Applications of Lattice Filters to Quadrature Mirror Filter Banks

1988

Gregory R. Jaspers
University of Central Florida

Find similar works at: https://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

Part of the Systems and Communications Commons

STARS Citation

https://stars.library.ucf.edu/rtd/4292

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
APPLICATION OF LATTICE FILTERS TO QUADRATURE MIRROR FILTER BANKS

BY

GREGORY R. JASPERS
B.S., University of Cincinnati, 1980

RESEARCH REPORT

Submitted in partial fulfillment of the requirements for the degree of Master of Science in the Graduate Studies Program of the College of Engineering University of Central Florida Orlando, Florida

Summer Term 1988
ABSTRACT

Presented is a method for designing and implementing lattice filters to be used in Quadrature Mirror Filter (QMF) Banks. Quadrature Mirror Filter Banks find use in applications where a signal must be split into subbands operated on then reconstructed in the output. Because of their structure, lattice filters do this very well and allow perfect reconstruction, even when the lattice coefficients must be quantized.

In this paper QMF’s and Lattice Filters are derived and analyzed. Application of the lattice filter is presented along with a design program and example of its use to implement a QMF.

The computer aided design procedure allows the user to input the stop-band frequency, normalized to the sampling frequency, and the desired attenuation. The resulting outputs are the lattice coefficients, and the Finite Impulse Response (FIR) coefficients of an FIR filter having the same characteristics. The program selects a set of coefficients based on optimal coefficients that are within the desired tolerance.

The filter design program was written in FORTRAN, with the filter coefficients stored in a data file on disk. Programs were written in MATHCAD© to show the lattice filter response and to simulate the QMF using these coefficients.
## TABLE OF CONTENTS

**LIST OF FIGURES** ......................................................................................................................... iv  

**Chapter**  
I. **INTRODUCTION** ........................................................................................................................  1  
II. **QUADRATURE MIRROR FILTERS** ............................................................................................. 3  
   Decimators .................................................................................................................................... 5  
   Interpolators ............................................................................................................................... 9  
   Effects of Decimation and Interpolation ..................................................................................... 11  
   Analysis Filters ........................................................................................................................... 16  
   Synthesis Filters ......................................................................................................................... 17  
III. **LATTICE FILTERS** .................................................................................................................. 24  
    Theory Development ................................................................................................................. 24  
    Relation to Lattice Filters Used in Linear Prediction ............................................................... 32  
IV. **LATTICE FILTER APPLICATION TO QUADRATURE MIRROR FILTERS** ......................... 35  
    Lattice Structure ...................................................................................................................... 35  
    Computer Aided Design Application ....................................................................................... 37  
V. **RESULTS AND DISCUSSION** ................................................................................................. 39  
    Filter Design Example ............................................................................................................. 39  
    Example of Lattice Filter Applied to a Quadrature Mirror Filter Bank ............................... 40  
    Flowchart .................................................................................................................................. 52  
    Discussion ................................................................................................................................. 53  

**Appendices**  
A. **FILTER DESIGN PROGRAM** ............................................................................................... 56  
B. **FILTER PROGRAM RUN** ..................................................................................................... 66  
C. **LEVINSON-DURBIN ALGORITHM** ....................................................................................... 68  
D. **MATHCAD® PROGRAMS** ..................................................................................................... 70  

**REFERENCES** ............................................................................................................................... 80
**LIST OF FIGURES**

1. Quadrature Mirror Filter Block Diagram ................................................................. 4
2. Diagram of Aliasing Due to Decimation ...................................................................... 8
3. Decimation of a High Pass Type Signal ..................................................................... 10
4. Interpolation of Signal $X(e^{j\omega})$ ................................................................... 12
5. Decimation and Interpolation With Two Different Inputs ........................................ 12
6. Decimation of the Signals in Figure 5 ...................................................................... 14
7. Interpolation of the Signals in Figure 6 .................................................................. 14
8a. Two Channel Signal Flow Graph ............................................................................. 18
8b. Two Channel Decimator/Interpolator Diagram ...................................................... 18
8c. Two Channel Flow Graph with Transfer Functions ............................................... 19
8d. Two Channel Filter................................................................................................... 20
9. Transfer Function Movement with Decimation and Interpolation ............................. 22
10. Lattice Building Block ............................................................................................ 26
11. Lattice Structure ...................................................................................................... 29
12. Denormalized Lattice Structure .............................................................................. 32
13. Analysis Bank Structure ........................................................................................ 36
14. Lattice Structure for QMF ...................................................................................... 37
15. Lattice Filter Blocks Frequency Response ............................................................. 41
16. Lattice Filter $p_{N-1}(z)$ Frequency Response ......................................................... 42
17. Lattice Filter $q_{N-1}(z)$ Frequency Response ........................................................ 43
18. Analysis Filters Output to Unit Sample Input ......................................................... 46
19. QMF Output to Unit Sample Input ........................................................................... 47
20. Analysis Filters Output to Low Frequency Sinewave Input................................................. 48
21. QMF Output to Low Frequency Input.............................................................................. 49
22. Analysis Filters Output to High Frequency Sinewave Input............................................ 50
23. QMF Output to High Frequency Sinewave Input.............................................................. 51
CHAPTER I
INTRODUCTION

The purpose of my research was to investigate the lattice filter and how it could be used in the implementation of the Quadrature Mirror Filter (QMF) Bank. QMF banks find use in applications requiring a signal be split into several frequency bands and undersampled before coding and transmission, and then suitably recombined at the output. A number of versions of such QMF banks are known, some based on Infinite Impulse Response (IIR) filters and some on Finite Impulse Response (FIR) filters. IIR filters as their name implies are recursive in nature and hence can be easily implemented digitally. However, IIR filters can exhibit instability and are more coefficient dependent in QMF applications. FIR filters on the other hand have guaranteed stability for non-recursive implementation, linear phase, and real impulse response coefficients. Because of this they tend to be used over IIR filters in QMF designs.

Based on this information, I first studied the QMF filter to see what was needed as far as structure and filter characteristics. The QMF is made up of four basic parts, analysis filters, synthesis filters, decimators, and Interpolators. In doing the research I learned that the lattice filter would fit the basic requirements and offer some advantages as compared to other FIR realizations, when used in the realization of the analysis and synthesis filters. Some of these advantages are:

- All of the analysis or synthesis filters are computed at once.
- Each filter is interrelated to the other, so that quantization effects are balanced out.
- The decimator or interpolator is included as part of the structure.
If more attenuation is needed, add more building blocks.

The lattice filters are analyzed by comparing them to the lattice structures used in linear prediction. Using this information and the constraints for perfect reconstruction in the QMF a structure is derived and analyzed. This computer program was then written which finds the lattice coefficients based on input tolerances and previously computed optimal coefficients fitting those of the desired tolerances. The computer program computes not only the lattice coefficients, but also the impulse coefficients of an FIR filter having the same characteristics. Magnitude plots of some example filters are given.

An example is provided of a simulated QMF with the appropriate analysis and synthesis designed using lattice filter structures. These coefficients of these lattice structures were computed using the previously mentioned computer program, and stored on floppy disk. To do the simulation MATHCAD® programs were written to simulate a QMF using lattice filters that have the coefficients stored on disk.

In general, by using lattice filters, the perfect reconstruction property of QMF's can be maintained. This means the output is a replica of the input, but the input signal had some type of operations performed on it.
CHAPTER II
QUADRATURE MIRROR FILTER BANKS

This chapter will look at Quadrature Mirror Filter Banks. Because of the many possibilities available, I will primarily look at the two channel case. The general theory will be presented, at times showing the relation to more than two channels, as part of development of the theory. In general, Quadrature Mirror Filter (QMF) banks find application where a discrete time signal is to be split into a number of consecutive bands in the frequency domain, so that each subband may be processed independently. The subbands are restored at the receiver to form a properly reconstructed signal that resembles the original signal. Shown in Figure 1 is a diagram for an M channel Quadrature Mirror Filter Bank. The input signal is sent to the analysis bank where the M filters $H_0(z), H_1(z), ..., H_M(z)$ split the signal into M frequency bands. These subband signals $X_k(n)$ are decimated by the factor M, then have some operation (encoding, transmission, etc.) done to them. At the received end another operation is performed, the subband signal is then interpolated by M, and recombined through the set of M synthesis filters $F_0(z), F_1(z), ..., F_M(z)$ [1]. I would like to analyze the QMF by first looking at Figure 1 and analyze each section. The decimators and interpolators will be examined first to see how they operate, then using this information, determine what is needed for the analysis and synthesis filters.

Figure 1 major components:

1. Decimator
2. Interpolator
3. Analysis Filter $H_k(z)$
4. Synthesis Filter $F_k(z)$
Figure 1. Quadrature Mirror Filter Block Diagram.
Decimators

An M fold decimator compresses the input by an integer number M times. It does this by retaining only those samples of \( x(n) \) which occur at times that are integer multiples of M. This means the output \( y(n) \) is related to the input \( x(n) \) by \( y(n) = x(Mn) \). This relationship is shown below for \( M=3 \), and \( x(n)=0 \) for \( n<0 \).

| \( n \) | 0 1 2 3 4 5 6 7 8 9 |
|---|---|---|---|---|---|---|---|---|---|
| \( x(n) \) | 5 7 1 0 2 4 9 -5 13 |
| \( y(n) \) | 5 0 9 7 .......... |

As another example consider \( x(n) \) shifted in time, using the same decimation as before with \( x'(n) = x(n-1) \).

| \( x'(n) \) | 0 5 7 1 0 2 4 9 -5 13 |
|---|---|---|---|---|---|---|---|---|---|
| \( y'(n) \) | 0 1 4 13 .......... |

\( y'(n) = x'(Mn) \) 

The above results show decimation is linear and time varying. To see this notice \( y'(n) = x'(Mn) \), where \( x'(n) = x(n-1) \). If the operation were time-invariant then \( y'(n) \) would equal \( x(M(n-1)) = x(Mn-M) \). The example shows that \( y'(n) = x'(n) = x(Mn-1) \) which is time-varying. Because of this, it cannot be represented by a transfer function. Additionally, since decimation is a compression in the time domain, then duality would suggest stretching in the frequency domain [2]. Because the decimator is time-invariant, the transform must be broken into parts for analysis. Looking at the data above, \( y(n) \) takes samples of \( x(n) \) at every \( M \) points, which gives a Z transform of:

\[
Y(z) = \sum_{n=-\infty}^{\infty} x(Mn) z^{-n}
\]
The problem with this value of $Y(z)$ is the fact that $x(n)$ is sampled at every $M$ point. The argument for $z$ must also reflect this fact, and can be taken care of by considering that $z$ is also sampled, similar to sampling of the $Z$ transform at every $M$th point on the unit circle. However, that means there will be a separate $Z$ transform for each $M$ point of $z$. Thus for an $M$ fold decimator the input-output relationship in the transform domain is [3]:

$$y(n) = x(Mn)$$

$$y(n) = \frac{1}{M} \sum_{k=0}^{M-1} x(Mn) e^{j \frac{2\pi kn}{M}}$$

$$Y(z) = \sum_{n=-\infty}^{\infty} y(n) z^{-n}$$

$$Y(z) = \sum_{n=-\infty}^{\infty} \frac{1}{M} \sum_{k=0}^{M-1} x(Mn) e^{j \frac{2\pi kn}{M}} z^{-n}$$

$$Y(z) = \frac{1}{M} \sum_{k=0}^{M-1} \sum_{n=-\infty}^{\infty} x(n) e^{j \frac{2\pi kn}{M}} z^{-\frac{n}{M}}$$ (since $y(n)=0$ except at integer multiples of $M$)
\[
Y(z) = \frac{1}{M} \sum_{k=0}^{M-1} \sum_{n=-\infty}^{\infty} x(n) e^{-j2\pi k/M} e^{-j2\pi nM/z} 
\]

or

\[
Y(z) = \frac{1}{M} \sum_{k=0}^{M-1} X(z^{1/M} w^k) \tag{1}
\]

where \(W = W_M = e^{-j2\pi/M}\) and \(z = e^{jw}\)

When \(M=2\):

\[
Y(e^{jw}) = 0.5 \left[ X(e^{jw/2}) + X(e^{j(w-2\pi)/2}) \right] 
\]

\[
= 0.5 \left[ X(e^{jw/2}) + X(e^{jw/2} (e^{-j\pi}) \right] 
\]

\[
= 0.5 \left[ X(e^{jw/2}) + X(-e^{jw/2}) \right] \tag{3}
\]
In Figure 2, a low pass type of signal, $X(e^{jw})$, is shown along with the two components of $Y(e^{jw})$ in equation (3). From Figure 2, $X(-e^{jw}/2)$ is the same as $X(e^{jw}/2)$ shifted $-\pi$, and is shown as a shifted signal. Unless $X(e^{jw})$ is band limited to $-\pi/2 \leq w \leq \pi/2$ the two components of $Y(e^{jw})$ will alias with each other. This aliasing is shown as the overlap region in Figure 2.

