. After the synthesis filters have been optimized for a given set of analysis filters, the analysis filters could be optimized. This process would then be repeated until the system has the desired characteristics.

In conclusion, the FIR model-matching technique provides a useful and flexible tool for studying and designing various multirate filter bank systems.

APPENDIX A

MATLAB CODE FOR CALCULATING SUBGRADIENTS

The optimization was implemented in MATLAB Version 4.2a [31]. In addition, the μ-Analysis and Synthesis Toolbox [32] and Control System Toolbox [33] were used in calculating the subgradients.

A.1 Ellipsoid Algorithm

```
% using ellipsoid algorithhm in boyd p326
% for finding || ||_inf for MIMO

% November 24, 1994

if odd == 0,
     n_tap_q = f_tap/2; % divide by 2 since half of
                                % taps  in Q (doubles later) (net *2)
     n_tap_eff = n_tap_q*4;     % times 4 because q is 2 by 2
else
     f_tap = f_tap +1;          % last tap to zero in synth filters
     n_tap_q = f_tap/2;
     n_tap_eff = n_tap_q*4 - 2; % adjust, since have 2 0's in q
end

q_new = [0.0001*ones(1,n_tap_eff)]'; % not quite zero to avoid / by zero
A_new= R*R*eye(n_tap_eff);

k = 0;
test = 100;
phi = 100;

anal                            % define anal filters
setup                           % get T1 and T2
```

```
[T1a, T1b, T1c, T1d] = d2cm(t1a, t1b, t1c, t1d, 2, 'tustin');
[T2a, T2b, T2c, T2d] = d2cm(t2a, t2b, t2c, t2d, 2, 'tustin');

while ((test > test_stop) & (phi > phi_stop)),
        k = k+1;
        q = q_new;
        A = A_new;
        [phi,phi_hinf, phi_h2, grad] = getsg(T1a, T1b, T1c, T1d, ...
                                T2a, T2b, T2c, T2d, q, n_tap_q, odd, noise);
        test = sqrt(grad'*A*grad);
        grad_new = grad/sqrt(grad'*A*grad);
        q_new = q - A*grad_new/(n_tap_eff + 1);
        A_new = (n_tap_eff*n_tap_eff/(n_tap_eff*n_tap_eff - 1))*...
                (A - 2/(n_tap_eff+1)*A*grad_new*grad_new'*A);
        if rem(k,100) == 0,
                save inter_strt10 A_new test grad_new phi q_new,
        end
end
```

A.2 Subgradient: Unconstrained

```
% calculates subgradient for cost function = hinf + noise*h2^2
% for BOTH even and odd,
% and don't care about linear phase in synth filters
% [phi, phi_hinf, phi_h2, grad] = getsg(t1a, t1b, t1c, t1d,
%                           t2a, t2b, t2c, t2d, q, n_tap, odd, noise)
% the function to be minimized has the form:
% || H ||_inf
% || T1 + QT2 ||_inf  which can be broken down into
% || T1 + [q0 0; 0 0]T2 + [0 q1;0 0]T2 + ...
%       + z^-1[q5 0; 0 0]T2 + ...||
% the first step is to determine a subgradient at Ho, which is
% the transfer fn at some arbitrary initial guess of Q
% (which is 2 by 2)
% it turns out that the subgradient is
% g = [phi_sg(H1) phi_sg(H2) ...]'
% where H1 = T2, H2 = z^-1T2, ...

% November 28, 1994
```

```
function [phi, phi_hinf, phi_h2, grad] = getsg(T1a, T1b, T1c, T1d, ...
                              T2a, T2b, T2c, T2d, q, n_tap, odd, noise);
% n_tap here means # of taps in q

%only go to second last tap,
%because may need to insert zeros for odd synth
for l = 1:n_tap-1;
        qimp(l,1:4) = [q(4*l-3,1) q(4*l-2,1) q(4*l-1,1) q(4*l,1)];
end

size_q = size(q,1);
if odd == 0,
        qimp(n_tap,1:4) = [q(size_q-3,1) q(size_q-2,1) ...
                           q(size_q-1,1) q(size_q,1)];
else
        qimp(n_tap,1:4) = [0 0 q(size_q-1,1) q(size_q,1)];
end

[qa, qb, qc, qd] = fir2ss(qimp, 2, 2);          % Q is 2 by 2 matrix

% ----------- 1) Determine Ho(jWo) ----------------
% bilinear trans  to continuous time

[Qa, Qb, Qc, Qd] = d2cm(qa, qb, qc, qd, 2, 'tustin');

% now multiply  and add to T1
[t2qa, t2qb, t2qc, t2qd] = series(T2a, T2b, T2c, T2d, Qa, Qb, Qc, Qd);

% calc Hinf norm
[ha, hb, hc, hd]=parallel(t2qa, t2qb, t2qc, t2qd, T1a, T1b, T1c, T1d);
system = pck(ha, hb, hc, hd);   % packed notation
out = hinfnorm(system);
Wo = out(1,3);
phi_hinf = out(1,1);

% evaluate stuff at this freq.
if isinf(Wo) == 0,
        HojWo = hc*inv((0+j*Wo)*eye(size(ha)) - ha)*hb + hd;
else
        HojWo = hd;
end
```

```
% get singular value decomposition of HojWo (p304 BOYD)
[U, S, V] = svd(HojWo);
uo = U(:,1);
uo = ((uo))'; % need in this format for formula
vo = V(:,1);


% -------------- 2) Now get T1(jWo), T2(jWo) ------------
if isinf(Wo) == 0,
        T1 = T1c*inv((0+j*Wo)*eye(size(T1a)) - T1a)*T1b + T1d;
        T2 = T2c*inv((0+j*Wo)*eye(size(T2a)) - T2a)*T2b + T2d;
        factor = (1-j*Wo)/(1+j*Wo);
else
        T2 = T2d;
        T1 = T1d;
        factor = -1;
end

% subgradient for H2 norm is straight forward, just take deriv
% (phi_h2)^2 = q1^2 + q2^2 + ... + qN^2
% where N is n_tap_q*4
% then the subgradient is
% dphi_h2/dq1 = 2*r1, evaluated at the current guess
% the rest of them are similar, have to remember to multiply by
% weighting factor, and finally just add to subgrad for hinf

% calculate h2 norm
phi_h2 = normest(q);
constant = noise*2;   % factor for H2 subgradient if use H2 norm sq

phi_sg = [];

% Now need to look at finite dim space
% so follow proc on p307 BOYD
% H1 = [1 0; 0 0]*T2, ..., H5 = [1 0; 0 0]*z^-1*T2, ...
for l = 1:n_tap,
        new_fact = factor^(l-1);
        if (l == n_tap) & (odd == 1),
            Q(1:2,1:4) = [0        0 0 0
                          new_fact 0 0 new_fact];
```

```
            phi_sg1  = real(uo*Q(1:2,1:2)*T2*vo);
            phi_sg2  = real(uo*Q(1:2,3:4)*T2*vo);
            phi_sg = [phi_sg;
                      phi_sg1+constant*q(4*l-3,1);
                      phi_sg2+constant*q(4*l-2,1)];
        else
            Q(1:2,1:8) = [new_fact 0 0 new_fact 0        0 0 0
                          0        0 0 0        new_fact 0 0 new_fact];
% according to p304, subgrad = real((uo*)(H)(v))
% where H is defined as above comment
                phi_sg1  = real(uo*Q(1:2,1:2)*T2*vo);
                phi_sg2  = real(uo*Q(1:2,3:4)*T2*vo);
                phi_sg3  = real(uo*Q(1:2,5:6)*T2*vo);
                phi_sg4  = real(uo*Q(1:2,7:8)*T2*vo);
                phi_sg = [phi_sg
                          phi_sg1+constant*q(4*l-3,1);
                          phi_sg2+constant*q(4*l-2,1);
                          phi_sg3+constant*q(4*l-1,1);
                          phi_sg4+constant*q(4*l,1)];
        end
end
% return fn value (inf norm + noise*h2norm)
phi = phi_hinf + noise*(phi_h2)^2;
grad =  phi_sg;
```

