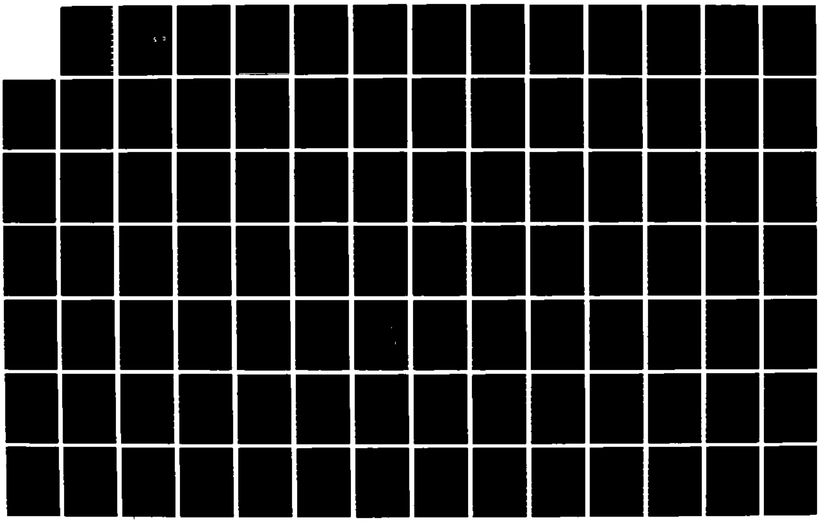
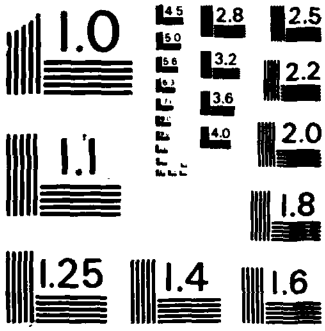
AD-A170 855 A ZERO EXTRACTION AND SEPARATION TECHNIQUE FOR SURFACE ACOUSTIC WAVE AND..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH K V LINDSAY 1986 1/2

UNCLASSIFIED AFIT/CI/NR-86-93T F/G 9/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(1)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/CI/NR 86-93T	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Zero Extraction and Separation Technique For Surface Acoustic Wave And Digital Signal Processing Fir Filter Implementation		5. TYPE OF REPORT & PERIOD COVERED THESIS/~~DISSERTATION~~
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Keith V. Lindsay		8. CONTRACT OR GRANT NUMBER(s)
PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: University of Central Florida		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433-6583		12. REPORT DATE 1986
		13. NUMBER OF PAGES 118
MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLAS
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC SELECTED AUG 1 3 1986 E

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-1

Lynn Wolaver GAUG 86

LYNN E. WOLAVER
Dean for Research and Professional Development
AFIT/NR

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED.

AD-A170 855

DTIC FILE COPY

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

86 8 13 ...

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

A ZERO EXTRACTION AND SEPARATION
TECHNIQUE FOR SURFACE ACOUSTIC WAVE AND
DIGITAL SIGNAL PROCESSING FIR
FILTER IMPLEMENTATION

BY

KEITH V. LINDSAY
B.S.E., University of Central Florida, 1984

THESIS

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

Spring Term
1986

ABSTRACT

Presented is a new method of separating the zeros of a Finite Impulse Response (FIR) filter producing an optimal digital filter or surface acoustic wave (SAW) design implementation. Overviews of zero extraction algorithms and of FIR filter design using the Remez Exchange algorithm are presented (McClellan et al. 1973).

The computer aided design (CAD) procedure presented allows the designer to specify the general filter characteristic which the Remez algorithm translates to FIR time domain coefficients. These coefficients are readily translated to the frequency (z) domain, producing an Nth order polynomial in z. The characteristic polynomial is factored to determine all roots or zeros using a three-stage factoring program presented by M.A. Jenkins (1975). The roots are optimally separated into two groups, each of which is recombined to form mutually exclusive functions. The two functions are then implemented as transducers of a SAW device or as a two-processor digital filter. The concept may be extended to more than two subgroups for multi-processor digital filter designs.

ACKNOWLEDGEMENTS

I would like to first thank my committee, chaired by Dr. Donald C. Malocha, for their patient support and guidance during this project. Mr. Samuel M. Richie originally suggested the basic concept of the thesis and was instrumental in its successful result. I also want to acknowledge the other members of the Solid State Devices Lab group for their encouragement and for the infectious enthusiasm they spread. Steve Wilkus of RF Monolithics provided considerable insight into this problem and suggested many references which were helpful in the literature search. Special thanks goes to two fellow Air Force officers and friends, Edward Raudenbush and James Walker, who helped encourage and enhance this effort.

It is rarely known from the outset of completing a thesis what results will be achieved and the effort required to achieve those results. This effort was greatly aided by various agencies around the University of Central Florida, such as the Computer Engineering VAX facility and personnel, the Library staff, and Dr. Al Pozefsky's State Technology Applications Center, to name a few. The task of assembling this report was greatly reduced by Sharon Darling, whose professional typing skills are readily apparent.

Most of all, I thank my wife, Kathy, for her tender understanding and encouragement through each difficult moment; and my two children, Kory and Kara, for the inspiration and desire to tackle the impossible.

TABLE OF CONTENTS

Appendix

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Numerous techniques exist for designing and implementing finite impulse response (FIR) filters. Many of these techniques can be traced to antenna array design methods popularized in the 1940s and 1950s (Balanis 1982). Among the more prominent antenna array design techniques are the methods by Fourier transform, Schelkunoff polynomial, Dolph-Chebyshev, Taylor line-source (Chebyshev Error) and Woodward. These have spawned many of the popular contemporary FIR design techniques such as the Remez exchange algorithm based on the Chebyshev Error method, popularized by McClellan, Parks and Rabiner (1973) and non-iterative eigenfunction synthesis design, introduced by Devries (1973) and similar to Woodward's method. Another FIR design approach employs a technique known as linear programming (Rabiner 1972a,b).

Each of these techniques will yield FIR transfer functions which may be readily implemented using a surface acoustic wave (SAW) device or a digital filter. The SAW filter, a two-transducer device, requires that the transfer function be split in some fashion between the transducers. The digital filter may be optionally implemented using two (or more) processors to increase throughput rate.

The conventional approach to implementing the SAW device is, basically, to construct one transducer such that it contains the entire FIR and to construct the other transducer such that it emulates a rect function. This imposes a requirement upon the first transducer that it be capable of handling all of the dynamics associated with the transfer function. Similarly, a single-processor digital filter implementation demands that the processor be able to handle a wide range of FIR coefficients, as well as all of them at once. These are not necessarily optimal implementations.

Some efforts to evenly split the transfer function between the two transducers of a SAW filter have been made by Morimoto, Kobayashi and Hibino (1980) and in work by Ruppel, Ehrmann-Falkenau, Stocker and Mader (1984, 1985). In both cases, these teams split the transfer function into groups of alternating zeros or roots of the transfer function about the unit circle. Any attempts to further optimize the filters were made at the expense of altering the overall frequency response in a process called compensation.

This thesis presents an approach to near optimally split the transfer function between the two transducers or processors without altering the overall frequency response. The approach seeks to minimize non-linear and finite wordlength error effects by reducing the required tap range for each transducer, or the required range of coefficients used in a fixed-point digital processor.

CHAPTER II

OBJECTIVE OF PROPOSED WORK

Optimal FIR Implementation Via Two-Transducer Design

Generally, the word "optimal" implies having attained a most favorable condition or degree. Many parameters must be considered during the design of a SAW or digital filter. Addressed in this thesis are those concerned primarily with filter order and coefficient dynamic range.

FIR Coefficients Via the Remez Algorithm

The first stage of the design is accomplished using the Remez exchange algorithm (Remez 1957) to generate a "best fit" Chebyshev polynomial to a set of frequency response specifications. The approximation, and subsequent conversion to an impulse response, is accomplished by the modified McClellan, Parks and Rabiner (1973) program presented in Appendix A. The program has been altered to permit the design of filters of up to an order of one-thousand. An initial guess of optimal filter order is obtained using a formulation presented by Vaidyanathan (1985), which states:

$$N_e = \frac{-10 \log_{10} \delta_1 \delta_2 - 13}{14.6 \Delta f} \qquad (2.1)$$

where:

$\Delta f = (\omega_s - \omega_p)/2\pi$

ω_s = stop-band edge frequency

ω_p = pass-band edge frequency

δ_1 = pass-band tolerances

δ_2 = stop-band tolerances

This N_e provides a starting point for the filter order in the program. The program then iterates, increasing N each time, until the specifications are met.

Once the FIR coefficients are found by the modified McClellan, Parks and Rabiner (1973) program, they may be easily arranged as the coefficients of a z-domain polynomial, i.e., the z-transform of:

$$h(n) = h(t - nT) = \begin{cases} a_n & n = 0, 1, 2, \ldots, N \\ 0 & \text{otherwise} \end{cases} \qquad (2.2)$$

is

$$H(z) = \sum_{n=0}^{N} h(n) z^{-n} \qquad (2.3)$$

where:

a_n = the coefficients generated by the program

At this point, the impulse response could be implemented as a single SAW transducer or as a single processor digital filter. In order to split up the response between two transducers (or processors), H(z) can be expressed as:

$$H(z) = \frac{\{\sum_{k=0}^{N} a_k z^{(N-k)}\}}{z^N} = H_1(z)\ H_2(z) \qquad (2.4)$$

This equation shows all of the poles of a finite impulse response filter to be at $z = 0$. All of the filter's zeros may be found by factoring the numerator. A judicious separation of the zeros (and poles) can then be assigned to $H_1(z)$ and $H_2(z)$, the two transducers of the SAW filter. In a similar fashion, the transfer function could be split into $H_{1...N}(z)$ for a DSP filter to increase speed and dynamic range.

Obtaining the Zeros of the Characteristic

This is the most difficult phase of the optimization. The factoring of high order polynomials was the subject of considerable effort by mathematicians during the mid-seventies. Of the many techniques surveyed, the Jenkins and Traub (1970) algorithm, and program by Jenkins (1975), appeared to be the best choice. This algorithm employs a three-stage process to determine the roots of an Nth order real polynomial. It is globally convergent and does so very rapidly. The program is extremely well written and is quite elaborate, to the point of compensating for specific machine accuracy limitations.

The Jenkins (1975) program factors the H(z) numerator polynomial and returns the real and imaginary portions of the roots of H(z). The complex roots will always appear in one of two possible ways. A set of roots may appear as complex conjugate pairs (quadratic factors), each with a magnitude of one corresponding to the $|z| = 1$ unit circle. These roots will always correspond to the stopband zeros of a filter design. Another form in which they may appear is as a set of two complex conjugate pairs arranged symmetrically about the unit circle so as to satisfy the condition that:

$$z_1 z_2^* = 1 \tag{2.5}$$

These zeros correspond to the passband zeros of the filter. A diagram best illustrates these concepts (see Figure 1). Real roots may occur on the unit circle or in a manner similar to the passband zero case. Summarizing the above, zeros of $H(z)$ will occur as first, second and fourth order factors.

Selection of Zeros and Subsequent Reconstitution

Once the zeros of the transfer function have been determined, they must be separated into sub-groups and recombined. A simple algorithm which generates all possible combinations of N roots taken K at a time is used to separate the roots and form the sub-groups. A polynomial is constructed from each group by multiplying the roots within its group and the split design is rated as to the desirability of the design.

The criteria used in the selection process employed here seeks to maximize the average tap or coefficient value, while minimizing the range and variance of those same values. These qualities are

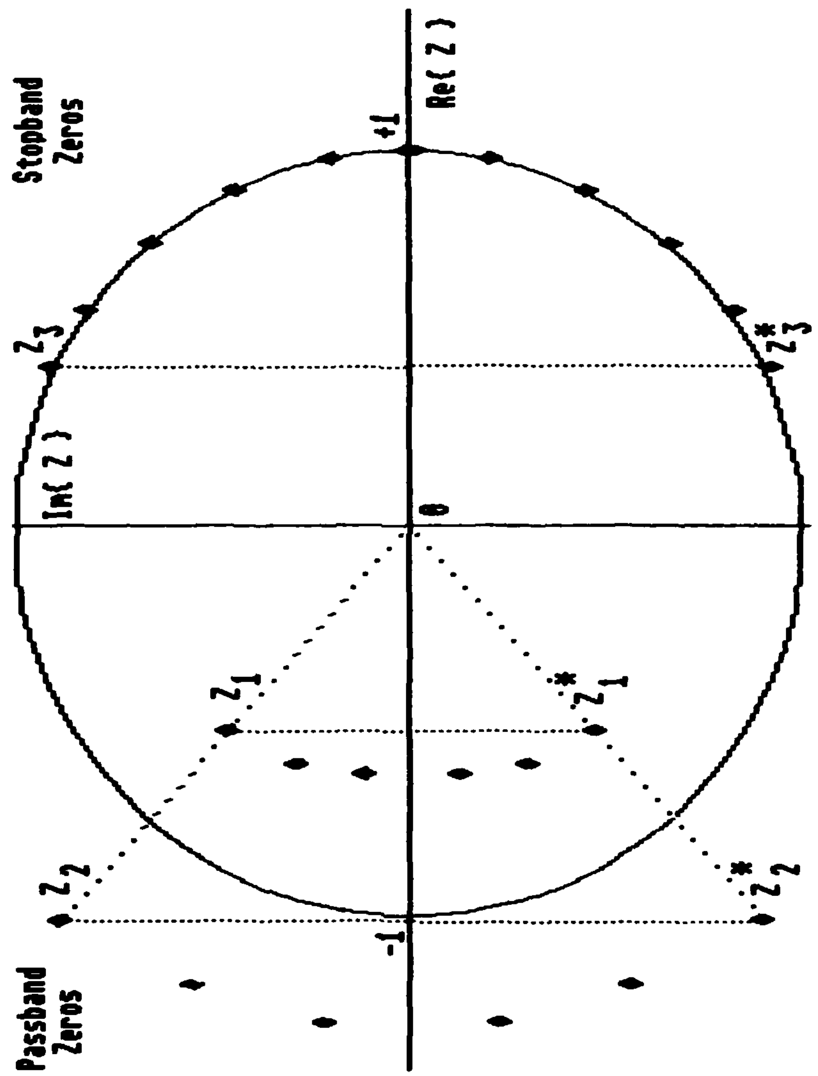


Figure 1. Typical Filter Zero Locations.

desirable in SAW devices since we usually desire a maximum finger overlap and want to avoid, as much as possible, large numbers of very small tap weights which may increase diffraction effects. In the case of digital filters, these problems translate to finite word length problems and device dynamic range.