One problem with decimation, when the input is not band limited, has to due with a highpass type of signal input into the decimator. The transform of a highpass type signal (band limited $\pi/2 \leq w \leq 3\pi/2$) is shown in Figure 3(a). If this signal is used as the transformed input to a two fold ($M=2$) decimator, the result will be as shown in Figure 3(b). It also turns out that a non-band limited signal of the form in Figure 3(c) will also yield the result given in Figure 3(b).
3(b). For the highpass case, the components of $Y(e^{jw})$ do not overlap, so the signal can be recovered if it is known that the signal was a highpass signal originally [4]. Therefore, in order to use a decimator, the input signal should first be bandlimited, by a bandpass filter, to reduce the effects of aliasing and allow reconstruction of the output signal. If this is done correctly, the output will resemble the original input signal.

**Interpolators**

An interpolator does the opposite function that a decimator does. It stretches in the time domain which causes a compression in the frequency domain. The output $y(n)$ is related to the input $x(n)$ by $y(n) = x(n/M)$ where $n$ is a multiple of $M$ and $y(n) = 0$ otherwise. This relationship is shown below for $M=3$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x(n)$</td>
<td>5</td>
<td>7</td>
<td>1</td>
<td>10</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>-5</td>
<td>13</td>
<td>7</td>
<td>-3</td>
</tr>
<tr>
<td>$y(n)$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

As another example consider $x(n)$ shifted in time, using the same interpolation as before with $x'(n) = x(n-1)$.

<table>
<thead>
<tr>
<th>$x'(n)$</th>
<th>...</th>
<th>5</th>
<th>7</th>
<th>1</th>
<th>10</th>
<th>2</th>
<th>4</th>
<th>9</th>
<th>-5</th>
<th>13</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y'(n)$</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Figure 3. Decimation of a High Pass Type Signal.
The transfer domain relationship for the interpolator is simpler than the decimator. For the interpolator analysis, let \( a = \frac{n}{M} \) then \( n = aM \) and:

\[
Y(z) = \sum_{n=\infty}^{\infty} x(n/M) z^{-n}
\]

\[
Y(z) = \sum_{n=\infty}^{\infty} x(a) z^{-aM}
\]

\[
Y(z) = X(z^M)
\]

\[
Y(z) \big|_{z=e^{jw}} = Y(e^{jw}) = X(z^M) \big|_{z=e^{jw}} = X(e^{jwM}) \tag{4}
\]

when \( M=2; Y(z) = X(z^2) \) and \( Y(e^{jw}) = X(e^{j2w}) \). Figure 4b shows the effects of interpolation on the signal \( X(e^{jw}) \) in Figure 4a for \( M=2 \). Interpolation causes \( M-1 \) images to appear of the original spectrum. When using an interpolator, a filter is desired at the output, usually a digital bandpass filter that will reject the images.

**Effects of Decimation and Interpolation**

As a continuation of decimation and interpolation, I will take a look at a bandlimited signal that is decimated by \( M=2 \) then interpolated by \( M=2 \). Two signals will be considered, the first where \( X(e^{jw}) \) is limited to \(-\pi/M \leq w \leq \pi/M\), and another where the signal is not limited to \(|w| \leq \pi/M\). The decimation and interpolation operation is shown in Figure 5. Figures 5a and 5b show two different input signals that will be analyzed for their own individual characteristics.
Figure 4. Interpolation of Signal $X(e^{jw})$.

Figure 5. Decimation and Interpolation with Two Different Inputs.
Additionally, Table 1 gives a list of values for different inputs w of the signals shown in Figure 5.

**TABLE 1**

<table>
<thead>
<tr>
<th>w</th>
<th>$X(e^{jw})$</th>
<th>$X_1(e^{jw/2})$</th>
<th>$X_1(-e^{jw/2})$</th>
<th>$X(e^{jw})$</th>
<th>$X_1(e^{jw/2})$</th>
<th>$X_1(-e^{jw/2})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-5\pi/4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$-\pi$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0</td>
</tr>
<tr>
<td>$-3\pi/4$</td>
<td>0</td>
<td>0.25</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>0</td>
</tr>
<tr>
<td>$-\pi/2$</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0.2</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>$-\pi/4$</td>
<td>0.5</td>
<td>0.75</td>
<td>0</td>
<td>0.6</td>
<td>0.8</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>$\pi/4$</td>
<td>0.5</td>
<td>0.75</td>
<td>0.25</td>
<td>0.6</td>
<td>0.8</td>
<td>0.4</td>
</tr>
<tr>
<td>$\pi/2$</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0.2</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>$3\pi/2$</td>
<td>0</td>
<td>0.25</td>
<td>0.75</td>
<td>0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
<tr>
<td>$\pi$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>$5\pi/4$</td>
<td>0</td>
<td>0</td>
<td>0.75</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
</tr>
<tr>
<td>$3\pi/2$</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
</tr>
<tr>
<td>$7\pi/4$</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>$2\pi$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Using Figure 5 and the data in Table 1 the outputs of the decimators and interpolators are shown. Figure 6a is a plot of the data in Table 1 corresponding to Figure 5a and shows the aliasing due to decimation. Figure 6b is a similar plot of the data for Figure 5b. The decimator output is sent to the interpolator which causes compression in the frequency domain. The interpolator output, with $M=2$, is shown in Figure 7a for the input signal from Figure 5a, and in Figure 7b for the input from Figure 5b.
Figure 6. Decimation of the Signals in Figure 5.

Figure 7. Interpolation of the Signals in Figure 6.
Using Figures 5 through 7 and the data in Table 1, the output $Y(e^{jw})$ is shown to be:

$$Y(e^{jw}) = X_1(e^{j2w}) \quad X_1(e^{jw}) = 0.5[X(e^{jw/2}) + X(-e^{jw/2})]$$

$$X_1(e^{jw}) = 0.5[X(e^{jw/2}) + X(e^{j(w/2-\pi)})]$$

$$Y(e^{jw}) = X_1(e^{j2w}) = 0.5[X(e^{jw}) + X(e^{j(w-\pi)})]$$ \hfill (5)

Figure 7 shows that signals bandlimited to $-\pi/M < w \leq \pi/M$ can be reconstructed in their original form, while signals that are not will be distorted in the reconstruction unless this aliasing is taken care of in the reconstruction process.

In order to determine the analysis and synthesis filters, look at Figure 1 in the transform domain for the case when $M=2$.

Analysis filter output: $X_{1k}(z) = X(z)H_k(z)$ \hfill (6)

Decimator output: $X_{2k}(z) = 1/2[ X_{1k}(z^{1/2}) + X_{1k}(-z^{1/2})]$

$$= 1/2[ X(z^{1/2})H_k(z^{1/2}) + X(-z^{1/2})H_k(-z^{1/2})]$$ \hfill (7)

Note: In general $X_{2k}(z) = \frac{1}{M} \sum_{n=0}^{M-1} X_{1}(z^{1/M^W_0})H_k(z^{1/M^W_0})$ \hfill (8)

Interpolator output: $X_{3k}(z) = X_{2k}(z^2) = \frac{1}{2} [ X(z)H_k(z) + X(-z)H_k(-z)]$ \hfill (9)

Synthesis Filter output: $X_{4k}(z) = X_{3k}(z)F_k(z) = \frac{1}{2} F_k(z)[X(z)H_k(z) + X(-z)H_k(-z)]$ \hfill (10)

QMF output: $X^*(z) = X_{40}(z) + X_{41}(z) = X_{30}(z)F_0(z) + X_{31}(z)F_1(z)$ \hfill (11)

$$X^*(z) = 1/2[H_0(z)F_0(z) + H_1(z)F_1(z)]X(z) + 1/2[H_0(-z)F_0(z) + H_1(-z)F_1(z)]X(-z)$$ \hfill (12)
In general, \( \hat{X}(z) = \frac{1}{M} \sum_{i=0}^{M-1} X(zW^{-1}) \sum_{i=0}^{M-1} H_k(zW^{-1})F_k(z) \) \( \{13\} \)

Because the output involves both \( X(z) \) and \( X(-z) \), a transfer function cannot be written down in closed form \( \{3\} \).

**Analysis Filters**

The purpose of the analysis filter is to separate the input signal into \( M \) frequency bands. To do this, there should be no overlap between the responses of each bandpass filter. However, each subband is against the adjacent subband which means the filters should have a "brickwall" or ideal response. In practice this requires a filter with a very large number of poles. If this is not possible, the filter response will cause overlapping of the subbands. The QMF allows the analysis filters to be less than ideal by permitting aliasing in the decimators, then correcting for this in the interpolator and synthesis filter \( \{2\} \). The analysis filter will be defined as \( H(z) \). Depending on the number of subbands needed, the input spectrum is divided into \( M \) subbands through each analysis filter \( H_k(z) \). Looking at \( \{6\} \) and \( \{8\} \) define \( H_k(z) \) as:

\[
H_k(z) = H(zW_M^k)
\]

For \( k=0 \), \( H_0(z) = H(z) \) which is the basic filter. Each filter \( H_k(z) \) is then a spectral factor of \( H(z) \). Additionally, this also serves as the decimation filter. When \( M=2 \) this reduces to:

\[
H_1(z) = H(zW_2^{-1}) = H(-z) \quad \{14\}
\]

\[
H_1(z) = H_0(-z)
\]
Synthesis Filters

The purpose of the synthesis filters are to cancel aliasing due to the analysis filters, decimators, and interpolators. Looking at equation (13), if the $W^{-1}$, $i 
eq 0$ equations are made to go to zero, then aliasing and imaging are cancelled and we can define $\frac{\hat{x}(z)}{x(z)} = T(z)$, with $T(z)$ called the distortion transfer function. Therefore, the relation for $T(z)$ yields (in Matrix form):

$$
\begin{bmatrix}
H_0(z) & H_1(z) & \ldots & H_{M-1}(z) \\
H_0(zW^{-1}) & H_1(zW^{-1}) & \ldots & H_{M-1}(zW^{-1}) \\
\vdots & \vdots & \ddots & \vdots \\
H_0(zW^{-M+1}) & H_1(zW^{-M+1}) & \ldots & H_{M-1}(zW^{-M+1})
\end{bmatrix}
\begin{bmatrix}
F_0(z) \\
F_1(z) \\
\vdots \\
F_{M-1}(z)
\end{bmatrix}
= 
\begin{bmatrix}
T(z) \\
0 \\
0 \\
\vdots \\
0
\end{bmatrix}
$$

For the system, if $T(z)$ is stable and all pass, then the system will not have amplitude distortion. Additionally, if the synthesis filters are chosen properly, aliasing can be eliminated. Looking at (12), if $F(z)$ is chosen to eliminate the aliasing terms ($X(-z)$ and $H(-z)$, which are $X(zW^n)$ and $H_k(zW^n)$ in (13), then the output can be written in terms of the input as:

$$
\frac{\hat{X}(z)}{X(z)} = T(z) = 1/2 [H_0(z)F_0(z) - H_1(z)F_1(z)] = 1/2 [H_0(z^2) - H_0(-z)]
$$

where $F_0(z) = H_0(z)$, $F_1(z) = -H_1(z)$

Additionally, let $H_1(z) = H_0(-z)$ then, from (14):

$$
T(z) = 1/2 [H_0^2(z) - H_0^2(-z)]
$$

$T(z)$ represents the overall transfer function or the distortion function [1]. The objective is to choose $H(z)$ and $F(z)$ such that $T(z)$ has minimum distortion. Because $T(z)$ is complex, it has amplitude and phase components. Depending on the application, one of these terms can be minimized without regard to the other term, but normally both terms need to have minimum
distortion. If aliasing, amplitude distortion, and phase distortion are eliminated, then there is perfect reconstruction and $X^\sim(n) = X(n)$. One solution is to use FIR filters since they have linear phase. If the Analysis and Synthesis filters are chosen properly, phase distortion, amplitude distortion, and aliasing can be eliminated using FIR filters [2].