A.3 Subgradient: Linear-Phase Constraint

A.3.1 Even Length Synthesis Filters

```
% calculates subgradient for cost function = hinf + noise*h2^2
% LINEAR PHASE, assuming N odd (ie n_tap even)
% [phi, phi_hinf, phi_h2, grad] = getsg(t1a, t1b, t1c, t1d,
%                          t2a, t2b, t2c, t2d, q, n_tap, odd, noise)
% the function to be minimized has the form:
% || H ||_inf
% || T1 + QT2 ||_inf  which can be broken down into
% || T1 + [q0 0; 0 0]T2 + [0 q1;0 0]T2 + ...
%       + z^-1[q5 0; 0 0]T2 + ...||
% the first step is to determine a subgradient at Ho, which is
```

```
% the transfer fn at some arbitrary initial guess of Q
% (which is 2 by 2)
% it turns out that the subgradient is
% g = [phi_sg(H1) phi_sg(H2) ...]'
% where H1 = T2, H2 = z^-1T2, ...

% November 28, 1994

function [phi,phi_hinf,phi_h2, grad] = ...
                        getsg_lpeven(T1a, T1b, T1c, T1d, ...
                        T2a, T2b, T2c, T2d, q, n_tap,noise);
% NOTE n_tap refers to nubmer of taps in synth!

% need double the coeffs in q,
% but they are just functions of the other half
qtemp = q;
for l = 1:2:n_tap-1,
        qtemp = [qtemp; q(n_tap-l,1); -q(n_tap-l+1,1)];
end

% now ready to get into proper format
for l = 1:n_tap/2;              % divide by 2 since have half coeffs
                                % in Q since later these will double
        qimp(l,1:4) = [qtemp(4*l-3,1) qtemp(4*l-2,1)...
                       qtemp(4*l-1,1) qtemp(4*l,1)];
end

[qa, qb, qc, qd] = fir2ss(qimp, 2, 2);          % Q is 2 by 2 matrix

% ----------- 1) Determine Ho(jWo) ----------------
% bilinear trans  to continuous time

[Qa, Qb, Qc, Qd] = d2cm(qa, qb, qc, qd, 2, 'tustin');

% now multiply  and add to T1
[t2qa, t2qb, t2qc, t2qd] = series(T2a, T2b, T2c, T2d, Qa, Qb, Qc, Qd);

[ha, hb, hc, hd]=parallel(t2qa, t2qb, t2qc, t2qd, T1a, T1b, T1c, T1d);
system = pck(ha, hb, hc, hd);   % packed notation
out = hinfnorm(system);
Wo = out(1,3);
```

```
% evaluate stuff at this freq.
if isinf(Wo) == 0,
        HojWo = hc*inv((0+j*Wo)*eye(size(ha)) - ha)*hb + hd;
else
        HojWo = hd;
end


% get singular value decomposition of HojWo (p304 BOYD)
[U, S, V] = svd(HojWo);
uo = U(:,1);
uo = ((uo))';

vo = V(:,1);


% -------------- 2) Now get T1(jWo), T2(jWo) ------------
if isinf(Wo) == 0,
        T1 = T1c*inv((0+j*Wo)*eye(size(T1a)) - T1a)*T1b + T1d;
        T2 = T2c*inv((0+j*Wo)*eye(size(T2a)) - T2a)*T2b + T2d;
        factor = (1-j*Wo)/(1+j*Wo);
else
        T1 = T1d;
        T2 = T2d;
        factor = -1;
end

% subgradient for H2 norm is straight forward, just take deriv
% (phi_h2)^2 = q1^2 + q2^2 + ... + qN^2
% where N is n_tap_q*4, but in this case, half the coeffs are in
% terms of the other half so...
% (phi_h2)^2 = 2*q1^2 + 2*q2^2 + ... + 2*(q(N/2))^2
% then the subgradient is
% dphi_h2/dq1 = 4*r1, evaluated at the current guess
% the rest of them are similar, have to remember to multiply by
% weighting factor, and finally just add to subgrad for hinf

% calculate h2 norm
phi_h2 = normest(qtemp);        % have to do 'whole' q
sg_h2 = [];

for k = 1:n_tap,                % step through all unique q
```

```
        sg_h2 = [sg_h2; 4*noise*q(k)];
end

raised = floor(n_tap/2);          % NOTE n_tap means # taps in synth
phi_sg = [];                      % just need to give phi_sg size = 0

if floor(n_tap/4) ~= ceil(n_tap/4),
     n_tap_temp = n_tap + 2; % to get right # of iter in for loop
else
     n_tap_temp = n_tap;
end
for k = 0:floor(n_tap_temp/4)-1, % NOTE n_tap is # of taps in synth
   Q(1:2,1:4) = [factor^k          0 0 factor^k
                 factor^(raised-k-1) 0 0 -(factor^(raised-k-1))];
   phi_sg = [phi_sg; real(uo*Q(1:2,1:2)*T2*vo)];
   phi_sg = [phi_sg; real(uo*Q(1:2,3:4)*T2*vo)];

    if 4*(k+1) <= n_tap,                % for 4, 8, 12, ... taps
      Q(1:2,1:4) = [factor^(raised-k-1) 0 0 -(factor^(raised-k-1))
                    factor^k            0 0 factor^k];
      phi_sg = [phi_sg; real(uo*Q(1:2,1:2)*T2*vo)];
      phi_sg = [phi_sg; real(uo*Q(1:2,3:4)*T2*vo)];
   end
end

grad = phi_sg + sg_h2;
% return fn value (here inf norm)
phi_hinf = out(1,1);
phi = phi_hinf + noise*(phi_h2)^2;
```