In order to evaluate the relative merits of one combination over another, the following figure of merit is proposed:

$$\text{Design FOM} = \frac{\bar{x}}{R_x \, \sigma_x^2} \tag{2.6}$$

where:

$\bar{x}$ = *average of the tap weights of both transducers combined*

R_x = the range of the coefficients

σ_x^2 = the variance of the coefficients

The ratio yields a figure of merit used to rate a given design. Obviously, this concept could be extended to more than two sub-transfer functions for the digital filter case.

This thesis proposes to study low order filters using this separation/reconstitution technique and to apply detected trends, if any, in a general sense.

CHAPTER III

FIR FILTER DESIGN

An excellent review of finite impulse response filter theory is presented by Lawrence R. Rabiner and Bernard Gold (1975) and by Rabiner, McClellan and Parks (1975). This review provides a theoretical background for implementing the Weighted Chebyshev Approximation filter design technique via the Remez Exchange Algorithm (Remez 1957). That presentation draws upon the work of Parks and McClellan (1973), who devised a general computer program incorporating the above. Their program was used in this thesis to provide the FIR transfer functions. An overview of the theory leading from FIR theory to a brief program description is presented. The following is adopted from Rabiner and Gold (1975).

FIR Filter Frequency Response Overview

A finite impulse response describes a system which can be modeled by a difference equation in the form:

$$y(n) = \sum_{r=0}^{M} \left(\frac{b_r}{a_o}\right) x(n - r) \tag{3.1}$$

Since the system output is the convolution of the system input, x(n), with the system impulse response, h(n), the impulse response can be readily seen to be:

$$h(n) = \begin{cases} \dfrac{b_n}{a_0}, & n = 0, 1, 2, \ldots, M \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

The above equation describes the discrete time domain coefficients of the system FIR. This same response may be described in the frequency domain by taking the Fourier transform of h(n) to obtain $H(e^{j\omega})$:

$$H(e^{j\omega}) = \sum_{k=-\infty}^{\infty} h(k)\, e^{-j\omega k} \tag{3.3}$$

Since h(n) is finite in length with respect to time, $H(e^{j\omega})$ must be infinite with respect to frequency. However, for discrete, sampled systems, $H(e^{j\omega})$ is periodic with respect to the sampling frequency, i.e.,:

$$H(e^{j\omega}) = H[e^{j(\omega+2\pi m)}] \qquad m = 0, \pm 1, \pm 2, \ldots \tag{3.4}$$

which is periodic in frequency with a period of 2π. This fact allows us to restrict our requirement to define $H(e^{j\omega})$ in practical filtering applications in terms of the sampling frequency, consisting of N samples over a period equaling the length of the time domain impulse response (without any augmenting zeros). It also allows us to state that:

$$H(e^{j\omega}) = \sum_{k=0}^{N-1} h(k)\, e^{-j\omega k} \tag{3.5}$$

The function, $H(e^{j\omega})$, can be described in terms of its magnitude and phase as:

$$H(e^{j\omega}) = \pm |H(e^{j\omega})| \quad e^{j\theta(\omega)} \qquad (3.6)$$

or

$$H(e^{j\omega}) = \hat{H} (e^{j\omega}) \, e^{j\theta(\omega)} \qquad (3.7)$$

where:

$\hat{H} (e^{j\omega})$ = a real function

$\theta(\omega)$ = constrained to describe a linear phase characteristic, i.e.,:

$$\theta(\omega) = - \alpha\omega \qquad -\pi \leq \omega < \pi \qquad (3.8)$$

with constant phase delay implied by the constant, $- \alpha$. The function can be written in trigonometric form as:

$$\pm |H(e^{j\omega})| \, \{\cos (\alpha\omega) - j \sin (\alpha\omega)\} \qquad (3.9)$$

In order to find α, equate the real and imaginary parts. Then, we may describe $\cos (\alpha\omega)$ and $\sin (\alpha\omega)$ as:

$$\pm |H(e^{j\omega})| \cos (\alpha\omega) = \sum_{n=0}^{N-1} h(n) \cos (\omega n) \qquad (3.10)$$

and

$$\pm\ |H(e^{j\omega})|\ \sin(\alpha\omega) = \sum_{n=0}^{N-1} h(n) \sin(\omega n) \tag{3.11}$$

and set up the ratio:

$$\frac{\sin(\alpha\omega)}{\cos(\alpha\omega)} = \tan(\alpha\omega) = \frac{\sum_{n=0}^{N-1} h(n) \sin(\omega n)}{\sum_{n=0}^{N-1} h(n) \cos(\omega n)} \tag{3.12}$$

Cross multiplying:

$$\sum_{n=0}^{N-1} h(n) \sin(\alpha\omega) \cos(n\omega) - \sum_{n=0}^{N-1} h(n) \cos(\alpha\omega) \sin(n\omega) = 0 \tag{3.13}$$

Using the trig identity:

$$\sin(u-v) = (\sin u)(\cos v) - (\cos u)(\sin v) \tag{3.14}$$

yields:

$$\sum_{n=0}^{N-1} h(n) \sin[(\alpha-n)\omega] = 0 \tag{3.15}$$

Equation (3.15) is in the form of a Fourier series. For equation (3.15) to be valid for an odd symmetrical series (see Figure 2), α and $h(n)$ must be:

$$\alpha = \frac{N-1}{2} \tag{3.16}$$

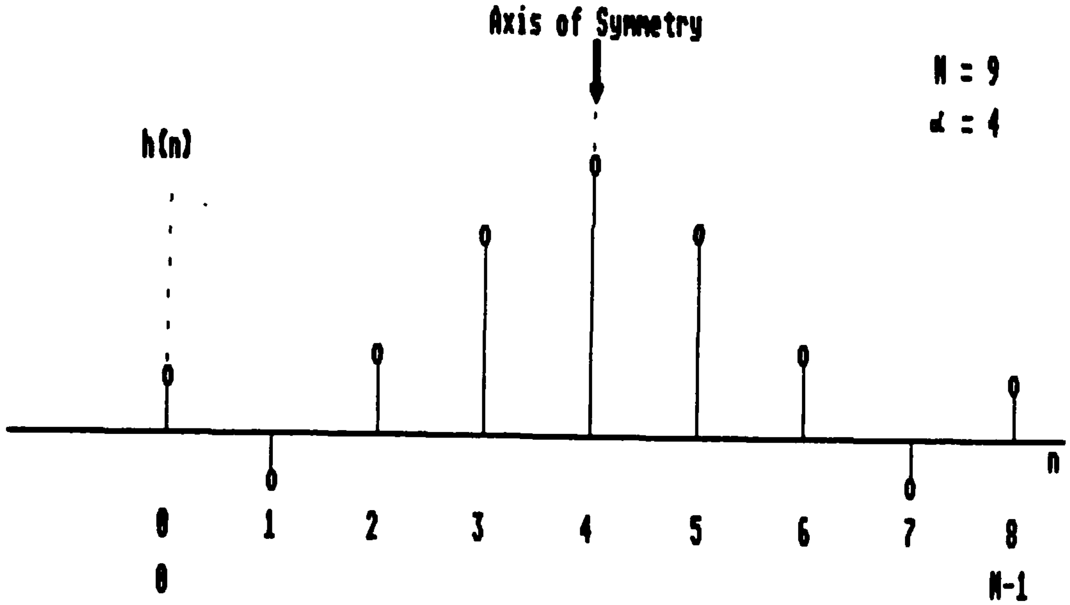


Figure 2. Odd Symmetrical FIR Series.

and:

$$h(n) = h(N-1-n) \qquad 0 \leq n \leq N-1 \tag{3.17}$$

In the case of an even, symmetrical series (see Figure 3), α will not be an integer. The use of the fractional delay obtained here is of primary significance when designing differentiators and Hilbert transformers. These are not discussed here, but the reader is referred to Rabiner and Gold (1975) for an in-depth discussion. These values for α and h(n) hold for constant group delay and constant phase filters. If constant phase delay (phase divided by frequency) is not required, i.e.,:

$$H(e^{j\omega}) = \pm |H(e^{j\omega})| \; e^{j\omega(\frac{\beta}{\omega} - \alpha)} = \pm |H(e^{j\omega})| \; e^{j(\beta-\alpha\omega)} \tag{3.18}$$

where the phase delay is given by $-\frac{\beta}{\omega} + \alpha$, then similar development (Rabiner and Gold 1975) for an odd, anti-symmetric series (see Figure 4) will lead to the result that:

$$\alpha = \frac{N-1}{2} \tag{3.19a}$$

$$\beta = \pm \frac{\pi}{2} \tag{3.19b}$$

and

$$h(n) = -h(N-1-n) \qquad 0 \leq n \leq N-1 \tag{3.19c}$$

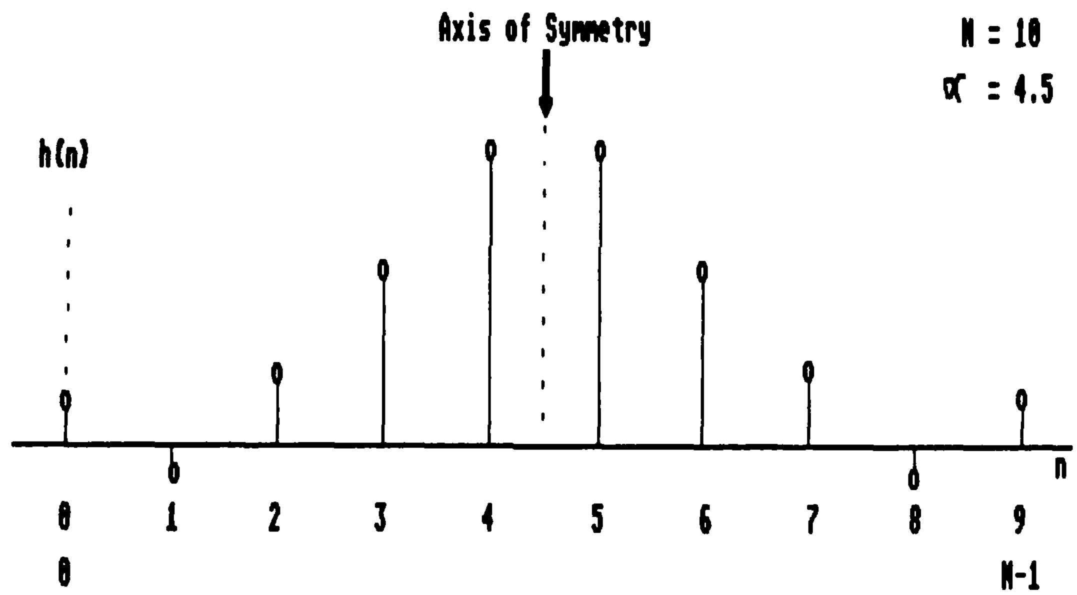


Figure 3. Even Symmetrical FIR Series.