In order to analyze the reconstruction problem, the QMF will be looked at in a different way. Figure 8a is a signal flow graph of a basic QMF structure using delays. Using signal flow graph notation [7] and Figure 8a, we see:

$$w_1 = x(z) \quad w_2 = w_1 \quad w_3 = z^{-1}w_1 \quad w_4 = Y(z) = w_3 + z^{-1}w_2$$ \hspace{1cm} (17)

$$w_4 = z^{-1}w_1 + z^{-1}w_1 = 2z^{-1}w_1$$

![Figure 8a. Two Channel Signal Flow Graph.](image)

Figure 8a shows the output $Y(z) = 2z^{-1}X(z)$, this implies then that $x^\sim(n) = 2x(n-1)$

![Figure 8b. Two Channel Decimator/Interpolator Diagram.](image)
Figure 8b shows a signal flow graph with the decimators and interpolators inserted between nodes w1-w2 and w3-w4. With the decimators and interpolators in line, using (1) and (4) for M=2 gives:

\[ w_2 = \frac{1}{2} [ X(e^{jw/2}) + X(e^{j((w-2\pi)/2)}) ] \]

Then \( w_2 = X(e^{jw}) = X(z) \). Using this result yields \( w_4 = w_3 + z^{-1}w_2 \), and \( X(z) = w_4 = z^{-1}w_2 + w_3 = z^{-1}X(z) + w_3 = z^{-1}X(z) + w_3 \)

\[ X^*(z) = 2z^{-1}X(z) \] \( (18) \)

\[ x^*(n) = y(n) = 2 x(n-1) \] \( (19) \)

![Flow Graph Diagram](image)

**Figure 8c. Two Channel Flow Graph with Transfer Functions.**

Now insert a transfer function as shown in Figure 8c. If \( S(z) = 1 \) then \( X^*(z) \) will be the same as (18). To look at this further, \( S(z) \) can be split into two transfer functions \( G(z) \) and \( R(z) \) where \( S(z) = G(z)R(z) \). This is shown in Figure 8d. If \( G(z) = R(z) = 1 \), then \( X^*(z) \) will be the same as (18).
Before proceeding further, I want to show how a transfer function can be moved within the diagram. For the following analysis use the basic diagram of Figure 8a, with the appropriate bank added or modified as described in the discussion. This means that the delays are not shown in Figure 9 but are assumed in the Figure and in the discussion. I will wrap it up by showing the case for $M=2$ with all terms included in (24). Let $w_1a$ be the output of the decimator in Figure 8d with $w_1$ as the input, and $w_1b$ as the output of $G(z)$. Then, $w_1a$ will be described as in (1). If this is applied as the input to $G(z)$, the output $w_1b$ will be:

$$w_{1b}(z) = \frac{1}{M} \sum_{i=0}^{M-1} X(z^{1/M})^i G(z)$$

Looking at this transformation there is a $z^{1/M}$ transformation of the input to the decimator. If the decimator and $G(z)$ are interchanged, $w_1a$ is now the input to the decimator meaning $w_{1a} = X(z)G(z) = w_{1a}(z)$. This yields:

$$w_{1b}(z) = \frac{1}{M} \sum_{i=0}^{M-1} w_{1a}(z^{1/M})^i$$
\[ w_{1b}(z) = \frac{1}{M} \sum_{i=0}^{M-1} X(z^{1/M} W^k) G(z^{1/M} W^i) \]  \hspace{1cm} (21)

To make this look like (20), replace \( G(z) \) by \( G(z^M) \) in equation (21). This will allow \( G(z) \) to be moved to either side of the decimator, by using \( G(z^M) \) as the input to the decimator, or \( G(z) \) as the output of the decimator in Figure 8d. Note: this assumes that the delays are present as shown. A complete breakdown of this for \( M=2 \) follows. A similar type of analysis needs to be performed for \( R(z) \) in Figure 8d. For this analysis, let the input to \( R(z) \) in Figure 8d be \( w_{1b} \) and the output of \( R(z) \), which is the input to the interpolator, be \( w_{1c} \). Using (4) this yields:

\[ w_{1c}(z) = w_{1b}(z) R(z) \]

\[ w_2 = w_{1c}(z^M) = w_{1b}(z^M) R(z^M) \]  \hspace{1cm} (22)

If the interpolator and \( R(z) \) are interchanged,

\[ w_{1c} = w_{1b}(z^M) \]

\[ w_2 = w_{1c}(z) R(z) = w_{1b}(z^M) R(z) \]  \hspace{1cm} (23)

For (23) to look like (22), replace \( R(z) \) with \( R(z^M) \). This allows the transfer function to be moved to either side of the interpolator, and is shown in Figure 9, where \( S(z) \) in Figure 8c has been broken into parts and moved to the outside of the decimators and interpolators. Using the results of (18) in Figure 9 gives an input, output relationship (the delays are not shown but are assumed as discussed earlier):

\[ Y(z) = X(z) G(z^M) R(z^M) \] for Figure 9

\[ Y(z) = X(z) S(z^M) \] with \( S(z) \) as in Figure 8c and 8d.
Figure 9. Transfer Function Movement with Decimation and Interpolation.

The complete analysis of Figure 8c with $M=2$ gives:

$$X^e(z) = \frac{1}{2}[X(z) S(z^2) + X(-z)S(z^2)]z^{-1} + \frac{1}{2}[X(z) S(z^2)z^{-1} + X(-z)S(z^2)(-z^{-1})]$$

$$X^e(z) = \frac{1}{2}[X(z) + X(z)] z^{-1} S(z^2)$$

$$X^e(z) = X(z)z^{-1}S(z^2)$$

An important relationship is shown in (24) [1].

$$T(z) = \frac{X^e(z)}{X(z)} = S(z^2) z^{-1}$$  \(25\)

Equation (25) shows $T(z)$ is linear, time-invariant, and aliasing has been canceled. Additionally, since $S(z) = G(z)R(z)$, if each subband function $S_k(z)$ is independent of $k$, and is a translated version of the basic filter $S(z)$, then $X^e(z)$ will be free from aliasing. Also, if $S(z)$ is chosen such that $S(z)=1$, then $X^e(z) = X(z)z^{-1}$, which in the time domain is $x^e(n) = x(n-1)$. Therefore $x^e(n)$ is a perfect reconstruction of $x(n)$ except for a delay. If $R(z)$ is chosen such that $R(z) = 1/G(z)$, then $S(z) = 1$ [1]. For the case when $M=2$:

$$R_0(z) = 1/G_0(z) \text{ and } R_1(z) = 1/G_1(z)$$

If Figure 9 is used to replace the channels in 8d, then 8d will look like Figure 1 and can be related to the basic QMF through the following relations for $M=2$ [2].

$$H_1(z) = H_0(zW^1) = H_0(-z)$$ This was shown earlier in (14).
The analysis filters can now be designed based on the above relations. Additionally, if the analysis filters are chosen to be linear phase FIR filters, then the QMF will have perfect reconstruction. This is because of the linear phase of an FIR filter, the delay relationships that cancel aliasing, and the fact that the synthesis filters are chosen based on the analysis filters [2]. There are many techniques and programs to design the FIR filters [5],[6], and I will not get into the actual design of FIR filters. However, there is an interesting filter type that can be implemented as an FIR filter and lends itself well to the constraints of the QMF and will be discussed in the next chapter.
CHAPTER III

LATTLCE FILTERS

In the last chapter, the quadrature mirror filter was presented. In this chapter I want to analyze the lattice filter. The reason I chose this filter for the implementation of the QMF, is its unique structure and building block capabilities. Additionally, lattice structures have been used in linear prediction [7], which means there have been previous research and design applications.

**Theory Development**

In general, a stable digital filter transfer function, $H(e^{j\omega})$, is bounded if $|H(e^{j\omega})| \leq 1$ for all $\omega$, and $|H(e^{j\omega})|^2 \leq 1$. Additionally, when $|H(e^{j\omega})| = 1$ the system is said to be "lossless" [8]. In matrix notation $H(e^{j\omega})H(e^{j\omega})^H \leq I$ for all $\omega$. The filter will be all pass if the equality holds. For the case when the equality does not hold, another "lossless" FIR function can be added to force the equality to unity. In other words,

$$
|H(e^{j\omega})|^2 + |R(e^{j\omega})|^2 = 1 \text{ for all } \omega
$$

(26)
From this define a vector $\mathbf{G}(z)$ of order $N$ where $\mathbf{G}(z) = \begin{bmatrix} H(z) & R(z) \end{bmatrix}^T$, then $\mathbf{G}(z)$ is “allpass”.

We should be able to find an FIR function to add to any $\mathbf{G}(z)$ to force $\mathbf{G}(e^{j\omega})\mathbf{G}(z) = 1$. To further analyze this “lossless” function, assume a “lossless” vector

$$\mathbf{G}_{N-1}(z) = \begin{bmatrix} p_{N-1}(z) & q_{N-1}(z) \end{bmatrix}^T$$

where

$$p_{N-1}(z) = \sum_{n=0}^{N-1} p_{N-1,n} z^{-n} \quad (27)$$

$$q_{N-1}(z) = \sum_{n=0}^{N-1} q_{N-1,n} z^{-n} \quad (28)$$

Let, $\tilde{q}_{N-1}(z) = q^T(z^{-1})$ then, (26), with (27) and (28), yields an allpass property

$$\tilde{p}_{N-1}(z)p_{N-1}(z) + \tilde{q}_{N-1}(z)q_{N-1}(z) = 1 \quad (29)$$

This property (29) will be referred to as the “Power Complementary Condition” (PCI)[8], and is an important relation for a lossless filter. Using this information, the filter can be broken into $N$ building blocks where each building block has the basic form shown in Figure 10.