A.3.2 Odd Length Synthesis Filters

```
% calculates subgradient for cost function = hinf + noise*h2^2
% LINEAR PHASE, assuming N even (ie n_tap ODD)
% [phi, phi_hinf, phi_h2, grad] = getsg(t1a, t1b, t1c, t1d,
%                          t2a, t2b, t2c, t2d, q, n_tap, odd, noise)
% the function to be minimized has the form:
% || H ||_inf
% || T1 + QT2 ||_inf  which can be broken down into
% || T1 + [q0 0; 0 0]T2 + [0 q1;0 0]T2 + ...
```

```
%       + z^-1[q5 0; 0 0]T2 + ...||
% the first step is to determine a subgradient at Ho, which is
% the transfer fn at some arbitrary initial guess of Q
% (which is 2 by 2)
% it turns out that the subgradient is
% g = [phi_sg(H1) phi_sg(H2) ...]'
% where H1 = T2, H2 = z^-1T2, ...

% November 28, 1994

function [phi, phi_hinf, phi_h2, grad] = ...
                        getsg_lpodd(T1a, T1b, T1c, T1d, ...
                         T2a, T2b, T2c, T2d, q, n_tap, noise);
% NOTE n_tap refers to number of taps in synth!
n_tap_o = n_tap-1;      % # of taps in odd filter (not counting 0)

% need double the coeffs in q,
% but they are just functions of the other half
f0 = [];
f1 = [];
for k = 1:4:size(q,1)-3,
        f0 = [f0; q(k+2,1); q(k,1)];
        f1 = [f1; q(k+3,1); q(k+1,1)];
end
if k < size(q,1),       % still need to add one to synth filters
        f0 = [f0; q(size(q,1)-1,1)];
        f1 = [f1; q(size(q,1),1)];
end

temp = size(f0,1)-1;            % have symmetry around centre coeff
f0 = [f0; flipud(f0(1:temp,:)); 0];
f1 = [f1; flipud(f1(1:temp,:)); 0];

qtemp = [];
% need to figure out q (note that f0 and f1 have same # of taps)
for k = 1:2:n_tap-1,
        qtemp = [qtemp; f0(k+1); f1(k+1); f0(k); f1(k)];
end

% now ready to get into proper format
for l = 1:n_tap/2;              % divide by 2 since have half coeffs
                                % in Q since later these will double
```

```
        qimp(1,1:4) = [qtemp(4*l-3,1) qtemp(4*l-2,1)...
                       qtemp(4*l-1,1) qtemp(4*l,1)];
end

[qa, qb, qc, qd] = fir2ss(qimp, 2, 2);          % Q is 2 by 2 matrix

% ----------- 1) Determine Ho(jWo) ----------------
% bilinear trans  to continuous time

[Qa, Qb, Qc, Qd] = d2cm(qa, qb, qc, qd, 2, 'tustin');

% now multiply  and add to T1
[t2qa, t2qb, t2qc, t2qd] = series(T2a, T2b, T2c, T2d, Qa, Qb, Qc, Qd);

[ha, hb, hc, hd]=parallel(t2qa, t2qb, t2qc, t2qd, T1a, T1b, T1c, T1d);
system = pck(ha, hb, hc, hd);   % packed notation
%out = hinfnorm(system,0.001,0); don't dp this way, may give probs
out = hinfnorm(system);
Wo = out(1,3);

% evaluate stuff at this freq.
if isinf(Wo) == 0,
        HojWo = hc*inv((0+j*Wo)*eye(size(ha)) - ha)*hb + hd;
else
        HojWo = hd;
end


% get singular value decomposition of HojWo (p304 BOYD)
[U, S, V] = svd(HojWo);
uo = U(:,1);
uo = ((uo))';

vo = V(:,1);


% -------------- 2) Now get T1(jWo), T2(jWo) ------------
if isinf(Wo) == 0,
        T1 = T1c*inv((0+j*Wo)*eye(size(T1a)) - T1a)*T1b + T1d;
        T2 = T2c*inv((0+j*Wo)*eye(size(T2a)) - T2a)*T2b + T2d;
        temp = (1+j*Wo)/(1-j*Wo);
else
```

```
        T1 = T1d;
        T2 = T2d;
        temp = -1;
end

% subgradient for H2 norm is straight forward, just take deriv
% (phi_h2)^2 = q1^2 + q2^2 + ... + qN^2
% where N is n_tap_q*4, in this case most (all except 2) of
% the coeffs are in terms of the others. see notes for complete expl.
% then the subgradient for most (all except 2) is
% dphi_h2/dq1 = 4*r1, evaluated at the current guess
% while for the two indep coeffs the subgrad is just
% dphi_h2/dq1 = 2*r1, evaluated at the current guess
% the rest of them are similar, have to remember to multiply by
% weighting factor, and finally just add to subgrad for hinf

% calc h2 norm
phi_h2 = normest(qtemp);

indep = floor((n_tap-1)/4)+1;          % these are the indep q's
sg_h2 = [];
for k = 1:2:n_tap                      % step through entries
        if k == indep,                 % these are indep
                sg_h2 = [sg_h2; 2*noise*q(k); 2*noise*q(k+1)];
        else                           % these count double
                sg_h2 = [sg_h2; 4*noise*q(k); 4*noise*q(k+1)];
        end
end

phi_sg = [];                    % just need to give phi_sg size = 0
start = floor(n_tap/2);

% see notes for explanation of this mess!
for k = 0:floor(n_tap_o/4),
        factor = temp^(-k);
        if (n_tap_o-1)/(4*k) == 1,
                Q = [0      0 0 0
                     factor 0 0 factor];
                phi_sg = [phi_sg; real(uo*Q(1:2,1:2)*T2*vo)];
                phi_sg = [phi_sg; real(uo*Q(1:2,3:4)*T2*vo)];
                break                          % stop now
        end
```

```
        if floor(start/2) ~= k,
                factor = factor + temp^(-start+2+k);
        end
        Q = [factor 0 0 factor
             0      0 0 0];
        phi_sg = [phi_sg; real(uo*Q(1:2,1:2)*T2*vo)];
        phi_sg = [phi_sg; real(uo*Q(1:2,3:4)*T2*vo)];
        factor = temp^(-k) + temp^(-start+1+k);
        Q = [0      0 0 0
             factor 0 0 factor];
        phi_sg = [phi_sg; real(uo*Q(1:2,1:2)*T2*vo)];
        phi_sg = [phi_sg; real(uo*Q(1:2,3:4)*T2*vo)];
end

grad = phi_sg + sg_h2;

phi_hinf = out(1,1);
phi = phi_hinf + noise*(phi_h2)^2;      % fn value hinf + noise* h2^2
```