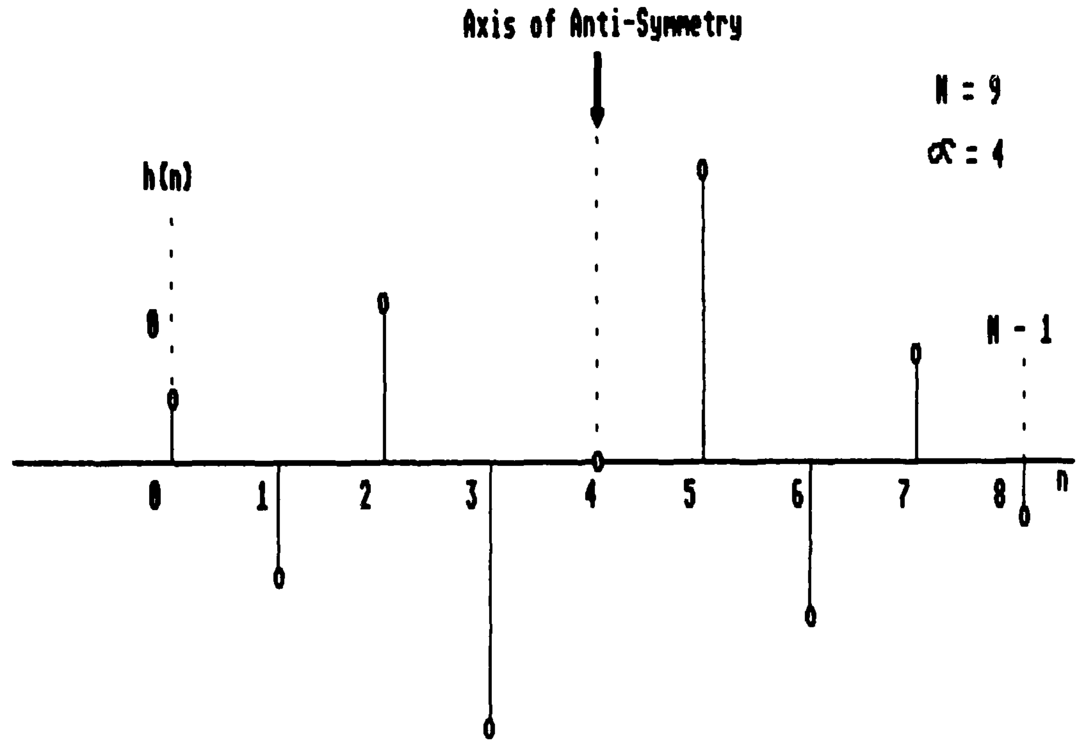


Figure 4. Odd Anti-Symmetrical FIR Series.

Again, the case of an even, anti-symmetric series (see Figure 5) is of primary interest in designing differentiators and Hilbert transformers. Equations (3.16) through (3.19) suggest four general classes that might characterize a linear phase finite impulse response filter:

1. Symmetrical impulse response, N odd
2. Symmetrical impulse response, N even
3. Anti-symmetrical impulse response, N odd
4. Anti-symmetrical impulse response, N even

It is now possible to describe $H(e^{j\omega})$ to account for these possibilities in the general relationship:

$$H(e^{j\omega}) = \hat{H}(e^{j\omega})\, e^{j(\beta-\alpha\omega)} \tag{3.20}$$

Rabiner and Gold (1975) next develop equations to define $H(e^{j\omega})$ in terms of $\hat{H}(e^{j\omega})$ for each of the above cases. The results of these developments are summarized as follows:

Case 1: Symmetrical impulse response, N odd

$$\hat{H}(e^{j\omega}) = \sum_{n=0}^{(N-1)/2} a(n) \cos(\omega n) \tag{3.21}$$

with

$$a(0) = h[(N-1)/2], \text{ and}$$

$$a(n) = 2h[\frac{(N-1)}{2} - n] \quad \text{for } n = 1, 2, \ldots, (N-1)/2$$

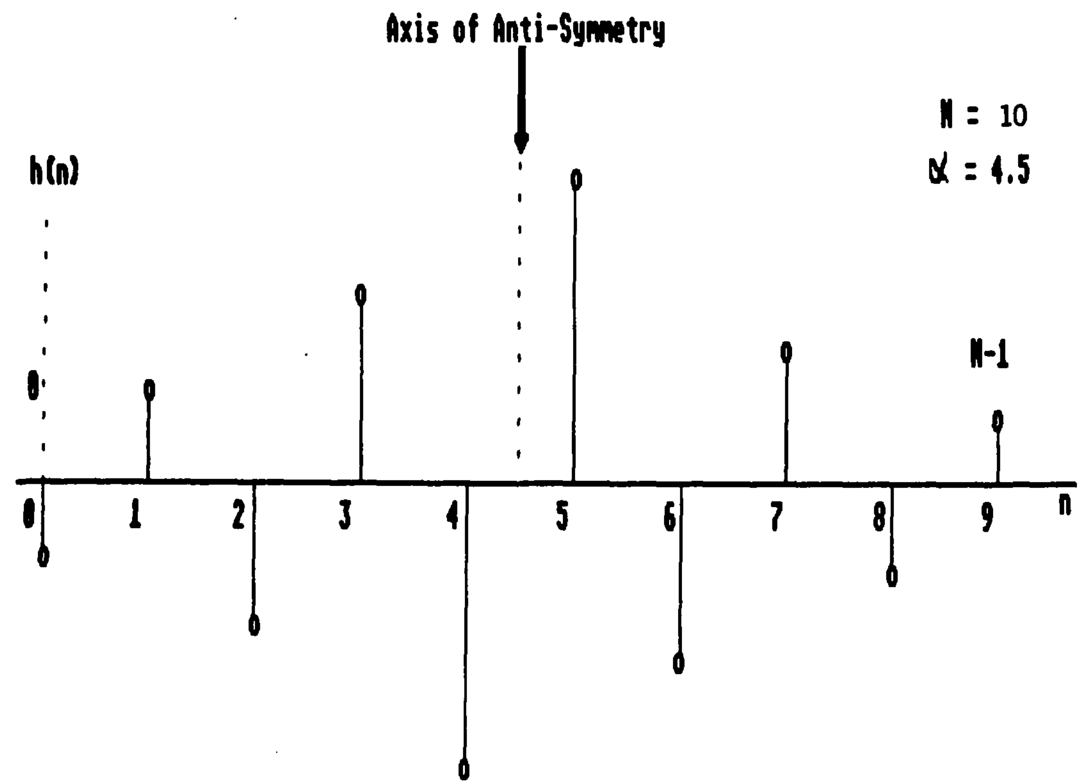


Figure 5. Even Anti-Symmetrical FIR Series.

which yields:

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2} \sum_{n=0}^{(N-1)/2} a(n) \cos(\omega n) \qquad (3.23)$$

Case 2: Symmetrical impulse response, N even

$$\hat{H}(e^{j\omega}) = \sum_{n=1}^{N/2} b(n) \cos[\omega(n-0.5)] \qquad (3.24)$$

with

$$b(n) = 2h(N/2 - n), \qquad n = 1, 2, \ldots, N/2 \qquad (3.25)$$

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2} \sum_{n=1}^{N/2} b(n) \cos[\omega(n-0.5)] \qquad (3.26)$$

Case 3: Anti-symmetrical impulse response, N odd

$$\hat{H}(e^{j\omega}) = \sum_{n=1}^{(N-1)/2} c(n) \sin(\omega n) \qquad (3.27)$$

with

$$c(n) = 2h[(N-1)/2 - n], \qquad n = 1, 2, \ldots, (N-1)/2 \qquad (3.28)$$

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2}\, e^{j\pi/2} \sum_{n=1}^{(N-1)/2} c(n) \sin(\omega n) \qquad (3.29)$$

Case 4: Anti-symmetrical impulse response, N even

$$\hat{H}(e^{j\omega}) = \sum_{n=1}^{N/2} d(n) \sin [\omega(n-0.5)] \tag{3.30}$$

with

$$d(n) = 2h(N/2 - n), \qquad n = 1, 2, \ldots, N/2 \tag{3.31}$$

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2}\, e^{j\pi/2} \sum_{n=1}^{N/2} d(n) \sin [\omega(n-0.5)] \tag{3.32}$$

Weighted Chebyshev Approximation

The frequency response of the desired system is completely described by $H(e^{j\omega})$. From the development in the first section of this chapter, it is obvious that this description for each case can be considered as a series of sine or cosine functions. These series can be easily related to Chebyshev polynomials.

The Chebyshev polynomial represents an expansion of cos (mu) for any value of m. We know that any real function can be represented as a sum of sinusoids. These sinusoids are of the form cos (mu), with m indicating the highest harmonic required to reconstruct the original function. Of course, some functions require that m approach ∞. Within given limits, however, it is possible to represent a desired frequency response curve as a sum of Chebyshev polynomials of finite length m. The Chebyshev polynomial expansions take the following forms:

$$
\begin{array}{llcl}
\underline{m} & \underline{\cos mu} & & \\
0 & 1 & = & 1 \\
1 & \cos u & = & \cos u \\
2 & \cos 2u & = & 2\cos^2 u - 1 \\
3 & \cos 3u & = & 4\cos^3 u - 3\cos u
\end{array}
\tag{3.33}
$$

Letting $z = \cos(u)$, or $u = \cos^{-1}(z)$, then:

$$
\begin{array}{llc}
\underline{m} & \underline{\cos mu} & \begin{array}{c}\text{Chebyshev}\\ \underline{\text{Designation}}\end{array} \\
0 & 1 & T_0(z) \\
1 & z & T_1(z) \\
2 & 2z^2 - 1 & T_2(z) \\
3 & 4z^3 - 3z & T_3(z)
\end{array}
\tag{3.34}
$$

A recursive relationship emerges:

$$T_m(z) = \cos[m \cos^{-1}(z)] = \cos(mu), \quad -1 \le z \le 1 \tag{3.35}$$

In essence, we are using a sum of Chebyshev polynomials to curve-fit to the desired frequency response from zero to one-half the sampling frequency. Given the four cases discussed at the end of the first section of this chapter, and using Chebyshev polynomials to represent the series, the problem defaults to determining the scaling of the coefficients with respect to frequency. This process is called "weighting" the approximation and is discussed in depth by Rabiner and Gold (1975). It is reiterated here briefly.

For the four cases described in the first section of this chapter, a general expression can be written to define $H(e^{j\omega})$ as:

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2} e^{j(\pi/2)L} \hat{H}(e^{j\omega}) \qquad (3.36)$$

The exponent L will take on a value of either 0 or 1, depending upon the case considered. Now, a table can be constructed which shows values for L and the form of $\hat{H}(e^{j\omega})$ for the appropriate case of symmetry and N (see Table 1). The previous discussion of the form of the Chebyshev polynomial would suggest that the expressions for $\hat{H}(e^{j\omega})$ may be converted to summations involving cosines (as opposed to sines) using ordinary trigonometric identities. Once this is done, Table 1 can be rewritten in terms of functions which are fixed functions of ω, which will be referred to as $Q(e^{j\omega})$, and as functions of the cosine series, which will be referred to as $P(e^{j\omega})$ (see Table 2). For cases 2 through 4, $Q(e^{j\omega})$ is constrained to be zero at either $\omega = 0$ or $\omega = \pi$, or both.

Now, it is possible to set up a relationship between the desired response at given frequencies to within a prescribed accuracy. To do so, let $D(e^{j\omega})$ represent the desired response of the filter and let $W(e^{j\omega})$ represent the weighting on the allowable error as a function of frequency regions or bands (i.e., the ratio of the

TABLE 1

DEFINITION OF L FOR THE FOUR FILTER CASES

	L	$H(e^{j\omega})$
Case 1: Symmetrical impulse response, N odd	0	$\hat{H}(e^{j\omega}) = \sum_{n=0}^{(N-1)/2} a(n) \cos(\omega n)$
Case 2: Symmetrical impulse response, N even	0	$\hat{H}(e^{j\omega}) = \sum_{n=1}^{N/2} b(n) \cos[\omega(n-0.5)]$
Case 3: Anti-symmetrical impulse response, N odd	1	$\hat{H}(e^{j\omega}) = \sum_{n=1}^{(N-1)/2} c(n) \sin(\omega n)$
Case 4: Anti-symmetrical impulse response, N even	1	$\hat{H}(e^{j\omega}) = \sum_{n=1}^{N/2} d(n) \sin[\omega(n-0.5)]$

TABLE 2

$Q(e^{j\omega})$ AND $P(e^{j\omega})$ DEFINED FOR THE FOUR FILTER CASES

	$Q(e^{j\omega})$	$P(e^{j\omega})$
Case 1: Symmetrical impulse response, N odd	1	$\hat{H}(e^{j\omega}) = \sum_{n=0}^{(N-1)/2} a(n) \cos(\omega n)$
Case 2: Symmetrical impulse response, N even	$\cos(\frac{\omega}{2})$	$\hat{H}(e^{j\omega}) = \sum_{n=0}^{(N/2)-1} b(n) \cos(\omega n)$
Case 3: Anti-symmetrical impulse response, N odd	$\sin(\omega)$	$\hat{H}(e^{j\omega}) = \sum_{n=0}^{(N-3)/2} c(n) \cos(\omega n)$
Case 4: Anti-symmetrical impulse response, N even	$\sin(\frac{\omega}{2})$	$\hat{H}(e^{j\omega}) = \sum_{n=0}^{(N/2)-1} d(n) \cos(\omega n)$

stopband ripple to the passband ripple). With these functions, the error for a given approximation can be calculated as:

$$E(e^{j\omega}) = W(e^{j\omega}) \, [D(e^{j\omega}) - \hat{H}(e^{j\omega})] \qquad (3.37)$$

with $\hat{H}(e^{j\omega})$ being the trial design. $\hat{H}(e^{j\omega})$ can be separated into its two symbolic parts to yield:

$$E(e^{j\omega}) = W(e^{j\omega}) \, [D(e^{j\omega}) - P(e^{j\omega}) \, Q(e^{j\omega})] \qquad (3.38)$$