Following the reasoning used for equations (26), (27), (28), let

$$\mathbf{G}_{m+1}(z) = \begin{bmatrix} p_{m+1}(z) & q_{m+1}(z) \end{bmatrix}^T \quad (30)$$
Looking at Figure 10 we can also define the overall transfer function of the block as
\[ G_{m+1}(z) = T_{m+1}(z) G_m(z). \] Additionally, \( G_{m+1}(z) \) is an \((m+1)\)th order FIR allpass vector, as suggested by (27) and (28). Using the forms for (27) and (28) for this vector yields:

\[
\begin{align*}
P_{m+1}(z) &= \sum_{n=0}^{N-1} p_{m+1,n} z^{-n} \\
Q_{m+1}(z) &= \sum_{n=0}^{N-1} q_{m+1,n} z^{-n}
\end{align*}
\]

Comparing (29) with (31) and (32) gives a similar result for the allpass property:

\[
\tilde{p}_{m+1}(z)P_{m+1}(z) + \tilde{q}_{m+1}(z)Q_{m+1}(z) = 1
\]

Assume \( m=0 \), then using (31), (32):

\[
P_1(z) = p_{1,0} + p_{1,1} z^{-1} \quad \text{and} \quad Q_1(z) = q_{1,0} + q_{1,1} z^{-1}
\]

applying (33) yields:

\[
p^{2}_{1,0} + q^{2}_{1,0} + (z+z^{-1})(p_{1,0}p_{1,1} + q_{1,0}q_{1,1}) = 1
\]
For (33) to hold true implies \( p_{1,0} p_{1,1} + q_{1,0} q_{1,1} = 0 \) which in general means:

\[
p_{m+1,0} p_{m+1,1} + q_{m+1,0} q_{m+1,1} = 0 \quad \text{(34)}
\]

For analysis purposes define [8]:

\[
P_m(z) = k_m + 1 p_{m+1}(z) + u_m + 1 p_{m+1}(z) \quad \text{(35)}
\]

\[
k_{m+1} = \frac{-q_{m+1,0} p_{m+1,1}}{\sqrt{p_{m+1,0}^2 + q_{m+1,0}^2}} \quad \text{(36)}
\]

\[
u_{m+1} = \frac{p_{m+1,0}}{\sqrt{p_{m+1,0}^2 + q_{m+1,0}^2}} \quad \text{(37)}
\]

Similarly define

\[
Q_m(z) = -u_m + 1 p_{m+1}(z) + k_m + 1 q_{m+1}(z) \quad \text{(38)}
\]

We can now put (12) and (15) into matrix form and define them as \( G_m(z) \).

\[
G_m(z) = \begin{bmatrix} P_m(z) \\ Q_m(z) \end{bmatrix} = \begin{bmatrix} k_{m+1} & u_{m+1} \\ -u_{m+1} z & k_{m+1} z \end{bmatrix} \begin{bmatrix} P_{m+1}(z) \\ Q_{m+1}(z) \end{bmatrix} \quad \text{(39)}
\]

Looking at the inversion property of matrices:

\[
\begin{bmatrix} k_{m+1} & u_{m+1} \\ -u_{m+1} z & k_{m+1} z \end{bmatrix}^{-1} = \frac{1}{(k_{m+1})^2 z + (u_{m+1})^2 z} \begin{bmatrix} k_{m+1} z & -u_{m+1} z \\ u_{m+1} z & k_{m+1} z \end{bmatrix} \quad \text{and (36) and (37) imply (} k_{m+1} \text{)}^2 + (u_{m+1})^2 = 1. \quad \text{Then:}
\]


\[
\begin{bmatrix}
k_{m+1} & u_{m+1} \\
-u_{m+1}z & k_{m+1}z
\end{bmatrix}^{-1} =
\begin{bmatrix}
k_{m+1} & -u_{m+1}z^{-1} \\
u_{m+1} & k_{m+1}z^{-1}
\end{bmatrix}
\]

which leads to:

\[
G_{m+1}(z) = \begin{bmatrix}
P_{m+1}(z) \\
Q_{m+1}(z)
\end{bmatrix} =
\begin{bmatrix}
k_{m+1} & u_{m+1} \\
-u_{m+1}z & k_{m+1}z
\end{bmatrix}^{-1} \begin{bmatrix}
P_{m}(z) \\
Q_{m}(z)
\end{bmatrix}
\]

\[
G_{m+1}(z) = \begin{bmatrix}
P_{m+1}(z) \\
Q_{m+1}(z)
\end{bmatrix} =
\begin{bmatrix}
k_{m+1} & -u_{m+1}z^{-1} \\
u_{m+1} & k_{m+1}z^{-1}
\end{bmatrix} \begin{bmatrix}
P_{m}(z) \\
Q_{m}(z)
\end{bmatrix}
\]

\{40\}

The transfer function \(T_{m+1}(z)\) can be seen from \{40\} to be:

\[
T_{m+1}(z) = \begin{bmatrix}
k_{m+1} & -u_{m+1}z^{-1} \\
u_{m+1} & k_{m+1}z^{-1}
\end{bmatrix}
\]

Note that \(T_m(z)T_{m+1}(z) = 1\), which implies \(T_{m+1}(z)\) is lossless.

Breaking out the \(G_{m+1}(z)\) matrix yields:

\[
P_{m+1}(z) = k_{m+1}P_m(z) - u_{m+1}z^{-1}Q_m(z)
\]

\[
Q_{m+1}(z) = u_{m+1}P_m(z) + k_{m+1}z^{-1}Q_m(z)
\]

This relationship leads to an alternative form for \{36\} and \{37\} \[8\].

\[
k_{m+1} = \frac{P_{m+1;0}}{\sqrt{P_{m+1;0}^2 + Q_{m+1;0}^2}} \quad \{41\}
\]
Examining (39) shows that the lattice filter can be synthesized from a given function $G(z)$ using a downwards recursion from the transfer function. Additionally, (40) enables a computation of $G(z)$ from a given set of coefficients $k_0, k_1, \ldots, k_{N-1}$ through an upwards recursion. Combining these relationships we can produce a structural form for the lattice filter.

We can also produce another form of the Lattice filter by assuming the coefficients $k_m^2$ and $u_m^2$ are reasonably close to unity, which is normally true for most digital filter applications.
[8]. Keeping the coefficients close to unity and letting $\alpha_m = \frac{u_m}{k_m}$ yields:

$$T_m(z) = \frac{1}{k_m} \begin{vmatrix} 1 & \frac{u_m}{k_m} \frac{z^{-1}}{z} & \frac{z^{-1}}{z^2} \\ \frac{u_m}{k_m} \frac{z^{-1}}{z} & \frac{1}{z} & \frac{1}{z^2} \\ \frac{1}{z^2} & \frac{1}{z^2} & \frac{1}{z^2} \end{vmatrix} \text{ since } k_m \approx 1$$

Following the procedures used in [40] for this case yields:

$$M_{m+1}(z) = T_{m+1}(z) G_m(z)$$

$$G_{m+1}(z) = \begin{vmatrix} 1 & -\alpha_m z^{-1} \\ \alpha_m z^{-1} & z^{-1} \end{vmatrix} G_m(z)$$

In non-matrix form:

$$P_{m+1}(z) = P_m(z) - a_m + 1 z^{-1} Q_m(z)$$

$$Q_{m+1}(z) = a_{m+1} P_m(z) + z^{-1} Q_m(z)$$

Define: $H_m(z) = \frac{P_m(z)}{P_0(z)}$ and $B_m(z) = \frac{Q_m(z)}{Q_0(z)}$. For $m = 0$, $H_0(z) = \frac{P_0(z)}{P_0(z)} = 1$, and $B_0(z) = 1$.

This implies:

$$H_{m+1}(z) = H_m(z) - a_m + 1 z^{-1} B_m(z)$$

$$B_{m+1}(z) = a_{m+1} H_m(z) + z^{-1} B_m(z) \quad (44)$$

Let $m = 0$ to examine this further. Using $m = 0$ yields:

$$H_1(z) = H_0(z) - a_1 z^{-1} B_0(z) \Rightarrow H_1(z) = 1 - a_1 z^{-1}$$
\[ B_1(z) = a_1H_0(z) + z^{-1}B_0(z) \Rightarrow B_1(z) = a_1 + z^{-1} \]

Then \( B_1(z) = z^{-1}H_1(-z^{-1}) \). In general \( B_m(z) = z^{-m}H_m(-z^{-1}) \) \( (45) \)

Equation (44) can be used to find a recursion relation to find the coefficients \( a_{m+1} \).

\[ H_m(z) = H_{m+1}(z) + a_{m+1}z^{-1}B_m(z) \]

\[ B_m(z) = -za_m + 1 H_m(z) + zB_{m+1}(z) \]

\[ \Rightarrow H_m(z) = H_{m+1}(z) + (a_{m+1}z^{-1})(-za_m + 1 H_m(z) + zB_{m+1}(z)) \]

\[ \Rightarrow H_m(z) = H_{m+1}(z) + a_{m+1}B_m(z) \]

\[ \Rightarrow H_m(z) = \frac{H_{m+1}(z) + a_{m+1}B_m(z)}{1 + a_m^2} \]

Using (45) this yields:

\[ H_m(z) = \frac{H_{m+1}(z) + a_{m+1}z^{-m+1} H_{m+1}(-z^{-1})}{1 + a_m^2} \] \( (46) \)

In general \( H_m(z) = \sum_{n=0}^{m} h_m(n) z^{-n} \)

This applied to (45) shows \( h_m(0) = 1 \) and \( a_m = h_m(m) \) which can be used when solving for the coefficients. Equation (46) is the general recursion relationship for the lattice structure shown in Figure 12, with \( |a_{m+1}| < 1 \).
Relation to Lattice Filters Used in Linear Prediction

By looking at the results and methods used in linear prediction, an analogy of the lattice filter can be done using this previously derived information. Since linear prediction is an inverse modeling operation from adaptive filtering, structures have been derived which are efficient and easily implemented. Makhoul [9] presented a lattice filter structure which is similar to Figure 12, except both coefficients $a_m$ are positive. This structure was used in linear-predictive coding of signals. Additionally, both IIR and FIR lattice structures have been proposed in linear prediction theory [10]. Since I was using FIR filters for the QMF structure, I examined the FIR lattice structure in linear prediction. As discussed earlier, Figure 12 is the denormalized structure used in linear prediction, except that the minus sign in front of the second $a_m$ is not there.

In linear prediction theory, given a set of samples $R(p)$ that are autoregressive of order $p$, the optimal linear predictor collapses to a $p$th order predictor. For any given $p$, the projection of $y_n$ onto the past $p$ samples will still provide the best linear prediction of $y_n$ that can be made on the basis of these past $p$ samples. As $p$ increases, more and more past information is taken into
account, which means the prediction of $y_n$ becomes better yielding a smaller mean-squared error.

If $e$ is the squared error, the objective is to minimize $e$. Additionally, if $a_i$ are the predictor coefficients, and $R(i)$ are the sample values, then $e$ will be:

$$e = \sum_{i=0}^{p} R(i)a_i$$  \hspace{1cm} (47)

which in matrix form yields

$$\begin{bmatrix} R(0) & R(1) & \cdots & R(p) \\ R(1) & R(0) & \cdots & R(p-1) \\ \vdots & \vdots & \ddots & \vdots \\ R(p) & R(p-1) & \cdots & R(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} e \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$  \hspace{1cm} (48)

Notice that this matrix has the same form as the matrix for $T(z)$ in the discussion on the QMF transfer function. Using the minimization of the mean square error, take the inverse of the autocorrelation matrix to find the predictor coefficients. Since the correlation matrix is in Topelitz form, the elements along each diagonal are equal, it can be solved by Levinson's method (N. Levinson proposed an elegant fast algorithm for solving these equations) [9]. This method is much faster than matrix inversion since it is proportional to the square of the number of coefficients to be determined, instead of the third power. This algorithm exposes the lattice structure and has been found to be extremely useful in a wide variety of signal processing applications, including spectral estimation, voice coding, and filter structures. The algorithm was rediscovered and improved by Durbin [6]. The Durbin procedure can be used to solve for a system of equations by recursion over $N$ stages. The important point here to note is that the error can be reduced to any desired value, by increasing the number of stages, since the error is reduced at each succeeding stage. Moreover, the procedure is sequential and can be easily solved by computer. This is another reason the lattice filter is attractive as a filter structure for the QMF. If more
attenuation is needed, add more sections. Appendix C is a listing of the Levinson–Durbin algorithm. Notice that this same algorithm, modified, can be used to find the coefficients for the QMF lattice filter. Also, (45) is the relation that will be used in the Levinson recursion.
CHAPTER IV
LATTICE FILTER APPLICATION TO QUADRATURE MIRROR FILTERS

As discussed earlier, QMF’s were presented followed by the lattice filter. What I want to do now is show a structure for the lattice filter that is ideal for a QMF bank. Because the two channel mirror filter (hence name quadrature) will need analysis and synthesis filters that are FIR half band mirrors, some parts of the general lattice drop out. The computer program to find the lattice coefficients was based on this structure.