A.4 Subgradient: Cosine-Modulated Constraint

```
% calculates subgradient for cost function = hinf + noise*h2^2
% for cos modulated, BOTH even and odd, not linear phase
% [phi, phi_hinf, phi_h2, grad] = getsg(t1a, t1b, t1c, t1d,
%                          t2a, t2b, t2c, t2d, q, n_tap, odd, noise)
% the function to be minimized has the form:
% || H ||_inf
% || T1 + QT2 ||_inf  which can be broken down into
% || T1 + [q0 0; 0 0]T2 + [0 q1;0 0]T2 + ...
%       + z^-1[q5 0; 0 0]T2 + ...||
% the first step is to determine a subgradient at Ho, which is
% the transfer fn at some arbitrary initial guess of Q
% (which is 2 by 2)
% it turns out that the subgradient is
% g = [phi_sg(H1) phi_sg(H2) ...]'
% where H1 = T2, H2 = z^-1T2, ...

% April 29, 1995
```

```
function [phi, phi_hinf, phi_h2, grad] = ...
             getsg_cos(T1a, T1b, T1c, T1d, ...
             T2a,T2b,T2c,T2d,q,n_tap,odd,noise,f0_mod,f1_mod,f_tap);
% n_tap here means # of taps in q

% get into proper format
q0 = q .* f0_mod';
q1 = q .* f1_mod';

for k=1:n_tap-odd,
        qimp(k,1:4) = [q0(2*k) q1(2*k) q0(2*k-1) q1(2*k-1)];
end

if odd == 1,              % for odd filters need to add zeros
        qimp(n_tap,1:4) = [0 0 q0(2*n_tap-1) q1(2*n_tap-1)];
end

[qa, qb, qc, qd] = fir2ss(qimp, 2, 2);          % Q is 2 by 2 matrix

% ----------- 1) Determine Ho(jWo) ----------------
% bilinear trans  to continuous time

[Qa, Qb, Qc, Qd] = d2cm(qa, qb, qc, qd, 2, 'tustin');

% now multiply  and add to T1
[t2qa, t2qb, t2qc, t2qd] = series(T2a, T2b, T2c, T2d, Qa, Qb, Qc, Qd);

% calc Hinf norm
[ha, hb, hc, hd]=parallel(t2qa, t2qb, t2qc, t2qd, T1a, T1b, T1c, T1d);
system = pck(ha, hb, hc, hd);   % packed notation
out = hinfnorm(system);
Wo = out(1,3);
phi_hinf = out(1,1);

% evaluate stuff at this freq.
if isinf(Wo) == 0,
        HojWo = hc*inv((0+j*Wo)*eye(size(ha)) - ha)*hb + hd;
else
        HojWo = hd;
end
```

```
% get singular value decomposition of HojWo (p304 BOYD)
[U, S, V] = svd(HojWo);
uo = U(:,1);
uo = ((uo))'; % need in this format for formula
vo = V(:,1);


% -------------- 2) Now get T1(jWo), T2(jWo) ------------
if isinf(Wo) == 0,
        T1 = T1c*inv((0+j*Wo)*eye(size(T1a)) - T1a)*T1b + T1d;
        T2 = T2c*inv((0+j*Wo)*eye(size(T2a)) - T2a)*T2b + T2d;
        factor = (1-j*Wo)/(1+j*Wo);
else
        T2 = T2d;
        T1 = T1d;
        factor = -1;
end

% subgradient for H2 norm
% (phi_h2)^2 = q1^2 + q2^2 + ... + qN^2
%  = (f0_mod(1)^2 + f1_mod(1)^2)p^2(1)+(f0_mod(2)^2
%     + f1_mod(2)^2)p^2(2)...
% have to remember to multiply by
% weighting factor, and finally just add
% (fo_mod(k)^2+f1_mod(k)^2)*constant*q(k) to subgrad for hinf

% calculate h2 norm
phi_h2 = normest(q);
constant = noise*2;   % factor for H2 subgradient if use H2 norm sq


phi_sg = [];

% Now need to look at finite dim space
% so follow proc. on p307 BOYD
% H1 = [1 0; 0 0]*T2, ..., H5 = [1 0; 0 0]*z^-1*T2,....

for k = 1:n_tap,
  new_fact = factor^(k-1);
  Q = new_fact*[0 0; f0_mod(2*k-1) f1_mod(2*k-1)];
  phi_sg1  = real(uo*Q*T2*vo);
  phi_sgh2 = constant*((f0_mod(2*k-1))^2 + ...
```

```
                    (f1_mod(2*k-1))^2)*q(2*k-1);
  phi_sg = [phi_sg;
            phi_sg1+phi_sgh2];
   if k <= n_tap-odd,
      Q = new_fact*[f0_mod(2*k) f1_mod(2*k); 0 0];
      phi_sg1  = real(uo*Q*T2*vo);
      phi_sgh2 = constant*((f0_mod(2*k))^2 + ...
                           (f1_mod(2*k))^2)*q(2*k);
      phi_sg = [phi_sg;
                phi_sg1+phi_sgh2];
   end
end

% return fn value (inf norm + noise*h2norm)
phi = phi_hinf + noise*(phi_h2)^2;
grad =  phi_sg;
```