$Q(e^{j\omega})$ may be factored out of the quantity in parentheses since it is a fixed function of frequency. This yields:

$$E(e^{j\omega}) = W(e^{j\omega}) \, Q(e^{j\omega}) \, [D(e^{j\omega})/Q(e^{j\omega}) - P(e^{j\omega})] \qquad (3.39)$$

Defining $[W(e^{j\omega}) \, Q(e^{j\omega})]$ as $\hat{W}(e^{j\omega})$, and $[D(e^{j\omega})/Q(e^{j\omega})]$ as $\hat{D}(e^{j\omega})$, equation (3.39) may be rewritten as:

$$E(e^{j\omega}) = \hat{W}(e^{j\omega}) \, [\hat{D}(e^{j\omega}) - P(e^{j\omega})] \qquad (3.40)$$

The problem now defaults to finding the values for the coefficients of the Chebyshev polynomials $[P(e^{j\omega})]$ such that the maximum error over each specified frequency band is minimized. To accomplish this, the Alternation Theorem is used. It states (Rabiner and Gold 1975):

Theorem: If $P(e^{j\omega})$ is a linear combination of r cosine functions, i.e.,:

$$P(e^{j\omega}) = \sum_{n=0}^{r-1} \alpha(n) \cos(n\omega) \qquad (3.41)$$

then a necessary and sufficient condition that $P(e^{j\omega})$ be the unique, best weighted Chebyshev approximation to a continuous function $\hat{D}(e^{j\omega})$ on A, a compact subset of $(0,\pi)$, is that the weighted error function $E(e^{j\omega})$ exhibit at least (r+1) extremal frequencies in A; i.e., there must exist (r+1) points ω_i in A such that $\omega_1 < \omega_2 < \ldots < \omega_{r+1}$ and such that $E(e^{j\omega_i}) = -E(e^{j\omega_{i+1}})$, i = 1, 2, ..., r, and $|E(e^{j\omega_i})| = \max [E(e^{j\omega})]$ for all ω in A.

Rabiner and Gold (1975) show that for the four cases of filter design presented, that the number of extremal frequencies in $\hat{H}(e^{j\omega})$ obey the following constraints:

$$\begin{array}{ll} \text{Case 1:} & N_e \leq (N+1)/2 \\ \text{Case 2:} & N_e \leq N/2 \\ \text{Case 3:} & N_e \leq (N-1)/2 \\ \text{Case 4:} & N_e \leq N/2 \end{array} \qquad (3.42)$$

The extremal frequencies, or extrema, are divided up between the stop and passbands of the filter and they describe the peaks and troughs of the Chebyshev approximation. A diagram best describes the relationship of the extremal frequencies and the shape of the Chebyshev approximation waveform (see Figure 6).

There are several ways in which to obtain the extremal frequencies of $\hat{H}(e^{j\omega})$. The first method, described briefly, was originally proposed by Herrmann and Schuessler. The method capitalizes on the fact that a local maxima $(+\delta)$ or a local minima

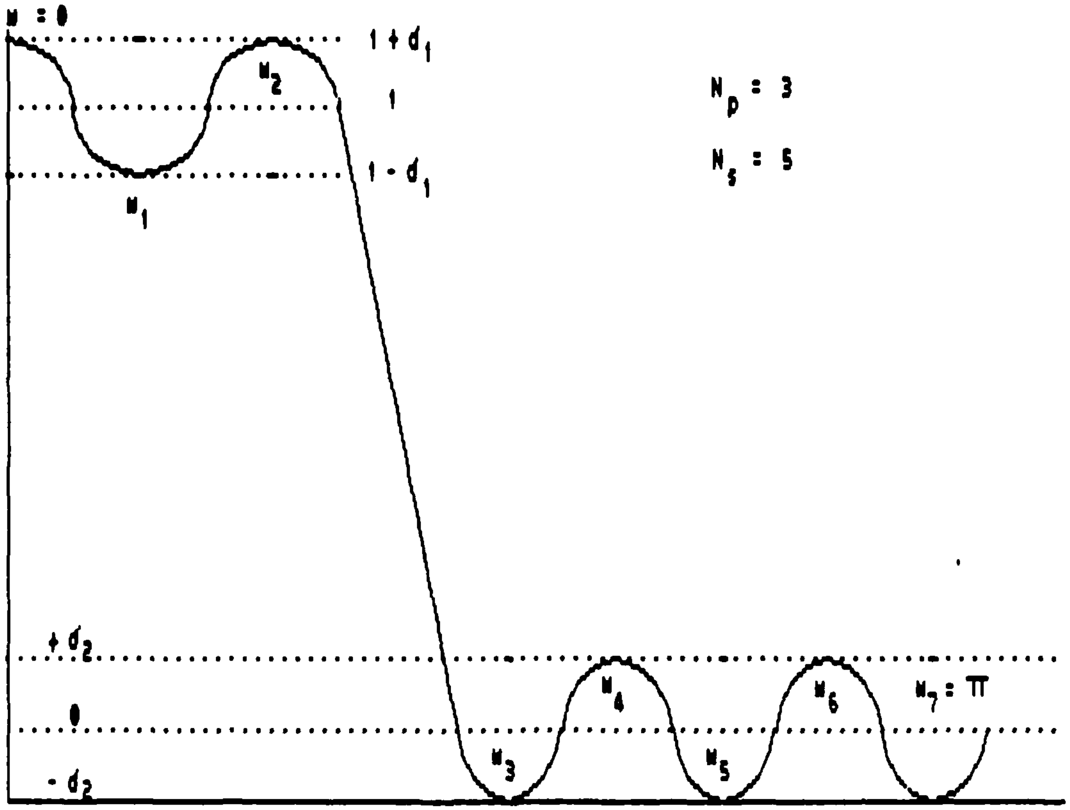


Figure 6. Chebyshev Approximation Extremal Frequencies.

$(-\delta)$ occurs in the region of an extrema, and that the derivative is zero at that point. Two equations in N_e with two N_e unknowns (N_e impulse response coefficients and N_e frequencies where $\hat{H}(e^{j\omega})$ obtains an extremal value) are:

$$\hat{H}(e^{j\omega_i}) = \pm \frac{\delta}{W(e^{j\omega_i})} + D(e^{j\omega_i}) \qquad i = 1, 2, \ldots, N_e \qquad (3.43)$$

and

$$\left. \frac{d\,[\hat{H}(e^{j\omega})]}{d\omega} \right|_{\omega = \omega_i} = 0 \qquad i = 1, 2, \ldots, N_e \qquad (3.44)$$

where $E(e^{j\omega_i}) = \pm\,\delta$, and these are solved iteratively for values of N_e. This procedure works well for filters with an order about 60 or less.

Another method used is one devised by Hofstetter, Oppenheim and Siegel which is called the Polynomial Interpolation Solution by Rabiner and Gold (1975). The basic idea behind this algorithm is that an initial guess of the extremal frequencies is made and $\hat{H}(e^{j\omega})$ is evaluated at these points. The algorithm then searches for the actual extrema found during that trial and iterates again, this time using the newly found extrema. Eventually, the process converges to the minimum ripple attainable for a given N_e. Very large order filters can be designed by this method. The Polynomial Interpolation Solution technique is very similar to the last technique to be described, the Remez Exchange Algorithm.

The Remez Exchange Algorithm

We have seen that the goal of obtaining the desired response $\hat{D}(e^{j\omega})$ is met by obtaining the approximating function $P(e^{j\omega})$ which best minimizes the weighted error function $E(e^{j\omega})$. The Remez Exchange Algorithm accomplishes this by using a dense grid of frequency points to find the extremal frequencies. An initial guess as to the location of the (r+1) frequencies is made, similar to the Polynomial Interpolation Solution method. Then, the error function is forced to have a value of $\pm\,\delta$. The signs alternate, since the extrema are expected to alternate above and below the indicated level by δ in the final design. These constraints generate the separate error function $[E(e^{j\omega})]$ equation for each extremal frequency, given from equation (3.40) as:

$$\hat{W}(e^{j\omega_k})\,[\hat{D}(e^{j\omega_k}) - P(e^{j\omega_k})] = (-1)^k\,\delta \qquad k = 0, 1, \ldots, r \qquad (3.45)$$

which generates an (r+1) x (r+1) matrix of equations to solve. Remez (1957) found an alternative closed-form solution (appropriately modified for the current variable set) to be:

$$\delta = \frac{a_o\hat{D}(e^{j\omega_o}) + a_1\hat{D}(e^{j\omega_1}) + \ldots + a_r\hat{D}(e^{j\omega_r})}{a_o/\hat{W}(e^{j\omega_o}) - a_1/\hat{W}(e^{j\omega_1}) + \ldots + (-1)^r\, a_r/\hat{W}(e^{j\omega_r})} \qquad (3.46)$$

where:

$$a_k = \prod_{\substack{i=0 \\ i\neq k}}^{r} \frac{1}{(x_k - x_i)} \qquad (3.47)$$

and

$$x_i = \cos \omega_i \tag{3.48}$$

At this point, the optimum δ for a given set of extremal frequencies is known. The next step is to form the approximating function $P(e^{j\omega})$ along the r extrema points by using the barycentric form of the Lagrange interpolation formula:

$$P(e^{j\omega}) = \frac{\sum_{k=0}^{r-1} \left(\frac{\beta_k}{x - x_k}\right) C_k}{\sum_{k=0}^{r-1} \frac{\beta_k}{(x - x_k)}} \tag{3.49}$$

where:

$$\beta_k = \prod_{\substack{i=0 \\ i \neq k}}^{r-1} \frac{1}{(x_k - x_i)} \tag{3.50}$$

and

$$C_k = \hat{D}(e^{j\omega_k}) - (-1)^k \frac{\delta}{\hat{\omega}(e^{j\omega_k})} \quad k = 0, 1, \ldots, r-1 \tag{3.51}$$

$$x_i = \cos \omega_i$$

$$x_k = \cos \omega_k \tag{3.52}$$

$$x = \cos \omega$$

Once the approximating function $P(e^{j\omega})$ has been formed, it is possible to evaluate $E(e^{j\omega})$ along a dense set of frequencies which are equally spaced along the frequency axis from zero to one-half the sample frequency. If:

$$|E(e^{j\omega})| \leq \delta \tag{3.53}$$

then an optimal approximation to the desired frequency response has been found. If the weighted error function exceeds δ, then a new set of (r+1) extremal frequencies is chosen by selecting the peaks of the error curve. This process quickly forces δ to converge to its maximum value for a given number of extremal frequencies. If there are more than (r+1) extrema in $E(e^{j\omega})$, then the new number of extrema is retained and used in the next iteration of the process.

The final impulse response coefficients are obtained by performing a 2^M point Inverse Discrete Fourier Transform on $P(e^{j\omega})$, where $2^M \geq N$. Note that this N is the filter order plus 1.

CHAPTER IV

ZERO EXTRACTION TECHNIQUES

The problem of extracting the zeros of a polynomial turns out to be far from simple, sparking the interest of mathematicians and scientists for centuries. With the advent of the digital computer, the factoring of high order polynomials has become possible, though not entirely without grief. Some of the more prominent approaches and associated problems are briefly discussed here.

Polynomial Theory

A polynomial in z is an equation which takes the form:

$$a_N z^N + a_{N-1} z^{N-1} + \ldots + a_2 z^2 + a_1 z + a_0 \qquad (4.1)$$

or, alternatively,

$$\sum_{n=0}^{N} a_n z^n \qquad (4.2)$$

where the coefficients a_N, a_{N-1}, ..., a_0 are real numbered constants. This form of equation is readily identified with the equation describing the finite impulse response filter z-domain representation. This polynomial can be expressed also as a product of its roots or zeros as:

$$\prod_{n=1}^{N} (z - z_n) \tag{4.3}$$

Notice that for a polynomial with N roots, there are N product-form terms and N+1 summation-form terms. This can cause some confusion at times. For example, the McClellan, Parks and Rabiner (1973) program discussed in the previous chapter displays a filter order of N when, in fact, N coefficients are actually meant.

Zero Characteristics of FIR Filters

When described by a polynomial in the $z = e^{j\omega}$ plane, FIR filter zeros plotted in the z-plane always have a distinct appearance. All of the stopband zeros will occur exactly on the unit circle and will always occur as complex conjugate pairs, unless they are real. The complex passband zeros will always occur in sets of four, one inside the unit circle, another outside such that the magnitude of one multiplied by the other will equal exactly one. This pair also has corresponding complex conjugates, hence the set of four.

Due to the nature of the Chebyshev polynomial approximation of the FIR frequency response, there are no repeating zeros or multiple roots to contend with. However, this does pose problems in other factoring situations and is discussed briefly.