Lattice Structure

With \( M=2 \), the QMF naturally turns out to be two channel. This means decimation and interpolation by a factor of two. For this case, the filters to perform this turn out to have symmetry about \( \pi/2 \). If the desired QMF analysis filters are of the form \( |H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2 = 1 \), for all \( \omega \), then they can be of the form [1]:

\[
H_0(z) = \frac{(A_0(z) + A_1(z))}{2} \tag{49}
\]

\[
H_1(z) = \frac{(A_0(z) - A_1(z))}{2} \tag{50}
\]

Here \( A_0(z) \) and \( A_1(z) \) are allpass functions. Additionally, this analysis bank can be implemented by the structure in Figure 13. Also, design \( H_0(z) \) such that its magnitude response has symmetry around \( \omega=\pi/2 \), and \( H_1(z) = H_0(-z) \). If the synthesis filters are then chosen as described in the chapter on QMF’s, the overall transfer function \( T(z) \) will be:
In this case (51) is allpass! Additionally, note that the structure of Figure 13 is of the same form as the lattice structure, which suggests the lattice structure as an ideal form.

\[ T(z) = \frac{A_0(z)A_1(z)}{2} \]  

(51)

By specifying symmetry of the filters around π/2, the filters will become FIR "Half-band" filters. In other words \( H(z) - H(-z) = c \cdot z^{-(N-1)} \), which means the stopband error curve is an image of the passband error curve [12]. This yields for the analysis and synthesis filters described in Chapter II:

\[ H_1(z) = z^{-(N-1)}H_0(-z^{-1}) \]  

(53)

\[ F_0(z) = z^{-(N-1)}H_0(z^{-1}) \]  

(54)

\[ F_1(z) = z^{-(N-1)}H_1(z^{-1}) \]  

(55)

These filters will then allow for perfect reconstruction as described earlier. Using these results it can be seen that N-1 will be odd. To see this, use \( F_0 \) for example. \( F_0(z) + F_0(-z) = \) constant (usually 1). For N-1 even the coefficient corresponding to \( z^{-(N-1)} \) is zero. Hence, N-1 can always be taken to be odd without loss of generality [8]. This means that the even coefficients in the lattice structure will be zero. This can be used by reducing the structural complexity, since
all the even blocks do not have to be used since their coefficient will be zero. Using the analysis and structure presented in Chapter III, and the fact that the even coefficients are zero, the lattice filter can be found for the QMF case as shown below in Figure 14.

![Lattice Structure for QMF](image)

Figure 14. Lattice Structure for QMF.

The lattice structure in Figure 14 is similar to the structures analyzed earlier. For the QMF case, $c = \frac{1}{\sqrt{2}}$, $a_m$ are defined as before, and $B = k_1k_2k_3...k_{N-1}$ or $B^2 = \prod_{m=1}^{N-1} \frac{1}{1+a_m^2}$. In this case the coefficients $a_m$ are no longer bounded above by unity. Additionally, the transfer functions become:

$$H_0(z) = B*P_{N-1}(z) = P'_{N-1}(z)$$

(56)

$$H_1(z) = B*Q_{N-1}(z) = Q_{N-1}(z)$$

(57)

**Computer Aided Design Application**

The coefficients for $P(z)$ and $Q(z)$ can be found from the recursion relations described in Chapter III. From there $P'(z)$ and $Q'(z)$ can be found. The recursion relations are set up in an algorithm that is similar to the Levinson-Durbin algorithm. The FORTRAN program looks for a stopband frequency and an attenuation factor. Based on this information, a table scan is done to find a set of coefficients that will meet those specifications. These coefficients were determined in
a manner similar to J.D. Johnson [10]. In this paper the coefficients were determined from a standard set of filter design parameters using the Hooke and Jeaves algorithm. The Hooke and Jeaves algorithm is a search algorithm that attempts to minimize a single objective function of several variables. The objective function used here was based on the stopband and ripple energy. These coefficients were further modified in [12] using a quasi-Newton method to optimize for the lattice structure. Once the program finds the proper set of lattice coefficients, it uses the recursion algorithm to convert the lattice coefficients into FIR impulse coefficients, that can be used as the FIR structure of the analysis or synthesis filters. The algorithm computes the impulse coefficients from the optimal lattice coefficients. Because the Levinson algorithm is recursive in both the forward and backward directions, a program could be written to find the lattice coefficients given the impulse response coefficients. For example, the impulse coefficients could have been computed from a standard FIR design program such as the McClellan-Parks algorithm [6], then convert these to lattice coefficients and implement the FIR filter in a lattice structure. I found that the optimal coefficients given in [12] give very good results for almost any QMF design, since the analysis and synthesis filters are mirrored around \( \pi/2 \), which means you will normally design at that frequency and adjust the transition band accordingly.
CHAPTER V
RESULTS AND DISCUSSION

In this chapter I will show how the simulation of the QMF was set up, and explain some of the theory and results. The program to find the Lattice coefficients was written in FORTRAN as stated earlier. These Lattice coefficients were then converted to appropriate Impulse coefficients by the same program and stored on disk. MATHCAD® was then used to simulate the Lattice Filters and QMF bank. The MATHCAD® programs are listed in Appendix D. Additionally, most of the results were plotted and are shown as Figures in this chapter.

Filter Design Example

A filter design program was written to make use of optimal coefficients that have been found from previous research which are based on a nonlinear optimization algorithm [12]. These coefficients are optimized and give good results for most design applications. The program was written in FORTRAN and is provided in disk form as an executable file. It begins by asking for a stopband in tenths of PI. This corresponds to the desired stopband of the analysis filters. The normal range is 0.78 to 0.5. With 0.5 being the stopband for an ideal half-band filter. The program then asks for attenuation in decibels. The normal range for this is approximately 20 to 80 DB. For both these inputs, the program looks for a floating point input, so a decimal point must be input with the number. The program then does a search of optimal coefficients that best fit the input stopband and attenuation. Note that the coefficients are optimal in the sense of a filter that is a very close approximation to the desired filter. The program prints out these lattice coefficients, then does a recursion on these coefficients. Basically the recursion multiplies out the terms of the lattice to get the final output for the last block. The output of the last block is the transfer function of the lattice, and it's terms give the impulse response of the lattice filter.
Because the recursion is based on the lattice structure shown in Figure 14, two sets of coefficients are given, \( P(z) \) and \( Q(z) \) as discussed in chapter IV. To show how the Lattice filter improves with each block added, I wrote a MATHCAD® program to plot the frequency response of a typical lattice filter for each building block. In order to keep it simple and show the point, the parameters were chosen to give a four block lattice. For this example, the parameters chosen were \( \omega_s = 0.78\pi \) and an attenuation factor of 41 db. The Lattice coefficient program gave the desired lattice and impulse response coefficients. I then wrote the MATHCAD® program LATTICE. This program uses the lattice coefficients generated by LATCOF. In this program I typed them in by hand as coefficients \( a_1 \) through \( a_4 \). The program then generates the frequency response of each lattice block as \( P_1(n), ..., P_7(n) \), and \( Q_1(n), ..., Q_7(n) \). The frequency response plots of \( P_3(n) \), \( P_5(n) \), and \( P_7(n) \) are shown in Figure 15. Notice here that each succeeding stage gives an improvement on the frequency response of the lattice. Additionally, note that the response of the lattice is approximately 40 db down at a normalized frequency of 0.4 (0.78\( \pi \)). In order to examine the impulse coefficients generated by LATCOF, the MATHCAD® program MAGNITUDE was written. This program takes the lattice impulse response coefficients \( P_0, ..., P_{N-1} \) that were stored on disk and finds the frequency response through a Fast Fourier Transform (FFT) analysis. For this example, LATCOF was run with stopband 0.62 and attenuation of 45db. MAGNITUDE was run with the output frequency response shown in Figure 16. Note here that the stopband is close to 0.31 (0.62\( \pi \)) with an attenuation of approximately 45db. MAGNITUDE was run again with \( P_0, ..., P_{N-1} \) replaced with \( Q_0, ..., Q_{N-1} \). This output is shown in Figure 17. Notice the plot is a high pass mirror of Figure 16, as expected, with similar results.

**Example of Lattice Filter applied to a Quadrature Mirror Filter Bank**

A set of programs was written to simulate the QMF bank. They start out with COEFSTORE, which reads the lattice impulse coefficients from disk that were stored during the lattice FORTRAN program LATCOF.
Figure 15. Lattice Filter Blocks Frequency Response.
Figure 16. Lattice Filter $P_{N-1}(z)$ Frequency Response.
Figure 17. Lattice Filter $Q_{N-1}(z)$ Frequency Response.
COEFSTORE transfers the impulse coefficients of \( P(z) \) and \( Q(z) \) from LATCOF to the analysis filters \( H_0(z) \) and \( H_1(z) \) respectively. Using the analysis filters, the program forms the synthesis filters from the analysis filter coefficients using (54) and (55). The newly formed filter's impulse response coefficients are stored on disk for the next programs. This was done to conserve memory in the computer, so that the programs could be run on an IBM compatible machine. For the input data, the program DATASAVE was written which simulates a desired input signal, samples it and stores the coefficients on disk to be used by FFTSAVE. This program is generic and needs to be changed anytime the input signal changes. Each version of the input signal used in the QMF analysis is shown as a separate DATASAVE program. As an additional item, this program adds \( N-1 \) zeros to the input signal. This was done in order to perform linear convolution. In general, if an \( N \) point Discrete Fourier Transform (DFT) is performed, and convoluted with another \( N \) point DFT, the resulting multiplication in frequency should yield a \( 2N-1 \) point inverse DFT for the convolution in time. However, because the original DFT's were \( N \) point, the resulting \( N \) point inverse DFT will yield circular convolution. If this is done, the system will produce an output which is similar to the desired results, but the system will be non-causal. To take care of this problem, the DFT's were formed of \( 2N-1 \) points, where the last \( N-1 \) points were zeros. Now the \( 2N-1 \) point DFT's when multiplied and the inverse DFT is taken, the resulting time domain response will be linear convolution as desired \([11]\). From here the program FFTSAVE is used to take the FFT of the filter impulse response coefficients and the input signal and form the frequency response which is stored on disk. In order to see the output of the analysis filters, the program ANALYSIS was written. It takes the input signal and the analysis filter response and performs linear convolution through use of the FFT. The frequency response of the analysis filters is plotted. To simulate the QMF, the program QMF was written. The purpose of this program is to use the frequency response of the analysis filters and input signal, multiply to find the filter output frequency response and send to the synthesis filter frequency response. The output of the
synthesis filters are added then Inverse Fourier Transformed to get the time domain output. Several examples were run and are shown in the following Figures 18-23. Figure 18 shows the analysis filters frequency response to a unit sample input. The mirror image of the filters is shown here. Figure 19 shows the time domain output when a unit sample is applied. Note that the output is delayed by 47 samples. This is expected since the QMF analysis theory showed that the output will be a delayed version of the input. The reason it's 47, is because LATCOF was run using a stopband of 0.6π. LATCOF gave a filter with 48 impulse coefficients. This delay also corresponds to the Synthesis and Analysis filter order N−1=47. Figure 20 shows the output of the analysis filters to a low frequency sinewave input. As expected, the lowpass analysis filter H₀ has the large response while the highpass filter H₁ has the small response indicated by the two plots. Figure 21 is the QMF time domain output of the low frequency input. Figure 22 is the output of the analysis filters to a high frequency input. In Figure 22, the analysis filter outputs show H₀ as small in magnitude, while H₁ has large amplitude. Figure 23 shows the QMF time domain output for the high frequency input. The flowchart is presented to show the way the programs were used to produce the results for the QMF analysis.
Figure 18. Analysis Filters Output to Unit Sample Input.
Figure 19. QMF Output to Unit Sample Input.
Figure 20. Analysis Filters Output to Low Frequency Sinewave Input.
Figure 21. QMF Output to Low Frequency Input.
Figure 22. Analysis Filters Output to High Frequency Sinewave Input.
Figure 23. QMF Output to High Frequency SineWave Input.
Flowchart

Load and Run FORTRAN Program LATCOF

Enter Stopband Frequency in tenths of Normalized radians of PI. Ex 0.6, which corresponds to 0.6π.