A.5 Subgradient: Alias-Free Constraint

A.5.1 Unconstrained

```
% alias free constraint general, except for linear phase
% AND synthesis filters must have SAME length

% for BOTH even and odd, and don't care about linear phase in synth
% filters
% [phi, phi_hinf, phi_h2, grad] = getsg(t1a, t1b, t1c, t1d,
%                          t2a, t2b, t2c, t2d, q, n_tap, odd, noise)
% the function to be minimized has the form:
% || H ||_inf
% || T1 + QT2 ||_inf  which can be broken down into
% || T1 + [q0 0; 0 0]T2 + [0 q1;0 0]T2 + ...
%       + z^-1[q5 0; 0 0]T2 + ...||
% the first step is to determine a subgradient at Ho, which is
% the transfer fn at some arbitrary initial guess of Q
% (which is 2 by 2)
% it turns out that the subgradient is
% g = [phi_sg(H1) phi_sg(H2) ...]'
% where H1 = T2, H2 = z^-1T2, ...
```

```
% March 6, 1995

function [phi, phi_hinf, phi_h2, grad] = ...
                getsg_noal(T1a, T1b, T1c, T1d, ...
                T2a, T2b, T2c, T2d, p, odd, noise, ...
                h0,h1,p_tap,f_tap,h_tap,q_tap);
f0 = conv(h1,p');
f1 = conv(h0,p');
q = [];
% need to figure out q (note that f0 and f1 have same # of taps)
for k = 1:2:f_tap-1,
        q = [q; f0(k+1); f1(k+1); f0(k); f1(k)];
end

%only go to second last tap,
%because may need to insert zeros for odd synth
for l = 1:q_tap-1;
        qimp(l,1:4) = [q(4*l-3,1) q(4*l-2,1) q(4*l-1,1) q(4*l,1)];
end

size_q = size(q,1);
if odd == 0,
        qimp(q_tap,1:4) = [q(size_q-3,1) q(size_q-2,1) ...
                           q(size_q-1,1) q(size_q,1)];
else
        qimp(q_tap,1:4) = [0 0 q(size_q-1,1) q(size_q,1)];
end

[qa, qb, qc, qd] = fir2ss(qimp, 2, 2);          % Q is 2 by 2 matrix

% ----------- 1) Determine Ho(jWo) ----------------
% bilinear trans  to continuous time

[Qa, Qb, Qc, Qd] = d2cm(qa, qb, qc, qd, 2, 'tustin');

% now multiply  and add to T1
[t2qa, t2qb, t2qc, t2qd] = series(T2a, T2b, T2c, T2d, Qa, Qb, Qc, Qd);

% calc Hinf norm
[ha, hb, hc, hd]=parallel(t2qa, t2qb, t2qc, t2qd, T1a, T1b, T1c, T1d);
system = pck(ha, hb, hc, hd);   % packed notation
```

```
out = hinfnorm(system);
Wo = out(1,3);
phi_hinf = out(1,1);

% evaluate stuff at this freq.
if isinf(Wo) == 0,
        HojWo = hc*inv((0+j*Wo)*eye(size(ha)) - ha)*hb + hd;
else
        HojWo = hd;
end


% get singular value decomposition of HojWo (p304 BOYD)
[U, S, V] = svd(HojWo);
uo = U(:,1);
uo = ((uo))'; % need in this format for formula
vo = V(:,1);


% -------------- 2) Now get T1(jWo), T2(jWo) ------------
if isinf(Wo) == 0,
        T1 = T1c*inv((0+j*Wo)*eye(size(T1a)) - T1a)*T1b + T1d;
        T2 = T2c*inv((0+j*Wo)*eye(size(T2a)) - T2a)*T2b + T2d;
        factor = (1-j*Wo)/(1+j*Wo);
else
        T2 = T2d;
        T1 = T1d;
        factor = -1;
end

% calculate h2 norm
phi_h2 = normest(q);

% calculate subgrad for h2 norm squared
% see notes for explanation
if h_tap < p_tap,
% pad with zeros at begin since imp resp, highest zero
        h0 = [h0 zeros(1,p_tap-h_tap)];
        h1 = [h1 zeros(1,p_tap-h_tap)];
        maxh = p_tap;
else
        maxh = h_tap;
```

```
end
for k = 1:p_tap,
        part(k) = 0;
        for l = 1:maxh-k+1,
                part(k) = part(k) + h1(l)*h1(k+l-1) + h0(l)*h0(k+l-1);
        end
end
part = 2*noise*part;
for k = 1:p_tap,
        sg_h2(k,1) = part(1)*p(k);
        for l = 2:p_tap-k+1,
                sg_h2(k,1) = sg_h2(k,1) + part(l)*p(k+l-1);
        end
        for l = 2:k,
                sg_h2(k,1) = sg_h2(k,1) + part(l)*p(k-l+1);
        end
end

% Now need to look at finite dim space
% so follow proc. on p307 BOYD
% calculate subgradient for hinf,  alias-free condition
h0_base1 = [];
h0_base2 = [];
h1_base1 = [];
h1_base2 = [];
for k = 2:2:h_tap, % remember descending order of z^(-1)
        h0_base1 = [h0(k-1) h0_base1];
        h0_base2 = [h0(k)   h0_base2];
        h1_base1 = [h1(k-1) h1_base1];
        h1_base2 = [h1(k)   h1_base2];
end
if k < h_tap, % this means the # of taps is odd,
              % so need to do last tap
        h0_base1 = [h0(k+1) h0_base1];
        h1_base1 = [h1(k+1) h1_base1];
end

% evaluate stuff at Wo,
% rest of entries are just delayed version of these
h0_base1 = polyval(h0_base1,factor);
h0_base2 = polyval(h0_base2,factor);
h1_base1 = polyval(h1_base1,factor);
```

```
h1_base2 = polyval(h1_base2,factor);

phi_sg = [];

% now get sg, note that in the case of odd p_tap will have only 4
% entries in Q for that case, hence the '-0.5'
for k = 0:p_tap/2-0.5,
     Q = [h1_base2*factor^k  h0_base2*factor^k
          h1_base1*factor^k  h0_base1*factor^k];
     phi_sg1  = real(uo*Q(1:2,1:2)*T2*vo);
     phi_sg = [phi_sg
               phi_sg1];
     if k <= p_tap/2-1 %(will skip if have last entry for odd p_tap)
               Q = [h1_base1*factor^k  h0_base1*factor^k
                    h1_base2*factor^(k+1)  h0_base2*factor^(k+1)];
               phi_sg1  = real(uo*Q(1:2,1:2)*T2*vo);
               phi_sg = [phi_sg
                         phi_sg1];
     end
end
phi_sg = phi_sg + sg_h2;        % add the subgradients for h2 and hinf

% return fn value (inf norm + noise*h2norm)
phi = phi_hinf + noise*phi_h2^2;
grad =  phi_sg;
```