Factoring Methods

Several unique approaches to zero extraction exist. The first was by none other than Sir Issac Newton (1642-1727). Since that time, other algorithms by Bairstow, Lin, Muller and Birge-Vieta have arrived (Ralston and Wilf 1960). These algorithms have the relative liability of not being able to assure convergence for any initial guess of a root. Other methods which virtually assure convergence within a class of problems are methods of Lehmer, Graeffe and Bernoulli. The Bisection method is probably the most crude method of root-finding, relying upon a purely iterative process of testing discrete points in the z-plane until the roots are found. The latter four cases have the disadvantage of slow convergence. There exist several matrix-oriented computer program packages, such as EISPACK (Smith et al. 1976), which are designed to find the eigenvalues of a matrix, another means of finding the zeros. However, these programs are extremely memory-inefficient. The method adopted for this thesis project is the Jenkins and Traub (1970) method which combines many of the above techniques, as well as ones by Traub, into a very complex algorithm and computer program which is convergent for a wide class of problems and is very machine-efficient. Discussed briefly are the more prominent methods.

Newton's Method

The process of finding a root by Newton's method is perhaps the best known and most easily understood (Blomquist 1968). It is based on beginning with an initial guess for the root and iteratively converging to the actual root with the method of steepest descent. The following relationship describes the method:

$$z_{n+1} = z_n - P(z_n)/P^1(z_n) \tag{4.4}$$

where z_n is the current guess of the root, $P(z_n)$ is the value of the polynomial at z_n, $P^1(z_n)$ is the value of the derivative of $P(z_n)$ and z_{n+1} is the next guess (or, eventually, the root). This process may be carried out iteratively to any practical, desired accuracy. The root is obtained when the difference between z_{n+1} and z_n is less than the required accuracy. Figure 7 graphically shows how successive iterations ultimately converge to the root.

Problems with this technique occur since it has no direct way of determining if a multiple root exists and the method does not always converge to the root. An example of how the method may fail is shown in Figure 8. This case demonstrates that choosing an initial guess too far from the actual root may prevent convergence. In this example, the method will oscillate between z_1 and z_0, since each point represents the other's successive approximation. The method also requires the use of complex arithmetic in evaluating $P(z_n)$ and $P^1(z_n)$ in the case of complex roots, which further limits this method.

Bairstow's Method

Prior to the Jenkins and Traub (1970) approach, the Bairstow method was regarded as one of the best techniques for extracting zeros. The primary advantage which this method has over Newton's Method is that it uses only real arithmetic to evaluate the

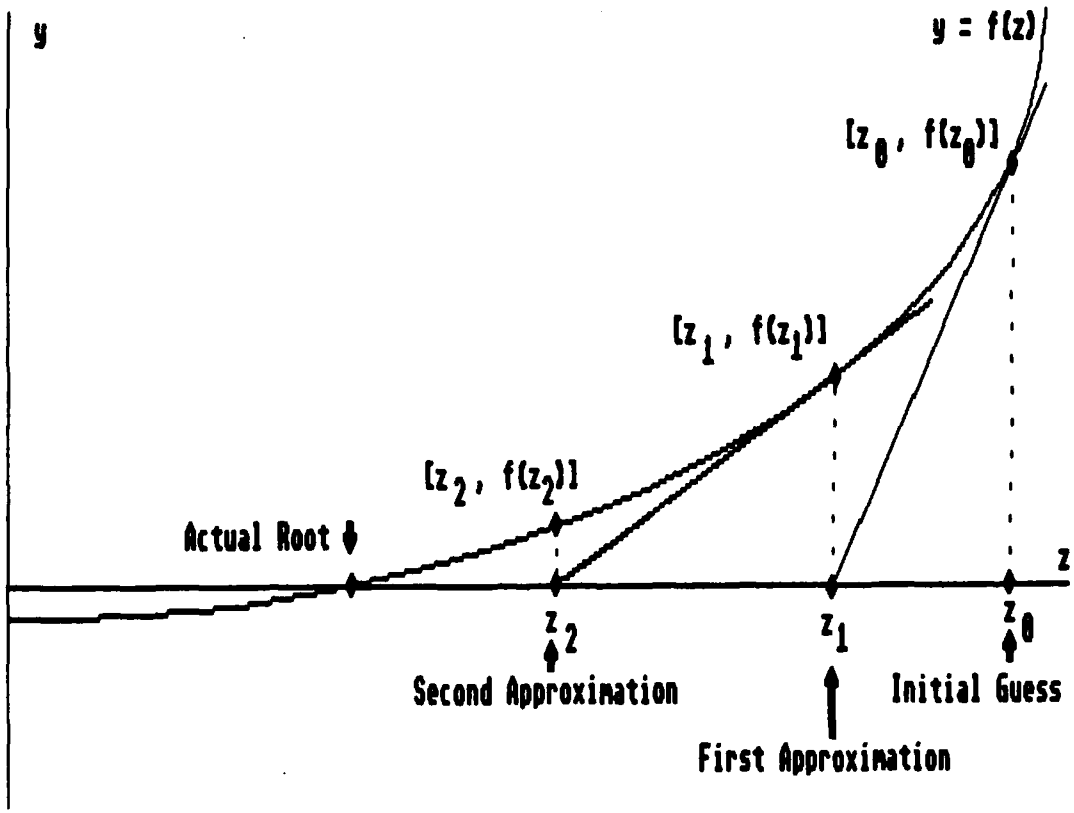


Figure 7. Newton's Method of Successive Approximation.

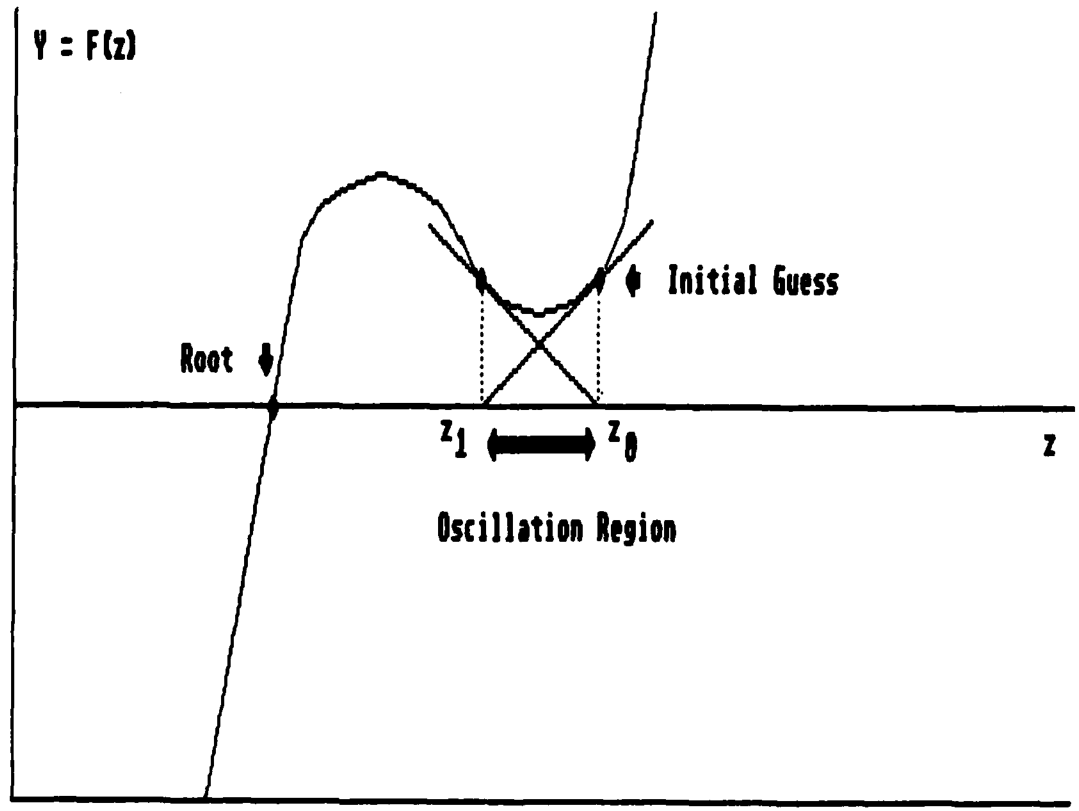


Figure 8. Newton's Method - Failure to Converge.

polynomial. The basic idea is to use Newton's Method to find unique quadratic factors to the polynomial using only real arithmetic, remove the quadratic via synthetic division and use the well-known quadratic formula to extract the complex roots of the quadratic factor. This technique automatically locates multiple roots, since it deflates the polynomial by an order of two for each quadratic factor found.

Simons, Weeks and Kotick (1983) developed an elegant formulation which best expresses Bairstow's Method. The algorithm begins with a polynomial in the form of:

$$P_N(z_n) = a_N z^N + a_{N-1} z^{N-1} + a_{N-2} z^{N-2} + \ldots + a_1 z^1 + a_0 z^0 \qquad (4.5)$$

Newton's Method is used to approach the root by using equation (4.4). However, Bairstow's Method evaluates $P_N(z)$ and its derivative(s) at the pair of complex points z_n and $z_n{}^*$ by using only real arithmetic as follows:

$$\text{Let } P_N(z) = \sum_{n=0}^{N} a_n z^n \qquad (4.6)$$

Let the quadratic factor take the form:

$$z^2 + \alpha z + \beta \qquad (4.7)$$

where:

$$\alpha = -2\sigma$$

and

$$\beta = \sigma^2 + \omega^2$$

Then, dividing $P_N(z)$ by this quadratic factor yields a polynomial of order N-2 with a remainder $R_1z + R_0$, i.e.,

$$\frac{P_N(z)}{z^2 + \alpha z + \beta} = P_{N-2}(z) + \frac{R_1z + R_0}{z^2 + \alpha z + \beta} \tag{4.8}$$

Multiplying both sides by the quadratic factor yields:

$$P_N(z) = P_{N-2}(z)\ (z^2 + \alpha z + \beta) + R_1z + R_0 \tag{4.9}$$

Obviously, if $R_1z + R_0 = 0$, then the roots are:

$$\frac{-\alpha \pm \sqrt{\alpha^2 - 4\beta}}{2} \tag{4.10}$$

by the quadratic formula. Therefore, the problem defaults to choosing values for α and β such that the remainder is zero. R_1 and R_0 can be related to $P_N(z)$ by the following:

$$
\begin{array}{r l}
 & a_N z^{N-2} + (a_{N-1} - \alpha a_{N-1}) z^{N-3} + [a_{N-2} - \alpha a_{N-1} + (\alpha^2 + \beta) a_N] z^{N-3} + \ldots \\
z^2 + \alpha z + \beta \,\big) & a_N z^N + a_{N-1} z^{N-1} + a_{N-2} z^{N-2} + a_{N-3} z^{N-3} + \ldots \\
 & \underline{a_N z^N + \alpha a_N z^{N-1} + \beta a_N z^{N-2}} \\
 & (a_{N-1} - \alpha a_N) z^{N-1} + (a_{N-2} - \beta a_N) z^{N-2} + a_{N-3} z^{N-3} + \ldots \\
 & \underline{(a_{N-1} - \alpha a_N) z^{N-1} + (\alpha a_{N-1} - \alpha a_N) z^{N-2} + (\beta a_{N-1} - \alpha \beta a_N) z^{N-3}} \\
 & [a_{N-2} - \alpha a_{N-1} + (\alpha^2 - \beta) a_N] z^{N-2} \\
 & \qquad + (a_{N-3} - \beta a_{N-1} + \alpha \beta a_N) z^{N-3} + \ldots \\
 & \qquad \vdots
\end{array}
\tag{4.11}
$$

Letting:

$$
\begin{aligned}
b_{N-2} &= a_N \\
b_{N-3} &= a_{N-1} - a_N \alpha = a_{N-1} - b_{N-2} \\
b_{N-4} &= a_{N-2} - a_N \alpha - a_{N-1} \alpha - a_N \alpha \\
&= a_{N-2} - b_{N-2} - b_{N-3} \alpha
\end{aligned}
\tag{4.12}
$$

a recursive relationship emerges:

$$
b_{N-2-n} = a_{N-2-n} - \beta b_{N-2-n} - \alpha b_{N-3-n} \tag{4.13}
$$

Iterating to n = N yields:

$$
\begin{aligned}
b_o &= a_2 - \beta b_2 - \alpha b_1 \\
b_{-1} &= a_1 - \beta b_1 - \alpha b_o \qquad (4.14) \\
b_{-2} &= a_o - \beta b_o - \alpha b_{-1}
\end{aligned}
$$

Since

$$\frac{R_1 z + R_o}{z^2 + \alpha z + \beta} = b_{-1} z^{-1} + b_{-2} z^{-2} \qquad (4.15)$$

then

$$R_1 z + R_o = (z^2 + \alpha z + \beta)\ (b_{-1} z^{-1} + b_{-2} z^{-2}) \qquad (4.16)$$

$$= b_{-1} z + b_{-2} + \alpha b_{-1} + \frac{\alpha b_{-2}}{z} + \frac{\beta b_{-1}}{z} + \frac{\beta b_{-2}}{z^2} \qquad (4.17)$$

Equating like coefficients:

$$
\begin{aligned}
R_1 &= b_{-1} \\
R_o &= b_{-2} + \alpha b_{-1} \\
&= a_o - \beta b_o - \alpha b_{-1} + \alpha b_{-1} \qquad (4.18) \\
&= a_o - \beta b_o
\end{aligned}
$$

The polynomial $P_N(z)$ and its derivative(s) may now be evaluated at any pair of complex conjugate points specified by the chosen quadratic factor. Newton's Method now follows easily, unless $P_N^{1}(z)$ evaluates to zero. If this is the case, multiple roots exist for the chosen pair, $\sigma \pm j\omega$. In this case:

$$\frac{P_N(z)}{P_N^{1}(z)} \tag{4.19}$$

is replaced by successive derivatives

$$\frac{P_N^{M}(z)}{P_N^{M+1}(z)} \tag{4.20}$$

until $P_N^{M+1}(z_n) \neq 0$. Now the root, as well as its multiplicity, is known.