Enter the desired stopband attenuation of the Lattice Filter with decimal point in Db. Ex 70DB would be entered as 70.0

A Table lookup is now done to find a set of optimal coefficients that is close to the desired parameters.

The optimal lattice coefficients close to the parameters are printed out.

A recursion routine is run to translate the lattice coefficients into impulse response coefficients. The recursion is an update of the coefficients for each added lattice section.

\[ P_{m+1}(z) = P_m(z) - a_m z^{-1} Q_m(z) \]

\[ Q_{m+1}(z) = a_m + 1 P_m(z) + z^{-1} Q_m(z) \]

Printout the impulse response coefficients and save them to disk.

Load MATHCAD©

Load Program DATASTORE
Write the desired equation for the input signal.

Program will save the data to disk for processing
### Load Program COEFSTORE
This program sets up the analysis and synthesis filters based on the previously found coefficients using equations (53) to (55).

### Load Program FFTSAVE
This program reads the filter coefficients and signal samples. Using a FFT routine, it computes the Fourier coefficients and stores them on disk in a file for each filter. Note that \( N \) zeros are added in order to accomplish convolution in later programs.

### Load Program ANALYSIS
This program reads the transformed coefficients along with the transformed signal coefficients. Using convolution through multiplication in frequency, the Analysis filter output is computed. This output is then plotted, showing the two Analysis filter outputs.

### Load Program QMF
This program takes the transformed signal and filters and performs the needed convolutions. The outputs of the synthesis filters are added then an Inverse FFT is performed to get to the time domain. The output time signal is plotted along with the input signal.

## Discussion
This paper was a presentation of Quadrature Mirror Filters using Lattice filters. The general theory of the QMF and Lattice filter was presented, followed by application of the Lattice filter to a QMF bank. I then used a computer to design the filter and simulate the system. The results as presented, show that the theory holds for the lattice coefficients. It's interesting to see that as lattice building blocks are added the filter response improves. This implies that more attenuation can be achieved by adding filter blocks. Additionally, the lattice stills holds it's basic
response with removal of blocks. This is not possible with a direct form, since removal of coefficients destroys the response of the system. Another interesting property that is shown by the lattice filter is the response of \( P(z) \) depends on \( Q(z) \) and vice-versa. This gives the mirror property, but it also suggests that whatever you do to the basic structure such as quantization, affects each side, so that there errors in the PCI condition will balance out in the overall transfer function. Indeed, if quantization is a problem in your QMF system, the lattice structure is ideal, because the quantization errors will not affect the perfect reconstruction property. This can possibly be an advantage when the lattice filter is implemented in hardware. For each block, two multiplications, one delay, and two additions are needed. This is comparable to a direct form implementation, although you could get by with less multipliers in a direct form. As far as speed goes, in the QMF with the decimation, the rate of operation is cut in half, which is seen in the filters since the even coefficients are zero. The plots of the QMF show that the outputs are a reconstruction of the input except for a scale factor and a delay. Therefore as suggested the system does produce a reconstructed output of the input while allowing the input signal to be broken into smaller subbands for coding or other operations. As far as my research goes, I looked at the lattice filter as it applies to Quadrature Mirror Filters. I feel further research could be done on the application of Lattice Filters to other areas, in particular speech processing, where signals must be broken down, analyzed or operated on and either reconstructed or modified in the output. Additionally I only examined the two channel QMF and the application of lattice filters to it, much more research could be done in the application to multirate filtering. Finally, the filter program was written based on a set of previously computed optimal filters [12], the output being one of these optimal filters that comes closest to the users desired response. In almost all cases this is optimal, but further research could be done to optimize any user input.
PROGRAM OR SUBROUTINE NAME: LATCOF

AUTHOR : GREGORY R. JASPERS

DATE : 30 JUN 88

MODIFIERS : NONE

VERSION : 0

PURPOSE : PROGRAM USES A TABLE LOOKUP TO FIND OPTIMAL LATTICE COEFFICIENTS, THEN USES A MODIFIED LEVINSON ALGORITHM TO FIND THE IMPULSE RESPONSE COEFFICIENTS.

ABSTRACT : DESIGNER INPUTS THE DESIRED STOPBAND AS TENTHS OF PI AND DESIRED ATTENUATION FOR THE TRANSITION BAND. PROGRAM GENERATES THE LATTICE COEFFICIENTS AND IMPULSE RESPONSE COEFFICIENTS.

DESCRIPTION OF PARAMETERS:

INPUT : STOPBAND FREQUENCY AND ATTENUATION IN DB.

OUTPUT : PRINTOUT OF LATTICE COEFFICIENTS AND IMPULSE RESPONSE COEFFICIENTS TO A PRINTER AND TO FLOPPY DISK.

FORMAT OF FILES

INPUT : NONE REQUIRED

OUTPUT : IMPULSE RESPONSE COEFFICIENTS TO 8 DECIMAL PLACES SEPARATED BY COMMA'S.
REAL C8A(5), C12A(7), C16A(9), C12B(7), C16B(9),
C24B(13), C16C(9), C24C(13), C32C(17), C48C(25),
C32F(17), C64E(33), C32E(17), C32D(17), C24D(13),
C16F(9), C24F(13), SCRAP3(35), C48D(25), C64D(33),
C70D(36), C48F(25), C48E(25), SCRAP(36), PHAT(70),
QHAT(70), AT
INTEGER JJ

DATA C8A /4.0,-0.2638026e+01,0.7454463e+00,
1 -0.2598479e+00,0.6388361e-01/

DATA C12A /6.0,-0.3676246e+01,0.1100022e+01,
1 0.5170637e+00,0.2362183e+00,-0.8441314e-01,
2 0.1716341e-01/

DATA C12B /6.0,-0.3096168e+01,0.9370946e+00,
1 -0.4569771e+00,0.2276283e+00,-0.9712722e-01,
2 0.2795064e-01/

DATA C16A /8.0,0.4699145e+01,0.1465103e+01,
1 -0.7597957e-00,0.4216733e-00,-0.2181804e-00,
2 0.9405991e-01,-0.2924380e-01,0.4905888e-02/

DATA C16B /8.0,0.3886354e-01,0.1218756e+01,
1 -0.6429331e+00,0.3707214e+00,-0.2068881e+00,
2 0.1023296e+00,-0.4016767e-01,0.9948452e-02/

DATA C16C /8.0,-0.2966504e+01,0.9334946e+00,
1 -0.5028173e+00,0.3051719e+00,-0.1879673e+00,
2 0.1104991e+00,-0.5811574e-01,0.2437997e-01/

DATA C16F /8.0,-0.2739714e+01,0.8611533e+00,
1 -0.4660051e+00,0.2868931e+00,-0.1817166e+00,
2 0.1120899e+00,-0.6389849e-01,0.3098943E-01/

DATA C24B /12.0,-0.5462615e+01,0.1765271e+01,
1 -0.9933745e+00,0.6405501e+00,-0.4290425e+00,
2 0.2844221e+00,-0.1798644e+00,0.1045704e+00,
3 -0.5351832e-01,0.2273594e-01,-0.7266256e-02,
4 0.1373516e-02/

DATA C24C /12.0,-0.3965091E+01,0.1282368e+01,
1 -0.7260119e+00,0.4756307e+00,-0.3284680e+00,
2 0.2293814e+00,-0.1576285e+00,0.1040772e+00,
3 -0.6431627e-01,0.3591622e-01,-0.1709523e-01,
4 0.6056575e-02/
| DATA C24F | /12.0,-0.3587773e+01,0.1158946E+01, |
| 1 | 0.6565104e+00,0.4319230e+00,0.3010528e+00, |
| 2 | 0.2135914e+00,0.1504406e+00,0.1030309e+00, |
| 3 | -0.6712935e-01,0.4045162e-01,-0.2154244e-01, |
| 4 | 0.9188643e-02/ |
| DATA C24D | /12.0,-0.3219439e+01,0.1037174e+01, |
| 1 | 0.5871553e+00,0.3877585e+00,0.2728663e+00, |
| 2 | 0.1968799e+00,-0.1423481e+00,0.1013410e+00, |
| 3 | -0.6975467e-01,0.4582990e-01,-0.2720419e-01, |
| 4 | 0.1403098e-01/ |
| DATA C32C | /16.0,-0.4947162e+01,0.1618750e+01, |
| 1 | 0.9366921e+00,0.6339676e+00,-0.4583882e+00, |
| 2 | 0.3411142e+00,-0.2558051e+00,0.1904199e+00, |
| 3 | -0.1388874e+00,0.9798419e-01,-0.6591646e-01, |
| 4 | 0.4157153e-01,0.2402169e-01,0.1228736e-01 |
| 5 | 0.5224147e-02,0.1576151e-02/ |
| DATA C32F | /16.0,-0.4419554e+01,0.1444688e+01, |
| 1 | 0.8358118e+00,0.5664486e+00,-0.4110873e+00, |
| 2 | 0.3079806e+00,-0.2334207e+00,0.1764921e+00, |
| 3 | -0.1316083e+00,0.9572774e-01,-0.6712079e-01, |
| 4 | 0.4473385e-01,0.2781714e-01,0.1569589e-01 |
| 5 | 0.7645295e-02,0.2854061e-02/ |
| DATA C32D | /16.0,-0.3897719e+01,0.1271721e+01, |
| 1 | 0.7545884e+00,0.4981979e+00,-0.3628759e+00, |
| 2 | 0.2738343e+00,-0.2099638e+00,0.1614840e+00, |
| 3 | -0.1233244e+00,0.9266442e-01,0.6785856e-01, |
| 4 | 0.4790362e-01,0.3214141e-01,0.2007797e-01, |
| 5 | 0.1127087e-01,0.5264834e-02/ |
| DATA C32E | /16.0,-0.2937657e+01,0.9474481e+00, |
| 1 | 0.5416529e+00,0.3658746e+00,-0.2676179e+00, |
| 2 | 0.2048268e+00,-0.1610002e+00,0.1285583e+00, |
| 3 | -0.1034792e+00,0.8345957e-01,0.6709826e-01, |
| 4 | 0.5350337e-01,0.4208714e-01,0.3245102e-01 |
| 5 | 0.2431732e-01,0.1748662e-01/ |
| DATA C48C | /24.0,-0.6315784e+01,0.2088393e+01, |
| 1 | 0.1232629e+01,0.8584247e+00,-0.6447275e+00, |
| 2 | 0.5040390e+00,-0.4027649e+00,0.3252960e+00, |
| 3 | -0.2634520e+00,0.2125598e+00,-0.1699807e+00, |
| 4 | 0.1339247e+00,-0.1034047e+00,0.7778612e-01, |
| 5 | -0.5664952e-01,0.3966167e-01,-0.2648138e-01, |
| 6 | 0.1670176e-01,-0.9831959e-02,0.5316054e-02, |
| 7 | 0.2578539e-02,0.1079437e-02,-0.3618121e-03, |
DATA C48F /24.0,-.6057506e+01,0.2000152e+00,0.7954759e-04/
1 -0.1178251e+01,0.8193606e+00,-0.6152533e+00,
2 0.4817286e+00,-0.3863457e+00,0.3139650e+00,
3 -0.2566057e+00,0.2097031e+00,-0.1705025e+00,
4 0.1372864e+00,-0.1089672e+00,0.8193606e+00,
5 -0.6152533e+00,0.4756036e+00,0.3381633e+00,
6 0.2299884e-01,-.1481523e-01,0.8920900e-02,
7 -0.4925761e-02,0.2417364e-02,0.9931265e-03,
8 0.2925763e-03/