A.5.2 Linear-Phase, Even Length

```
% calculates subgradient for cost function = hinf + noise*h2^2
% LINEAR PHASE, assuming N odd (ie n_tap even)
% [phi, phi_hinf, phi_h2, grad] = getsg(t1a, t1b, t1c, t1d,
%                       t2a, t2b, t2c, t2d, q, n_tap, odd, noise)
% the function to be minimized has the form:
% || H ||_inf
% || T1 + QT2 ||_inf  which can be broken down into
% || T1 + [q0 0; 0 0]T2 + [0 q1;0 0]T2 + ...
%       + z^-1[q5 0; 0 0]T2 + ...||
% the first step is to determine a subgradient at Ho, which is
% the transfer fn at some arbitrary initial guess of Q
% (which is 2 by 2)
```

```
% it turns out that the subgradient is
% g = [phi_sg(H1) phi_sg(H2) ...]'
% where H1 = T2, H2 = z^-1T2, ...

% November 28, 1994

function [phi,phi_hinf,phi_h2, grad] = ...
                getsg_lpevenal(T1a, T1b, T1c, T1d, ...
                T2a, T2b, T2c, T2d, q, n_tap,noise);
% NOTE n_tap refers to nubmer of taps in synth!

% need double the coeffs in q,
% but they are just functions of the other half
qtemp = q;
for l = 1:2:n_tap-1,
        qtemp = [qtemp; q(n_tap-l,1); -q(n_tap-l+1,1)];
end

% now ready to get into proper format
for l = 1:n_tap/2;              % divide by 2 since have half coeffs
                                % in Q since later these will double
        qimp(l,1:4) = [qtemp(4*l-3,1) qtemp(4*l-2,1)...
                       qtemp(4*l-1,1) qtemp(4*l,1)];
end

[qa, qb, qc, qd] = fir2ss(qimp, 2, 2);          % Q is 2 by 2 matrix

% ----------- 1) Determine Ho(jWo) ----------------
% bilinear trans  to continuous time

[Qa, Qb, Qc, Qd] = d2cm(qa, qb, qc, qd, 2, 'tustin');

% now multiply  and add to T1
[t2qa, t2qb, t2qc, t2qd] = series(T2a, T2b, T2c, T2d, Qa, Qb, Qc, Qd);

[ha, hb, hc, hd]=parallel(t2qa, t2qb, t2qc, t2qd, T1a, T1b, T1c, T1d);
system = pck(ha, hb, hc, hd);   % packed notation
out = hinfnorm(system);
Wo = out(1,3);

% evaluate stuff at this freq.
if isinf(Wo) == 0,
```

```
        HojWo = hc*inv((0+j*Wo)*eye(size(ha)) - ha)*hb + hd;
else
        HojWo = hd;
end


% get singular value decomposition of HojWo (p304 BOYD)
[U, S, V] = svd(HojWo);
uo = U(:,1);
uo = ((uo))';
vo = V(:,1);


% -------------- 2) Now get T1(jWo), T2(jWo) ------------
if isinf(Wo) == 0,
        T1 = T1c*inv((0+j*Wo)*eye(size(T1a)) - T1a)*T1b + T1d;
        T2 = T2c*inv((0+j*Wo)*eye(size(T2a)) - T2a)*T2b + T2d;
        factor = (1-j*Wo)/(1+j*Wo);
else
        T1 = T1d;
        T2 = T2d;
        factor = -1;
end

% subgradient for H2 norm is straight forward, just take deriv
% (phi_h2)^2 = q1^2 + q2^2 + ... + qN^2
% where N is n_tap_q*4, but in this case, half the coeffs are in
% terms of the other half so...
% (phi_h2)^2 = 2*q1^2 + 2*q2^2 + ... + 2*(q(N/2))^2
% then the subgradient is
% dphi_h2/dq1 = 4*r1, evaluated at the current guess
% the rest of them are similar, have to remember to multiply by
% weighting factor, and finally just add to subgrad for hinf

% calculate h2 norm
phi_h2 = normest(qtemp);        % have to do 'whole' q
sg_h2 = [];

for k = 1:n_tap,                % step through all unique q
        sg_h2 = [sg_h2; 4*noise*q(k)];
end
```

```
raised = floor(n_tap/2);          % NOTE n_tap means # taps in synth
phi_sg = [];                      % just need to give phi_sg size = 0

if floor(n_tap/4) ~= ceil(n_tap/4),
   n_tap_temp = n_tap + 2; % to get right # of iter in for loop
else
   n_tap_temp = n_tap;
end
% NOTE n_tap is # of taps in synth
for k = 0:floor(n_tap_temp/4)-1,
   Q(1:2,1:4) = [factor^k          0 0 factor^k
                 factor^(raised-k-1) 0 0 -(factor^(raised-k-1))];
   phi_sg = [phi_sg; real(uo*Q(1:2,1:2)*T2*vo)];
   phi_sg = [phi_sg; real(uo*Q(1:2,3:4)*T2*vo)];

    if 4*(k+1) <= n_tap,                % for 4, 8, 12, ... taps
      Q(1:2,1:4) = [factor^(raised-k-1) 0 0 -(factor^(raised-k-1))
                    factor^k            0 0 factor^k];
      phi_sg = [phi_sg; real(uo*Q(1:2,1:2)*T2*vo)];
      phi_sg = [phi_sg; real(uo*Q(1:2,3:4)*T2*vo)];
   end
end

grad = phi_sg + sg_h2;

% return fn value (here inf norm)
phi_hinf = out(1,1);
phi = phi_hinf + noise*(phi_h2)^2;
```