The final step of the process is to remove each quadratic factor from the polynomial (polynomial deflation) using synthetic division. Bairstow's Method is again applied to the resulting polynomial until all of the roots are found.

The disadvantage of Bairstow's Method lies in the necessity to make a reasonably accurate guess of the quadratic factor. A bad guess can prevent convergence in the same manner as happens in Newton's Method. Both of the above processes suffer from machine rounding problems which occur with successive deflation of the original, high ordered polynomial.

Jenkins and Traub Method

This is a highly complex method which attempts to incorporate all of the above advantages while avoiding the mentioned pitfalls. The program incorporating the algorithm was the subject of Jenkins' doctoral dissertation (Jenkins 1975). The process incorporates three stages of zero extraction using Bairstow's Method, Newton's Method and three shifting techniques used to hasten convergence. The method assures rapid convergence for a wide class of polynomials. Zeros are removed in roughly increasing order of modulus; i.e., the zeros closest to the origin are generally removed first, the ones furthest from the origin are removed last. This is done in order to reduce the instability problems which may accompany the deflation process. A discussion of the variable shift algorithm is beyond the scope of this thesis and the reader is referred to the Jenkins and Traub (1970) paper for a formal theoretical treatment.

One of the interesting aspects addressed by Jenkins in writing the program is that it takes into account the specific capabilities and limitations of floating point manipulations on a given machine. This feature allows the program to be customized to a specific machine in order to achieve the highest possible zero extraction accuracy for that machine (using the Jenkins and Traub algorithm).

The program in Appendix B appears to be the current state-of-the-art in non-matrix methods of polynomial factoring. For example, the 1986 IMSL math libraries make use of this algorithm for their zero extraction approach. Schelin (1983) also indicated that this was the prime non-matrix type algorithm as of 1982.

In practice, the program handles up to roughly an order of one-hundred with little difficulty. Convergence problems begin to occur with increasing frequency beyond this limit, based upon actual tests.

CHAPTER IV

ZERO SEPARATION AND RECONSTITUTION

The prime reason for judicious zero separation is based on a desire to increase the dynamic range of the transducer or DSP device without sacrificing any of its transfer characteristics. The dynamic range is largely affected by relatively small FIR coefficients. These small coefficients correspond to small area overlaps of fingers in SAW devices and to small register coefficients in DSP filters. The net effect in the SAW device is for the small overlap to appear more like a point source wave generator as opposed to a desired planar wave source. Second order effects also begin to become more predominant for this situation. Similarly, DSP filters suffer from rounding effects when forced to sum products of very large and very small filter coefficients, even when floating-point arithmetic is used. The following DSP example demonstrates this problem.

1. Let the internal number representation be from - 1.00 to 1.00.

2. Let the system function be:

$$y(n) = (1.00)\ x(n) + (0.02)\ x(n-1)$$

which is an FIR filter with coefficients 1.00 and 0.02.

3. Let $x(0) = x(1) = 0.20$ and $x(-1) = 0.0$, then it follows that:

$$y(0) = (1.00)\ (0.20) + (0.02)\ (0.00) = 0.20$$

$$y(1) = (1.00)\ (0.20) + (0.02)\ (0.20) = 0.204 \text{ (actual)}$$

However, $y(1) = 0.204$ will be truncated to 0.20, since the internal precision only allows two decimal places.

In practice, it may not be possible to completely avoid the problems associated with dynamic range, but it is desirable to attain the best dynamic range for a given design. The above example demonstrates that three qualities of the FIR coefficients bear close scrutiny during the design process: the average, the variance and the range of the coefficient values.

Statistical Qualities

The highest average value for the FIR coefficients provides the greatest average finger overlap on a SAW device, thus the highest average energy injection into (or removal from) the substrate. Similarly, the highest average coefficient value in a DSP filter produces data with the highest average value within the register working ranges. Since sign changes may be handled easily in either type of device, the average is taken of the absolute value of the FIR coefficients. The average is defined as:

$$\overline{x} \triangleq \frac{\sum_{n=1}^{N} |h(n)|}{N} \qquad (5.1)$$

The variance of the FIR coefficients indicates how much they change on the average. In other words, the variance indicates how smooth the overall envelope is. Again, since sign changes may be handled easily in either type of device, the variance is taken of the absolute value of the FIR coefficients. The variance is defined as:

$$\sigma^2 \overset{\Delta}{=} \frac{N \times \sum_{n=1}^{N} [h(n)]^2 - \{\sum_{n=1}^{N} |h(n)|\}^2}{N\ (N-1)} \tag{5.2}$$

The range of the coefficients is an absolute indication of now far apart the minimum and maximum coefficient values are. Since the coefficients are normally scaled to a maximum of 1.0, the range will provide an indication of the minimum coefficient value. As in the above two statistical qualities, the range is taken for the absolute coefficient values. It is defined as:

$$\text{Range} \overset{\Delta}{=} |h(n)_{max}| - |h(n)_{min}| \tag{5.3}$$

These three qualities allow the designer some means of rating one given design against another quantitatively, leading to a design Figure of Merit (FOM).

The Figure of Merit (FOM)

A design objective which requires the best split of the zeros of the transfer function requires that the designer be able to rate one design over another until the best is found. This can be done by assigning an FOM to the design based on the three statistical qualities discussed above. The goal addressed here is to split the zeros between two transducers or processors such that a gain in available dynamic range is found. To do this, the following procedure relating the combined transducer coefficient average, variance and range values to a figure of merit is proposed:

1. Normalize all $h_1(n)$ coefficients to 1.0 maximum.
2. Normalize all $h_2(n)$ coefficients to 1.0 maximum.

   (NOTE: For the special case where all of the FIR coefficients are implemented by $h_1(n)$, then $h_2(n)$ is set to 1 to represent the impulse function since the convolution of the impulse response with an impulse is the impulse response.)

3. Form an array consisting of $|h_1(n)|$.
4. Concatenate the $|h_2(n)|$ values to this array.
5. Find the average ($\overline{x}$), variance (σ^2) and range of the newly-formed array.
6. Apply the relation:

$$\mathrm{FOM} = \frac{\overline{x}}{\sigma^2 \cdot \mathrm{Range}} \tag{5.4}$$

This equation reflects a desire to maximize the average while minimizing the variance and range of the coefficient values. It provides the designer with a means of rating one set of split zeros versus another.

Splitting the Zeros

In order to determine if an optimum zero-splitting pattern might exist, all possible combinations for distributing the zeros must be made and each combination tested using the FOM procedure. This process is very tedious but may be readily implemented on a computer due to the iterative nature of the process. Since there are:

$$\sum_{K=0}^{N/2} \frac{N!}{K!\,(N-K)!} \tag{5.5}$$

total, non-repeating combinations, the process has a practical computational upper limit of about FIR order N = 40 (which has over 10^{11} combinations) on a VAX 11-750. However, equations of low order may be used to model any trends applicable to the higher order cases encountered in SAW devices.

Computer Implementation

Appendix C shows a listing of the program COMBO used to generate (Beckenbach 1964) and rate all of the combinations of zeros provided by the Jenkins (1975) program. These zeros are the result of factoring the frequency response provided by the McClellan, Parks and Rabiner (1973) program.

COMBO first generates an array consisting of all of the zeros (complex conjugate pairs or reals) to be placed in $H_1(z)$, while all others are assumed to be placed in $H_2(z)$. The program insures that for responses which contain five real zeros, a sufficient number of combinations occur. Once a combination has been specified, program control is passed to the zero reconstitution subroutine.

The reconstitution subroutine POLYRECON is responsible for multiplying the appropriate zeros together to form a test case $H_1(z)$ and $H_2(z)$. This algorithm uses either a real root to form a linear factor or a complex conjugate pair to form a quadratic factor. All arithmetic performed is real. The $H_1(z)$ and $H_2(z)$ arrays are readily converted to scaled $h_1(n)$ and $h_2(n)$ arrays. The arrays are combined as described in the FOM procedure, the statistics are performed by the HSTAT subroutine and an FOM is assigned. Next, the FOM is compared to the FOM of the previous design and a decision is made as to which is best. If the new design is best, those results are stored and the program proceeds to iterate again. Upon completion, the results are passed back to the calling routine for further analysis.

CHAPTER VI

COMPUTER AIDED DESIGN APPLICATION

The Solid State Devices Lab group at the University of Central Florida, headed by Dr. Donald Malocha, uses a computer analysis system called SAWCAD, developed at UCF, to design SAW filter devices. Added to this program are the McClellan, Parks and Rabiner (1973) (Remez) program, the Jenkins (1975) zero location program and the Optimizing program discussed in Chapter V. The following is a brief discussion of the use of the added features to SAWCAD. The complete SAWCAD package is not discussed here. Further information on other aspects of SAWCAD can be obtained by referring to Richie (1983).

A listing of the main menu is shown in Figure 9. A filter design is initiated by selecting the (C)hebyshev [REMEZ] option to the main menu. The program proceeds to the McClellan, Parks and Rabiner (1973) program which has been somewhat modified. The data entry routine consists of an interactive session between the computer and the designer. During this session, the designer is asked to specify the filter type (i.e., bandpass, differentiator or Hilbert transformer), the number of distinct frequency bands up to $f_{sample}/2$, the start and stop frequencies of each band, the maximum dB level in each band and the maximum allowable ripple in the primary passband (multiple passband designs are possible).

```
==========================================
<<<S A W C A D   M A I N   M E N U>>>
==========================================

(E)igen_synthesis      (A)nalysis_of_design

(C)hebyshev [REMEZ]    (Z)ero extraction

           (S)plit Transducers

(G)raphics_menu        (M)ultipy_data_files

(R)ead_disk_file       (W)rite_disk_file

(H)elp_status          (Q)uit_SAWCAD

 COMMAND : ==> █
```

Figure 9. SAWCAD Main Menu.

Once frequency range, function and amplitude information is entered, the program makes an initial guess of the required filter order and initiates a design. If the design fails to converge to the required specifications, the filter order is increased and the process is repeated until convergence is obtained. Once a valid design is found, the design report is printed and control is passed back to the SAWCAD main menu.

At this point, the impulse response coefficients exist in memory only. The SAWCAD (W)rite function is selected and a disk file containing the impulse response coefficients may be written. This step is highly recommended. Control passes back to the SAWCAD main menu.

Any number of options are now open to the designer. The obtained response could be used immediately with other SAWCAD functions if desired (i.e., FFT, Graphics analysis, etc.). The next option we shall concern ourselves with here is the (Z)ero Extraction option. Selecting this option requires no further input, since the program calls the Jenkins and Traub program to factor the z-transform of the impulse response. The program returns the zeros and places them in the amp and phase variables of the SAWCAD program. The designer must be aware that the impulse response is no longer in memory!

Next, the optimum split for low order designs can be found using the (S)plit Transducers selection on the SAWCAD main menu.

No further user input is required since the program iterates until the best split design is found. Once found, the program prompts the user for transducer 1 and 2 file names under which to save the impulse response coefficients.

Generation of the design is now complete. It may be checked by multiplying the transducer 1 and 2 data files together and comparing the product with the original response generated by the Remez method. These comparisons may be done graphically using the powerful graphics and FFT facilities of the SAWCAD environment.

CHAPTER VII

RESULTS

In an attempt to determine a general algorithm applicable to filters of any order or type, eight different low-order test filters were designed. Four of the filters were low pass designs and four were high pass. The input design specs appear in Table 3.

The designs were made using the Remez technique, factored by the Jenkins (1975) program, and were subjected to the optimizing program to test all possible split designs. The composition of each transducer was recorded to reflect the number of stopband and passband zeros, and whether these zeros were real or complex. These results are tabulated in Table 4.

Careful study of the results does not indicate any clearly emerging pattern. In three out of four of the cases with very narrow passbands (LP1, HP1 and HP4), the passband zeros all appeared on one transducer accompanied by several stopband zeros. However, LP4 passband zeros were split between the two transducers. The other cases did not produce results which might indicate a predictable pattern.

Another test was devised to test and rate a conventional no-split design, a strictly passband-stopband split, the "alternating zero" algorithm employed by other design groups (Morimoto et al.