DATA C48D /24.0,-.5228952e+01,0.1724735e+00,0.7954759e-04/
1 -0.1014721e+01,0.7051436e+00,-0.5296424e+00,
2 0.4153386e+00,-0.3341087e+00,0.2728079e+00,
3 -0.2244915e+00,0.1851713e+00,-0.1524203e+00,
4 0.1247027e+00,-0.1010221e+00,-0.8072150e-01,
5 -0.6336105e-01,0.4863688e-01,0.3632357e-01,
6 0.2623183e-01,0.1817794e-01,0.1196394e-01,
7 -0.7368346e-02,0.4146564e-02,0.2039489e-02,
8 0.787339e-03/

DATA C48E /24.0,-.3836487e+01,0.1247866e+01,0.7954759e-04/
1 -0.7220668e+00,0.4951553e+00,-0.3688423e+00,
2 0.2885146e+00,-0.2327588e+00,0.1913137e+00,
3 -0.1598938e+00,0.1348106e+00,-0.114032le+00,
4 0.9681786e-01,0.8223478e-01,0.6963367e-01,
5 -0.5867790e-01,0.4913793e-01,0.4081778e-01,
6 0.3353566e-01,-0.2713113e-01,0.2149517e-01,
7 -0.1658255e-01,0.1238607e-01,-0.8895189e-02,
8 0.6072120e-02/

DATA C64D /32.0,-.6541668e+01,0.2167132e+01,0.7954759e-04/
1 -0.1284988e+01,0.9024673e+00,-0.6869024e+00,
2 0.5474343e+00,-0.4490405e+00,0.3753422e+00,
3 -0.3176566e+00,0.2709592e+00,-0.2321494e+00,
4 0.1992207e+00,-0.1708276e+00,0.1460453e+00,
5 -0.1242279e+00,0.1049204e+00,-0.8780164e-01,
6 0.7264466e-01,0.5928753e-01,0.4761084e-01,
7 -0.3751935e-01,0.2892705e-01,0.2174538e-01,
8 0.1587490e-01,0.1120080e-01,0.7592156e-02,
9 -0.4904821e-02,0.2987045e-02,-0.1686799e-02,
+ 0.8595028e-03,0.3749449e-03,0.1224688e-03/

DATA C64E /32.0,-.4327252e+01,0.1425816e+01,0.7954759e-04/
1 -0.8392532e+00,0.5855141e+00,-0.4435185e+00,
2 0.3525763e+00,0.2891957e+00,0.2423609e+00,
3 -0.206240e+00,0.1773966e+00,-0.1537840e+00,
4 0.1340248e+00, -0.1171987e+00, 0.1026636e+00,  
5 -0.8996029e-01, 0.7875330e-01, -0.6879375e-01,  
6 0.5989449e-01, -0.5191327e-01, 0.4474097e-01,  
7 -0.3829306e-01, 0.3250329e-01, -0.2731889e-01,  
8 0.2269681e-01, -0.1860079e-01, 0.1499897e-01,  
9 -0.1186202e-01, 0.9161671e-02, -0.6869633e-02,  
+ 0.4956829e-02, -0.3392964e-02, 0.2146366e-02/  

DATA C70D /35.0, -0.6404257e+01, 0.2126039e+01,  
1 -0.1265223e+01, 0.8926746e+00, -0.6829382e+00,  
2 0.5472484e+00, -0.4514609e+00, 0.3796413e+00,  
3 -0.3233572e+00, 0.2777332e+00, -0.2397611e+00,  
4 0.2074910e+00, -0.1796115e+00, 0.1552163e+00,  
5 -0.1336672e+00, 0.1145095e+00, -0.9741762e-01,  
6 0.8215835e-01, -0.6856392e-01, 0.5651222e-01,  
7 0.4591068e-01, 0.3668317e-01, -0.2875914e-01,  
8 0.2206526e-01, -0.1651930e-01, 0.1202692e-01,  
9 -0.8481189e-02, 0.5764517e-02, -0.3752771e-02,  
+ 0.2320577e-02, 0.1347044e-02, 0.7210614e-03,  
* 0.3454922e-03, 0.1398132e-03, -0.4105982e-04/  

C  Set up for Stopband attenuation

OPEN(6,FILE='PRN')
OPEN(7,FILE='A:COEF.DAT')
FORMAT(2X, 'INPUT WS*PI   EX 0.63')
FORMAT(F8.3)
FORMAT(/)
WRITE(6,506)
WRITE(*,500)
READ(*,501) WS
IF(WS .LT. 0.56) GOTO 26
IF(WS .LT. 0.59) GOTO 25
IF(WS .LT. 0.61) GOTO 24
IF(WS .LT. 0.66) GOTO 23
IF(WS .LT. 0.74) GOTO 22
IF(WS .GE. 0.74) GOTO 21

C  This section gets the coefficients based on desired attenuation

504 FORMAT(2X, 'WS', F5.3, 10X, 'ATTENUATION', F5.1)
21 WRITE(*,503)
READ(*,501) AT
WRITE(6,504) WS, AT
IF(AT .LT. 45.0) THEN
   CALL COEFIX(C8A(1),C8A,SCRAP)
GOTO 27
ELSEIF(AT .LT. 67.0) THEN
   CALL COEFIX(C12A(1),C12A,SCRAP)
   GOTO 27
ENDIF
CALL COEFIX(C16A(1),C16A,SCRAP)
GOTO 27
22 WRITE(*,503)
READ(*,501) AT
WRITE(6,504) WS, AT
IF(AT .LT. 46.0) THEN
   CALL COEFIX(C12B(1),C16B,SCRAP)
   GOTO 27
ELSEIF(AT .LT. 63.0) THEN
   CALL COEFIX(C16B(1),C16B,SCRAP)
   GOTO 27
ENDIF
CALL COEFIX(C24B(1),C24B,SCRAP)
GOTO 27
23 WRITE(*,503)
READ(*,501) AT
WRITE(6,504) WS, AT
IF(AT .LT. 39.0) THEN
   CALL COEFIX(C16C(1),C16C,SCRAP)
   GOTO 27
ELSEIF(AT .LT. 51.0) THEN
   CALL COEFIX(C24C(1),C24C,SCRAP)
   GOTO 27
ELSEIF(AT .LT. 71.0) THEN
   CALL COEFIX(C32C(1),C32C,SCRAP)
   GOTO 27
ENDIF
CALL COEFIX(C48C(1),C48C,SCRAP)
GOTO 27
24 WRITE(*,503)
READ(*,501) AT
WRITE(6,504) WS, AT
IF(AT .LT. 34.0) THEN
   CALL COEFIX(C16F(1),C16F,SCRAP)
   GOTO 27
ELSEIF(AT .LT. 43.0) THEN
   CALL COEFIX(C24F(1),C24F,SCRAP)
   GOTO 27
ELSEIF(AT .LT. 59.0) THEN
   CALL COEFIX(C32F(1),C32F,SCRAP)
   GOTO 27
ENDIF
CALL COEFIX(C48F(1),C48F,SCRAP)
GOTO 27
WRITE(*,503)
READ(*,501) AT
WRITE(6,504) WS,AT
IF(AT .LT. 36.0) THEN
   CALL COEFIX(C24D(1),C24D,SCRAP)
   GOTO 27
ELSEIF(AT .LT. 48.0) THEN
   CALL COEFIX(C32D(1),C32D,SCRAP)
   GOTO 27
ELSEIF(AT .LT. 65.0) THEN
   CALL COEFIX(C48D(1),C48D,SCRAP)
   GOTO 27
ELSEIF(AT .LT. 77.0) THEN
   CALL COEFIX(C64D(1),C64D,SCRAP)
   GOTO 27
ENDIF
CALL COEFIX(C70D(1),C70D,SCRAP)
GOTO 27
WRITE(*,503)
READ(*,501) AT
WRITE(6,504) WS,AT
IF(AT .LT. 27.0) THEN
   CALL COEFIX(C32E(1),C32E,SCRAP)
   GOTO 27
ELSEIF(AT .LT. 36.0) THEN
   CALL COEFIX(C48E(1),C48E,SCRAP)
   GOTO 27
ENDIF
CALL COEFIX(C64E(1),C64E,SCRAP)
CONTINUE
WRITE(6,506)
JJ=SCRAP(1)
DO 30 1=2,JJ+1
   WRITE(6,502) I-1,SCRAP(I)
   SCRAP3(I-1)=SCRAP(I)
30 CONTINUE
WRITE(6,506)
K=JJ+JJ
CALL GETPQ(SCRAP3,JJ,K,PHAT,QHAT)
DO 31 I=1, JJ+JJ

C Printout Impulse response coefficients

WRITE (6, 505) I, PHAT(I), I, QHAT(I)
WRITE (7, 507) PHAT(I)
31 CONTINUE
STOP
END

C This subroutine takes the lattice coefficients and sets
C them up in an array for printing and use by the impulse
C response subroutine

SUBROUTINE COEFIX(J, SCRAPl, SCRAP2)
INTEGER J1, J2
REAL SCRAPl(J), SCRAP2(36)
J1 = SCRAPl(1)
DO 60 I = 1, J1+1
SCRAP2(I) = SCRAPl(I)
60 CONTINUE
RETURN
END

C This subroutine finds coefficients for P(z) and Q(z)
C
SUBROUTINE GETPQ(ALPHA, N, J, P, Q)
REAL P(J), Q(J), pold(100), qold(100)
REAL alpha(n)

n2 = n+n

C compute scale factor

prod = alpha(1) * alpha(1) + 1
DO 10 I = 2, N
prod = prod * (alpha(I) * alpha(I) + 1)
10 CONTINUE
beta = SQRT(0.5/ prod)
pold(1) = 1
pold(2) = -alpha(1)
quold(1) = alpha(1)
quold(2) = 1
m = 2
p(1) = pold(1)
p(2) = pold(2)
do 40 j=4,n2,2
  q(1)=alpha(m)*pold(1)
  q(2)=alpha(m)*pold(2)
  if(j.eq.4) goto 22
do 20 i=3,j-2
  p(i)=pold(i)-alpha(m)*qold(i-2)
  q(i)=alpha(m)*pold(i)+qold(i-2)
20 continue
22 do 25 i=j-1,j
  p(i)=-alpha(m)*qold(i-2)
  q(i)=qold(i-2)
25 continue
do 30 i=1,j
  pold(i)=p(i)
  qold(i)=q(i)
30 continue
m=m+l
40 continue
do 50 i=1,n2
  p(i)=beta*p(i)
  q(i)=beta*p(i)
50 continue
end
APPENDIX B