A.5.3 Linear-Phase, Odd Length

```
% alias free constraint general, LINEAR PHASE, ODD length
% AND synthesis filters must have SAME length
% [phi, phi_hinf, phi_h2, grad] = getsg(t1a, t1b, t1c, t1d,
%                          t2a, t2b, t2c, t2d, q, n_tap, odd, noise)
% the function to be minimized has the form:
% || H ||_inf
% || T1 + QT2 ||_inf  which can be broken down into
% || T1 + [q0 0; 0 0]T2 + [0 q1;0 0]T2 + ...
%       + z^-1[q5 0; 0 0]T2 + ...||
```

```
% the first step is to determine a subgradient at Ho, which is
% the transfer fn at some arbitrary initial guess of Q
% (which is 2 by 2)
% it turns out that the subgradient is
% g = [phi_sg(H1) phi_sg(H2) ...]'
% where H1 = T2, H2 = z^-1T2, ...

% March 6, 1995

function [phi, phi_hinf, phi_h2, grad] = ...
                getsg_lpoddal(T1a, T1b, T1c, T1d, ...
                T2a, T2b, T2c, T2d, p, noise, ...
                h0,h1,p_tap,f_tap,h_tap,q_tap);
f0 = conv(h1,p');
f1 = conv(h0,p');

% only need first p_tap-1,
% then rest is according to lin phase relationship
f0 = [f0(1,1:p_tap) fliplr(f0(1,1:p_tap-1)) 0];
f1 = [f1(1,1:p_tap) fliplr(f1(1,1:p_tap-1)) 0];

q = [];
% need to figure out q (note that f0 and f1 have same # of taps)
for k = 1:2:f_tap-1,
        q = [q; f0(k+1); f1(k+1); f0(k); f1(k)];
end

for l = 1:q_tap,
        qimp(l,1:4) = [q(4*l-3,1) q(4*l-2,1) q(4*l-1,1) q(4*l,1)];
end

[qa, qb, qc, qd] = fir2ss(qimp, 2, 2);          % Q is 2 by 2 matrix

% ----------- 1) Determine Ho(jWo) ----------------
% bilinear trans  to continuous time

[Qa, Qb, Qc, Qd] = d2cm(qa, qb, qc, qd, 2, 'tustin');

% now multiply  and add to T1
[t2qa, t2qb, t2qc, t2qd] = series(T2a, T2b, T2c, T2d, Qa, Qb, Qc, Qd);

% calc Hinf norm
```

```
[ha, hb, hc, hd]=parallel(t2qa, t2qb, t2qc, t2qd, T1a, T1b, T1c, T1d);
system = pck(ha, hb, hc, hd);   % packed notation
out = hinfnorm(system);
Wo = out(1,3);
phi_hinf = out(1,1);

% evaluate stuff at this freq.
if isinf(Wo) == 0,
        HojWo = hc*inv((0+j*Wo)*eye(size(ha)) - ha)*hb + hd;
else
        HojWo = hd;
end


% get singular value decomposition of HojWo (p304 BOYD)
[U, S, V] = svd(HojWo);
uo = U(:,1);
uo = ((uo))'; % need in this format for formula
vo = V(:,1);


% -------------- 2) Now get T1(jWo), T2(jWo) ------------
if isinf(Wo) == 0,
        T1 = T1c*inv((0+j*Wo)*eye(size(T1a)) - T1a)*T1b + T1d;
        T2 = T2c*inv((0+j*Wo)*eye(size(T2a)) - T2a)*T2b + T2d;
        factor = (1-j*Wo)/(1+j*Wo);
else
        T2 = T2d;
        T1 = T1d;
        factor = -1;
end

% calculate h2 norm
phi_h2 = normest(q);

% calculate subgrad for h2 norm squared
% see notes for explanation

% since by definition, h_tap < p_tap, must pad h's with zeros
h0 = [h0 zeros(1,p_tap-h_tap)]; % pad with zeros at begin since
h1 = [h1 zeros(1,p_tap-h_tap)]; % imp resp, highest zero
```

```
for k = 1:p_tap,   % step through all p_i's (see notes for details)
      sg_h2(k) = 0;
% do "centre" one
      l = 0;
      for l = 1:p_tap-k,
              sg_h2(k) = sg_h2(k) + 2*(h1(l)^2 + h0(l)^2)*p(k);
      end
      sg_h2(k) = sg_h2(k) + (h1(l+1)^2 + h0(l+1)^2)*p(k);
% do "below"
      index = 0;
      for l = k+1:p_tap,
         index = index+1;
         for m = 1:p_tap-k-index,
            sg_h2(k) = sg_h2(k) + 2*(h1(m)*h1(m+index) + ...
                       h0(m)*h0(m+index))*p(k+index);
         end
         sg_h2(k) = sg_h2(k) + ...
                    (h1(p_tap-k-index+1)*h1(p_tap-k+1) + ...
                    h0(p_tap-k-index+1)*h0(p_tap-k+1))*p(k+index);
      end
% do "above"
      index = 0;
      for l=1:k-1,
              index = index+1;
              for m = 1:p_tap-k,
                   sg_h2(k) = sg_h2(k) + 2*(h1(m)*h1(m+index) + ...
                              h0(m)*h0(m+index))*p(k-l);
              end
      sg_h2(k) = sg_h2(k) + (h1(p_tap-k+1)*h1(p_tap-k+1+index) + ...
                 h0(p_tap-k+1)*h0(p_tap-k+1+index))*p(k-l);
      end
end
sg_h2 = (2*noise*sg_h2)';              % take care of factors

% Now need to look at finite dim space
% so follow proc. on p307 BOYD
% calculate subgradient for hinf,  alias-free condition
% see notes for explanation

F0 = [];
F1 = [];
for k = 1:p_tap,
```

```
        % need temps for F0 and F1, since size won't initially match
        % with F's
        F0temp = [zeros(1,p_tap-k)];
        F1temp = [zeros(1,p_tap-k)];
        for l = 1:k,    % (may have padded h's with zeros above)
                F0temp = [F0temp h1(l)];
                F1temp = [F1temp h0(l)];
        end
        F0(k,:) = F0temp;
        F1(k,:) = F1temp;
end
F0 = [F0(1:size(F0,1),:)
      flipud(F0(1:size(F0,1)-1,:))
      zeros(1,p_tap)];
F1 = [F1(1:size(F1,1),:)
      flipud(F1(1:size(F1,1)-1,:))
      zeros(1,p_tap)];

for k = p_tap:-1:1,
      entry_11 = 0;
      entry_12 = 0;
      entry_21 = 0;
      entry_22 = 0;
      for l = 1:(size(F0,1))/2    % note that F0 and F1 have same size
              entry_11 = factor^(l-1)*F0(2*l,k) + entry_11;
              entry_12 = factor^(l-1)*F1(2*l,k) + entry_12;
              entry_21 = factor^(l-1)*F0(2*l-1,k) + entry_21;
              entry_22 = factor^(l-1)*F1(2*l-1,k) + entry_22;
      end
      phi_sg1 = real(uo*[entry_11 entry_12; entry_21 entry_22]*T2*vo);
      phi_sg = [phi_sg
                phi_sg1];
end
phi_sg = phi_sg + sg_h2;        % add the subgradients for h2 and hinf

% return fn value (inf norm + noise*h2norm)
phi = phi_hinf + noise*phi_h2^2;
grad =  phi_sg;
```