TABLE 3

DESIGN INPUT SPECS

DESIGNATION	PASSBAND REGION	STOPBAND REGION	PASSBAND RIPPLE (db)	SIDELOBE LEVEL (db)
LP1	0.0 -0.1	0.2 -0.5	0.5	-38
LP2	0.0 -0.2	0.3 -0.5	0.5	-38
LP3	0.0 -0.3	0.4 -0.5	0.5	-38
LP4	0.0 -0.1	0.15-0.5	0.5	-38
HP1	0.4 -0.5	0.0 -0.3	0.5	-38
HP2	0.3 -0.5	0.0 -0.2	0.5	-38
HP3	0.2 -0.5	0.0 -0.1	0.5	-38
HP4	0.45-0.5	0.0 -0.4	0.5	-38

TABLE 4

TRANSDUCER COMPOSITIONS

DESIGNATION	TYPE ZERO*	TRANSDUCER 1		TRANSDUCER 2		RATIO #1/#2
		COMPLEX	REAL	COMPLEX	REAL	
LP1	S	8	0	6	0	12/ 6
	P	4	0	0	0	
LP2	S	4	0	4	0	6/10
	P	0	2	4	2	
LP3	S	2	0	2	0	7/ 7
	P	4	1	4	1	
LP4	S	4	0	20	1	6/27
	P	0	2	4	2	
HP1	S	6	0	6	0	6/10
	P	0	0	0	4	
HP2	S	6	0	8	0	8/18
	P	0	2	8	2	
HP3	S	2	0	4	0	9/ 9
	P	4	3	4	1	
HP4	S	8	0	20	0	8/24
	P	0	0	0	4	

* S = stopband, P = passband

1980 and Ruppel et al. 1984), and the split algorithm presented here. The HP1 filter spec was arbitrarily chosen as the subject of the test. Figure 10 shows the overall frequency response of the HP1 filter.

The conventional no-split design yielded a figure of merit (FOM) of about 2.5214, whereas the passband-stopband split (i.e., all passband zeros on one transducer and all stopband zeros on the other) yielded an FOM of about 3.588. Figures 11 and 12 show the frequency response of each transducer. The passband transducer shows nearly unity gain at 0 with a gradual rolloff in the region of f_0, a very difficult response to implement on a SAW device.

The "alternating zero" approach yielded a FOM of about 6.5756. This shows an improvement in the quality of the design as compared to the passband-stopband split method. Figures 13 and 14 show the frequency response of each transducer. The zeros are clearly orthogonal. Figures 15 and 16 show the impulse response representations. This design does appear to have merit in the case where a fifty-fifty split is highly desired.

The technique presented here produced a design with a FOM of about 9.753. Figures 17 and 18 show the frequency response of each transducer and figures 19 and 20 show the corresponding impulse response plots.

The last two designs do not appear to pose fabrication problems which might plague the passband-stopband split design case. As a check of the split, the frequency responses of the last design were multiplied together via SAWCAD and replotted. That plot is exactly the same as the original frequency response in Figure 10, as expected.

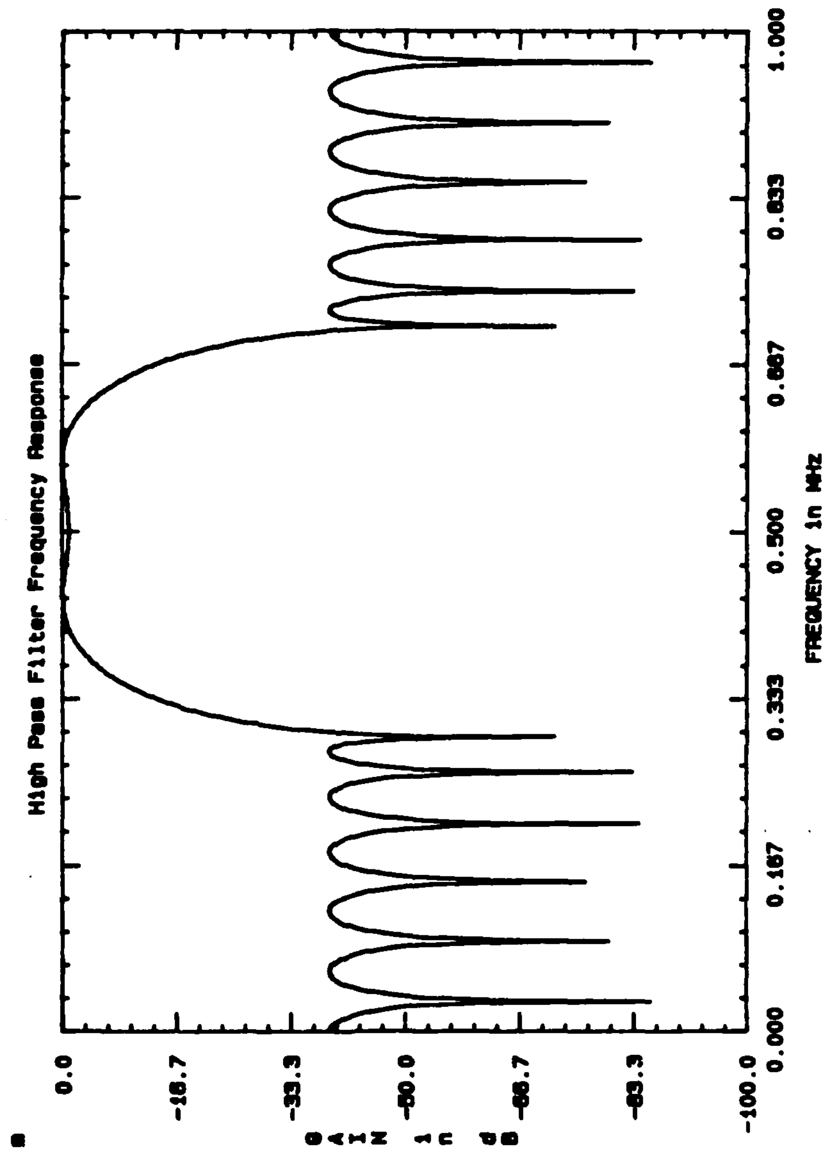


Figure 10. Overall Frequency Response.

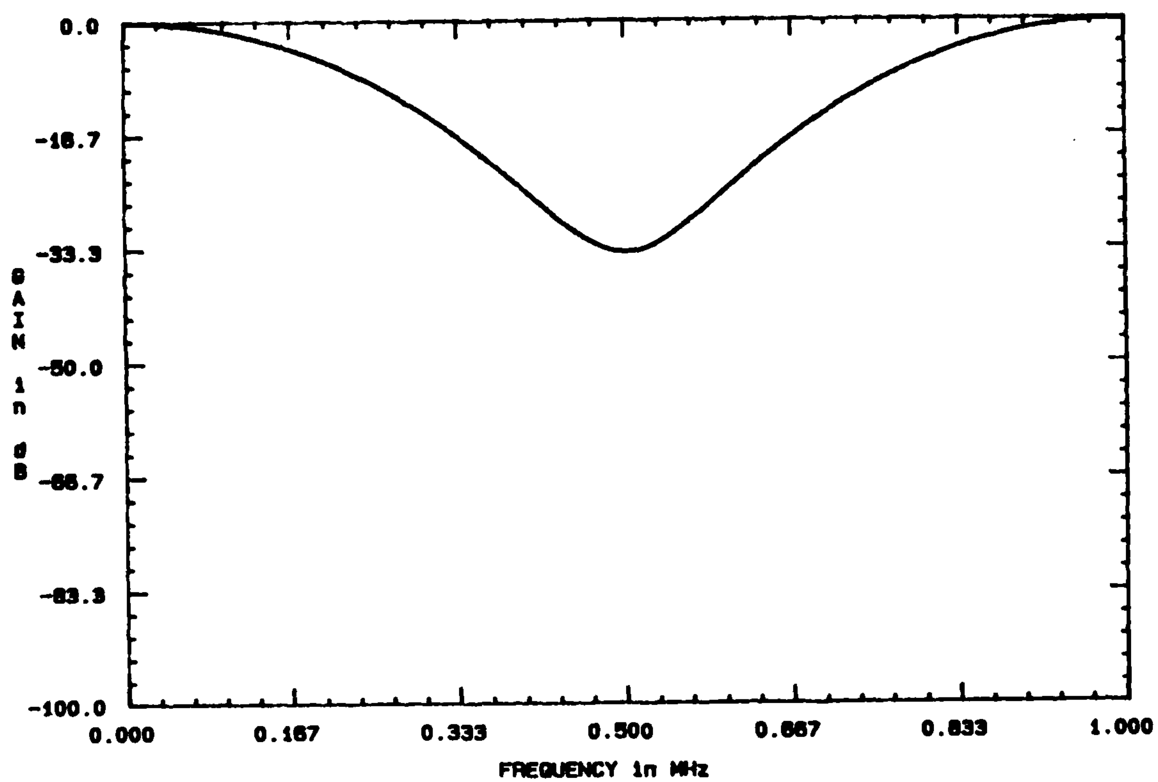


Figure 11. Transducer 1 Frequency Response (Passband-Stopband Design).

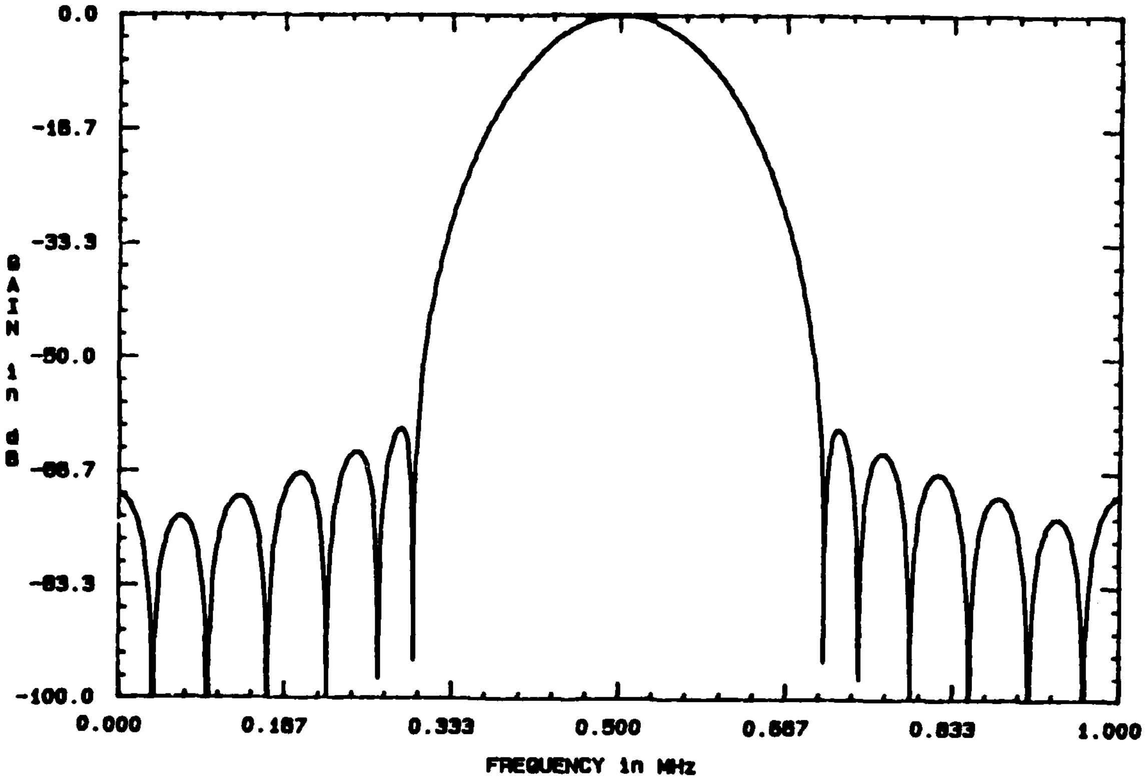


Figure 12. Transducer 2 Frequency Response (Passband-Stopband Design).

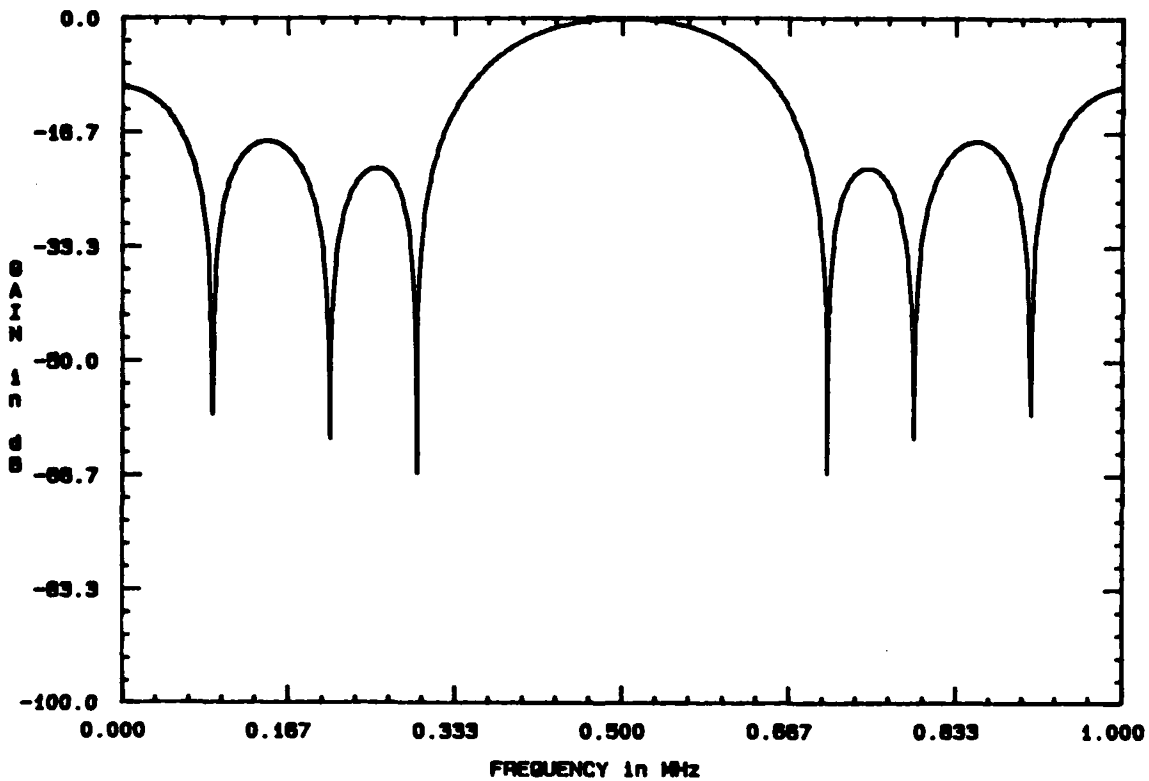


Figure 13. Transducer 1 Frequency Response (Alternating Zero Design).