FILTER PROGRAM RUN
ATTENUATION 45.0

Lattice Coefficients

\[ a_1 = -3.96509100 \]
\[ a_2 = 1.28236800 \]
\[ a_3 = -0.72601190 \]
\[ a_4 = 0.47563070 \]
\[ a_5 = -0.32846800 \]
\[ a_6 = 0.22938140 \]
\[ a_7 = -0.15762850 \]
\[ a_8 = 0.10407720 \]
\[ a_9 = -0.06431627 \]
\[ a_{10} = -0.03591627 \]
\[ a_{11} = -0.01709523 \]
\[ a_{12} = 0.00605658 \]

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 = 0.70494370E-01</td>
<td>Q 0 = 4.2695440E-03</td>
</tr>
<tr>
<td>1 = 0.27951660E+00</td>
<td>Q 1 = 1.6929130E-02</td>
</tr>
<tr>
<td>2 = 0.46913240E+00</td>
<td>Q 2 = 1.6361740E-02</td>
</tr>
<tr>
<td>3 = 0.34849320E+00</td>
<td>Q 3 = -2.6679000E-02</td>
</tr>
<tr>
<td>4 = -0.24238570E-01</td>
<td>Q 4 = -0.56341740E-02</td>
</tr>
<tr>
<td>5 = -0.20649250E+00</td>
<td>Q 5 = 0.28345480E-02</td>
</tr>
<tr>
<td>6 = -0.33948510E-01</td>
<td>Q 6 = 1.2521250E-01</td>
</tr>
<tr>
<td>7 = 0.12880190E+00</td>
<td>Q 7 = 0.11832640E-02</td>
</tr>
<tr>
<td>8 = 0.33139230E-01</td>
<td>Q 8 = 0.22928650E-01</td>
</tr>
<tr>
<td>9 = -0.85224920E-01</td>
<td>Q 9 = 0.32691750E-02</td>
</tr>
<tr>
<td>10 = -0.21976500E-01</td>
<td>Q 10 = 0.37468870E-01</td>
</tr>
<tr>
<td>11 = 0.57272140E-01</td>
<td>Q 11 = 0.11111580E-01</td>
</tr>
<tr>
<td>12 = 0.11111580E-01</td>
<td>Q 12 = -0.57272140E-01</td>
</tr>
<tr>
<td>13 = -0.37468870E-01</td>
<td>Q 13 = -0.21976500E-01</td>
</tr>
<tr>
<td>14 = -0.32691750E-02</td>
<td>Q 14 = 0.85224920E-01</td>
</tr>
<tr>
<td>15 = -0.22928650E-01</td>
<td>Q 15 = 0.33139230E-01</td>
</tr>
<tr>
<td>16 = -0.11832640E-02</td>
<td>Q 16 = -0.12880190E+00</td>
</tr>
<tr>
<td>17 = -0.12521250E-01</td>
<td>Q 17 = -0.33948150E-01</td>
</tr>
<tr>
<td>18 = -0.28345480E-02</td>
<td>Q 18 = -0.20649250E+00</td>
</tr>
<tr>
<td>19 = 0.56341740E-02</td>
<td>Q 19 = -0.24238570E-01</td>
</tr>
<tr>
<td>20 = -0.26679000E-02</td>
<td>Q 20 = -0.34849320E+00</td>
</tr>
<tr>
<td>21 = -0.16361740E-02</td>
<td>Q 21 = 0.46913240E+00</td>
</tr>
<tr>
<td>22 = 0.16929130E-02</td>
<td>Q 22 = -0.27951660E+00</td>
</tr>
<tr>
<td>23 = -0.42695440E-03</td>
<td>Q 23 = 0.70494370E-01</td>
</tr>
</tbody>
</table>
APPENDIX C

LEVINSON-DURBIN ALGORITHM
Levinson-Durbin Algorithm

Given: \( n, R[0,n] \)

To Compute: \( \{ \mu_m, a_{k;m}, c_m : 1 \leq m, 1 \leq k \leq m \} \)

Initialization: \( \mu_0 = R(0) \)
\( a_{00} = 1 \)

Body: For \( m = 0 \) to \( n - 1 \)
\[ p_m = \sum_{k=0}^{m} R(m+1-k)a_{km} \]
\[ c_{m+1} = -p_m/\mu_m \]
\[ \mu_{m+1} = \mu_m (1 - c_{m+1}^2) \]
\[ a_{m+1;m} = 0 \]
For \( k = 0 \) to \( m + 1 \)
\[ a_{k;m+1} = a_{km} + c_{m+1} a_{m+1-k;m} \]
end loop on \( k \)
end loop on \( m \)
APPENDIX D
MATHCAD® PROGRAMS
Program: COEFSTORE

This program takes the impulse response coefficients and converts them to the analysis and synthesis filter impulse response coefficients, then saves them to disk.

\[ N := \text{READ}(\text{lcoef}) \]

\[ n := 0 \ldots N - 1 \quad M := 512 \quad Q := M - N \]

\[ h0 := \text{READ}(\text{coefficient}) \quad h1 := \text{READ}(\text{coefficient}) \quad \text{WRITE}(M) := M \]

\[ \text{WRITE}(h0) := h0 \quad n \]

\[ \text{WRITE}(h1) := h1 \quad n \]

\[ f0 := h0 \quad f1 := h1 \quad n \]

\[ f0 := h0 \quad f1 := h1 \quad n \]

\[ \text{WRITE}(f0) := f0 \quad \text{WRITE}(f1) := f1 \quad n \]

\[ m := 0 \ldots M - 1 \quad x := 0 \quad m \]

\[ q := 0 \ldots Q - 1 \quad y := 0 \quad q \]

\[ \text{APPEND}(h0) := y \quad \text{APPEND}(h1) := y \quad q \]

\[ \text{APPEND}(f0) := y \quad \text{APPEND}(f1) := y \quad q \]
Program: DATASTORE

This program stores the samples of a specified input function \( x_m \)

\[
N := 48 \quad n := 0 \ldots N - 1 \quad M := 512 \quad Q := M - N
\]

\[
M_m := 0 \ldots \frac{M}{2} - 1
\]

set the input as a unit sample pulse of magnitude 1

\[
x := 0
\]

\[
x := 1
\]

\[
WRITE(x_m) := x
\]
Program: DATASTORE

This program stores the samples of a specified input function \( x_m \)

\[
N := 48 \quad n := 0 \ldots N - 1 \quad M := 512 \quad Q := M - N
\]

\[
m := 0 \ldots 1 \quad 2
\]

set the input as a low frequency sine wave

\[
x := \sin \left[ \frac{m}{4 \cdot \pi \cdot 12} \right]_m^M
\]

\[
\text{WRITE}(x_m) := x_m
\]
Program: DATASTORE

This program stores the samples of a specified input function $x_m$

$$N := 48 \quad n := 0 \ldots N - 1 \quad M := 512 \quad Q := M - N$$

$$m := 0 \ldots \frac{M}{2} - 1$$

set the input as a high frequency sine wave

$$x := \sin \left( 4.\pi \cdot \frac{m}{M} \right)$$

WRITE($x_m$) := $x_m$
Program: FFTSAVE

This program takes the Fourier Transform of the impulse response coefficients and stores them in data files for processing.

\[
\begin{align*}
M & := \text{READ}(M) \\
m & := 0 \ldots M - 1 \\
t & := 0 \\
n & := 0 \ldots M - 1 \\
k & := 0 \ldots \frac{M}{2} \\
T & := \text{FFT}(t) \\
\text{WRITE}(F_{xm}) & := T \\
\text{WRITE}(F_{h0}) & := T \\
\text{WRITE}(F_{h1}) & := T \\
\text{WRITE}(F_{f0}) & := T \\
\text{WRITE}(F_{f1}) & := T
\end{align*}
\]
Program: ANALYSIS

This program takes the input data $x_m$ that has been transformed and applies it to the Analysis Filters. The Analysis filters output frequency response is plotted.

\[
M := \text{READ}(M) \quad m := 0 \ldots M - 1
\]
\[
L := \frac{M}{2} \quad l := 0 \ldots L \quad f := \frac{1}{M}
\]
\[
H_0 := \text{READ}(F_{h0}) + j \cdot \text{READ}(F_{h0})
\]
\[
H_1 := \text{READ}(F_{h1}) + j \cdot \text{READ}(F_{h1})
\]
\[
X_M := \text{READ}(F_{xm}) + j \cdot \text{READ}(F_{xm})
\]
\[
Y_0 := H_0 \cdot X_M \quad Y_1 := H_1 \cdot X_M
\]
Program: QMF

This program takes the input data $x_m$ that has been transformed and applies it to the Quadrature Mirror Filter Model. The output $x_0$ is plotted.

$$M := \text{READ}(M) \quad m := 0 \ldots M - 1$$
$$L := \frac{M}{2} \quad l := 0 \ldots L$$

$$H_0 := \text{READ}(F_{h0}) + j \text{READ}(F_{h0})$$
$$H_1 := \text{READ}(F_{h1}) + j \text{READ}(F_{h1})$$

$$F_0 := \text{READ}(F_{f0}) + j \text{READ}(F_{f0})$$
$$F_1 := \text{READ}(F_{f1}) + j \text{READ}(F_{f1})$$

$$X_M := \text{READ}(F_{xm}) + j \text{READ}(F_{xm})$$

$$Y_0 := H_0 \cdot X_M \quad Y_1 := H_1 \cdot X_M$$

$$x := \text{READ}(x_m)$$

$$Y_0 := F_0 \cdot Y_0 \quad Y_1 := F_1 \cdot Y_1$$

$$X_M := Y_0 + Y_1$$

$$x_0 := \text{ifft}(X_M)$$
Program: LATTICE

This program plots the frequency response for each block of a four stage lattice filter.

\[ M := 128 \]
\[ n := 0 \ldots N - 1 \]
\[ z := \exp\left[j \cdot 2 \cdot \frac{c}{M}\right]^n \]
\[ \begin{align*}
    P_0 & := \frac{1}{\sqrt{2}} \\
    Q_0 & := \frac{1}{\sqrt{2}} \\
    P_1(n) & := P_0 - z^n a_1 Q_0 \\
    Q_1(n) & := a_1 P_0 + z^n Q_0 \\
    P_3(n) & := P_1(n) - a_2 z^n Q_1(n) \\
    Q_3(n) & := a_2 P_1(n) + z^n Q_1(n) \\
    P_5(n) & := P_3(n) - a_3 z^n Q_3(n) \\
    Q_5(n) & := P_3(n) a_3 + z^n Q_3(n) \\
    P_7(n) & := P_5(n) - a_4 z^n Q_5(n) \\
    Q_7(n) & := P_5(n) a_4 + z^n Q_5(n)
\end{align*} \]
Program: MAGNITUDE

This program sets up the data points for the frequency response, does a Fast Fourier Transform then plots the frequency response in Db.

\[ N := 2048 \]

\[ n := 0 .. N - 1 \]

\[ h := 0 \]

\[ K := \text{READ}(lcoef) \]

\[ K := 2^K \]

\[ k := 0 .. K - 1 \]

\[ h := \text{READ}(coefp) \]

\[ u := \text{fft}(h) \]

\[ m := 0 .. \frac{N}{2} \]

\[ H := 20 \cdot \log_{10} \left| \begin{array}{c} \frac{u_m}{u_k} \end{array} \right| \]

\[ f := - \frac{m}{N} \]
REFERENCES