REFERENCES

[1] C. Esteban and C. Galand. Application of quadrature mirror filters to split band voice coding shemes. In *IEEE International Conference on ASSP*, pages 191–195, Hartford, May 1977.

[2] R. E. Crochiere, R. V Cox, and J. D. Johnston. Real-time speech coding. *IEEE Transactions on Communications*, COM-30(4):621–634, April 1982.

[3] J. W. Woods and S. D. O'Neil. Subband coding of images. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-34(5):1278–1288, October 1986.

[4] P. P. Vaidyanathan. Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial. *Proceedings of the IEEE*, 78(1):56–93, January 1990.

[5] M. Vetterli. A theory of multirate filter banks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(3):356–372, March 1987.

[6] P. P. Vaidyanathan. Quadrature mirror filter banks, m-band extensions and perfect reconstruction techniques. *IEEE ASSP Magazine*, pages 4–20, July 1987.

[7] P. P. Vaidyanathan. *Multirate Systems.* Prentice-Hall, Englewood Cliffs, 1993.

[8] T. Q. Nguyen. Near-perfect reconstruction pseudo-QMF banks. *IEEE Transactions on Signal Processing*, 42(1):65–75, January 1994.

[9] J. D. Johnston. A filter family designed for use in quadrature mirror filter banks. In *ICASSP*, pages 291–294, Denver, Colorado, April 1980.

[10] E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou. Design of low-delay FIR QMF banks using the Lagrange-multiplier approach. *Electronic Letters*, 30(12):924–926, June 1994.

[11] D. R. Koilpillai and P. P. Vaidyanathan. A spectral factorization approach to pseudo-QMF design. *IEEE Transactions on Signal Processing*, 41(1):82–92, January 1993.

[12] K. Nayebi, T. P. Barnwell, III, and M. J. T. Smith. Low delay FIR filter banks: Design and evaluation. *IEEE Transactions on Signal Processing*, 42(1):24–31, January 1994.

[13] V. K. Jain and R. Crochiere. Quadrature mirror filter design in the time domain. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-32(2):353–361, April 1984.

[14] B. Horng and A. N. Wilson, Jr. Lagrange multiplier approaches to the design of two-channel perfect-reconstruction linear-phase FIR filter banks. *IEEE Transactions on Signal Processing*, 40(2):364–374, February 1992.

[15] K. Nayebi, T. P. Barnwell, III, and M. J. T. Smith. Time-domain filter bank analysis: A new design theory. *IEEE Transactions on Signal Processing*, 40(6):1412–1429, June 1992.

[16] T. Chen and B. A. Francis. Design of multirate filter banks by $\mathcal{H}_\infty$ optimization. to appear in IEEE Transactions on Signal Processing, 1995.

[17] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing.* Prentice-Hall, 1989.

[18] M. Bellanger, G. Bonnerot, and M. Coudreuse. Digital filtering by polyphase network: application to sample rate alteration and filter banks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-24:109–114, April 1976.

[19] R. E. Crochiere and L. R. Rabiner. *Multirate Digital Signal Processing.* Prentice-Hall, Englewood Cliffs, 1983.

[20] K. Nayebi, T. P. Barnwell, III, and M. J. T. Smith. On the design of uniformly and nonuniformly spaced pseudoquadrature mirror filters. *IEEE Transactions on Signal Processing*, 42(4):825–833, April 1994.

[21] H. Nussbaumer. Pseudo QMF filter bank. *IBM Technical Disclosure Bulletin*, 24(6):3081–3087, November 1981.

[22] H. Nussbaumer and M. Vetterli. Pseudo quadrature mirror filters. In *Proc. Int. Conference on Digital Signal Processing*, pages 8–12, Florence, Italy, September 1984.

[23] R. V. Cox. The design of uniformly and nonuniformly spaced pseudo-quadrature mirror filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-34(5):1090–1096, October 1986.

[24] V. P. Sathe and P. P. Vaidyanathan. Effects of multirate systems on the statistical properties of random signals. *IEEE Transactions on Signal Processing*, 41(1):131–146, January 1993.

[25] S. P. Boyd and C. H. Barrett. *Linear Controller Design: Limits of Performance.* Prentice-Hall, 1991.

[26] H. Shu, T. Chen, and B. A. Francis. Uniformly optimal design of hybrid multirate filter banks. preprint, 1995.

[27] M. Vetterli. Perfect transmultiplexers. In *Proc. IEEE International Conference on ASSP*, pages 2567–2570, Tokyo, Japan, April 1986.

[28] T. Chen and B. A. Francis. *Optimal Sampled-Data Control Systems.* Springer-Verlag, London, 1995.

[29] C. Barratt, N. Khraishi, and S. Norman. Unix QDES manual entry. 1987.

[30] S. Boyd, V. Balakrishnan, C. Barratt, N. Khraishi, X. Li, D. Meyer, and S. Norman. A new CAD method associated architectures for linear controllers. *IEEE Transactions on Automatic Control*, AC-33(3):268–283, March 1988.

[31] The Mathworks Inc. Matlab. Version 4.2a, 1994.

[32] The Mathworks Inc. Mu-analysis and synthesis toolbox. Version 2.0, 1993.

[33] The Mathworks Inc. Control system toolbox. Version 3.0b, 1993.