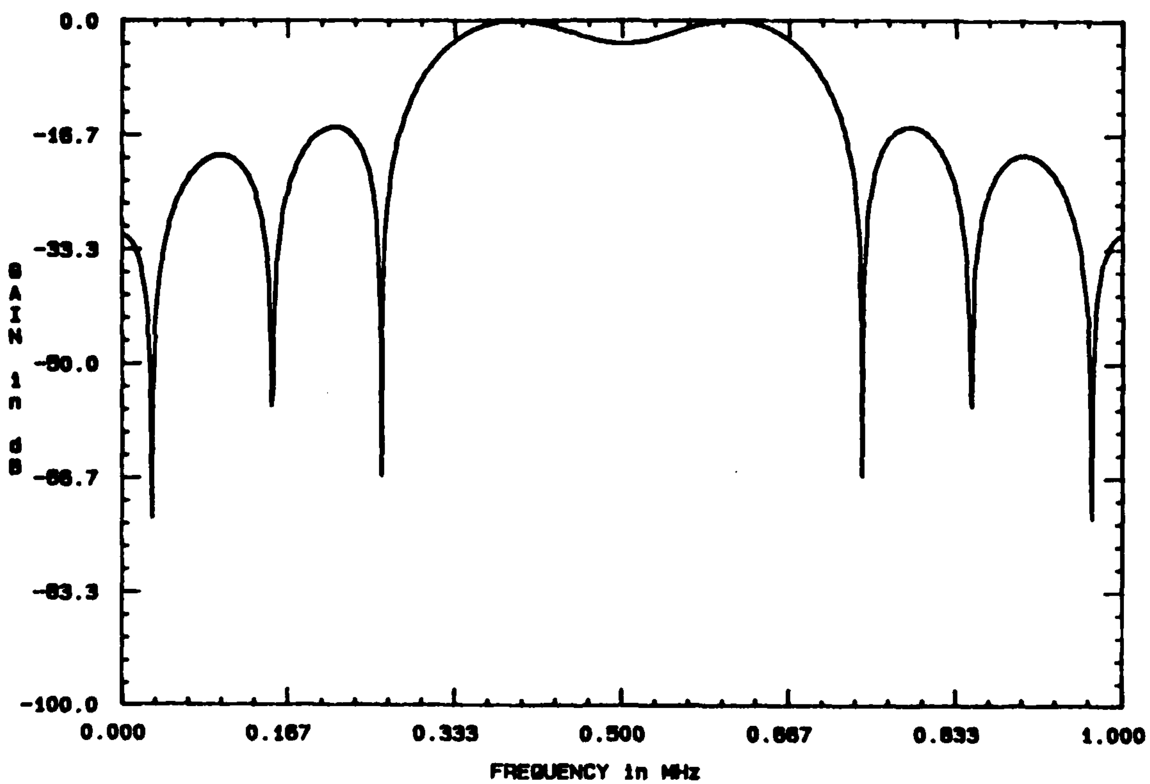


Figure 14. Transducer 2 Frequency Response (Alternating Zero Design).

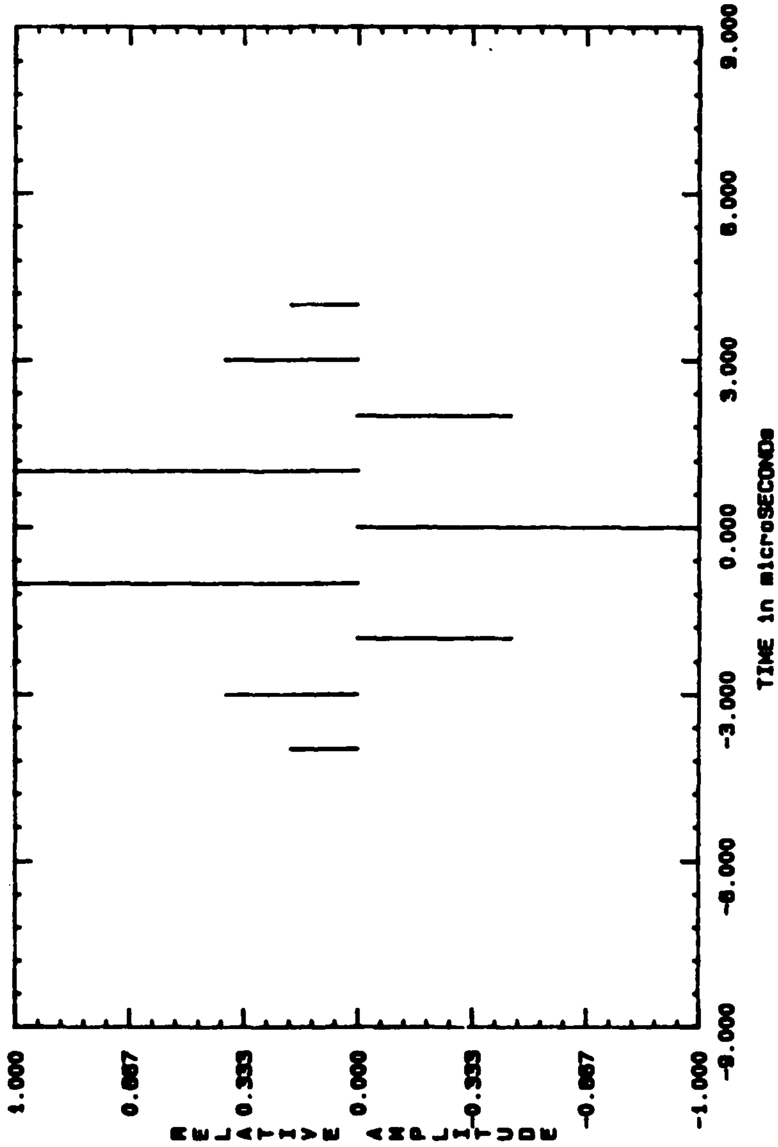


Figure 15. Transducer 1 Impulse Response (Alternating Zero Design).

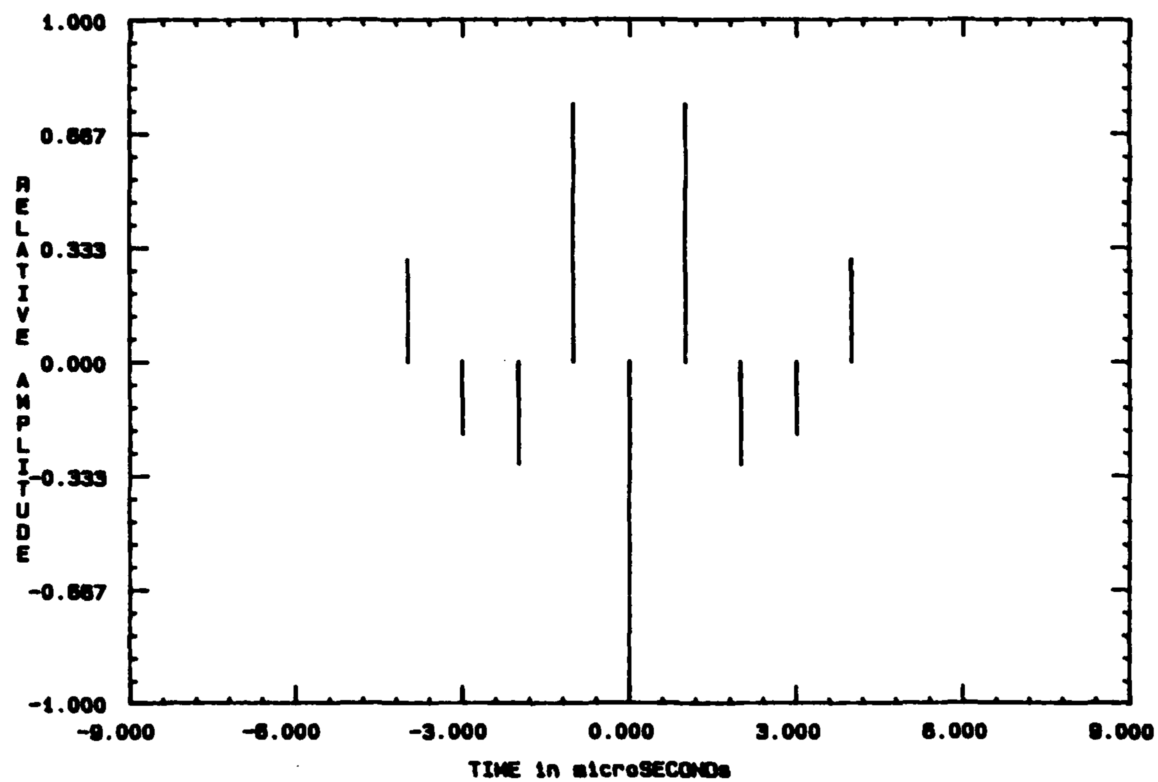


Figure 16. Transducer 2 Impulse Response (Alternating Zero Design).

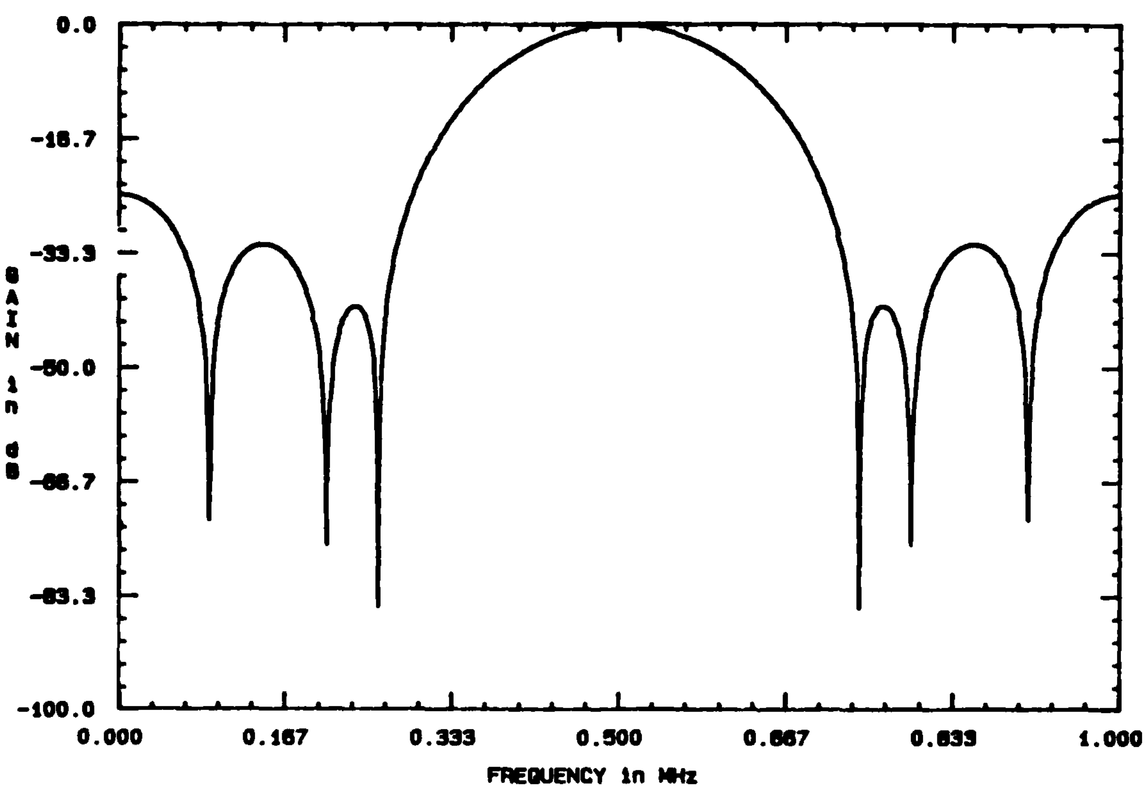


Figure 17. Transducer 1 Frequency Response (Fully Optimized Design).

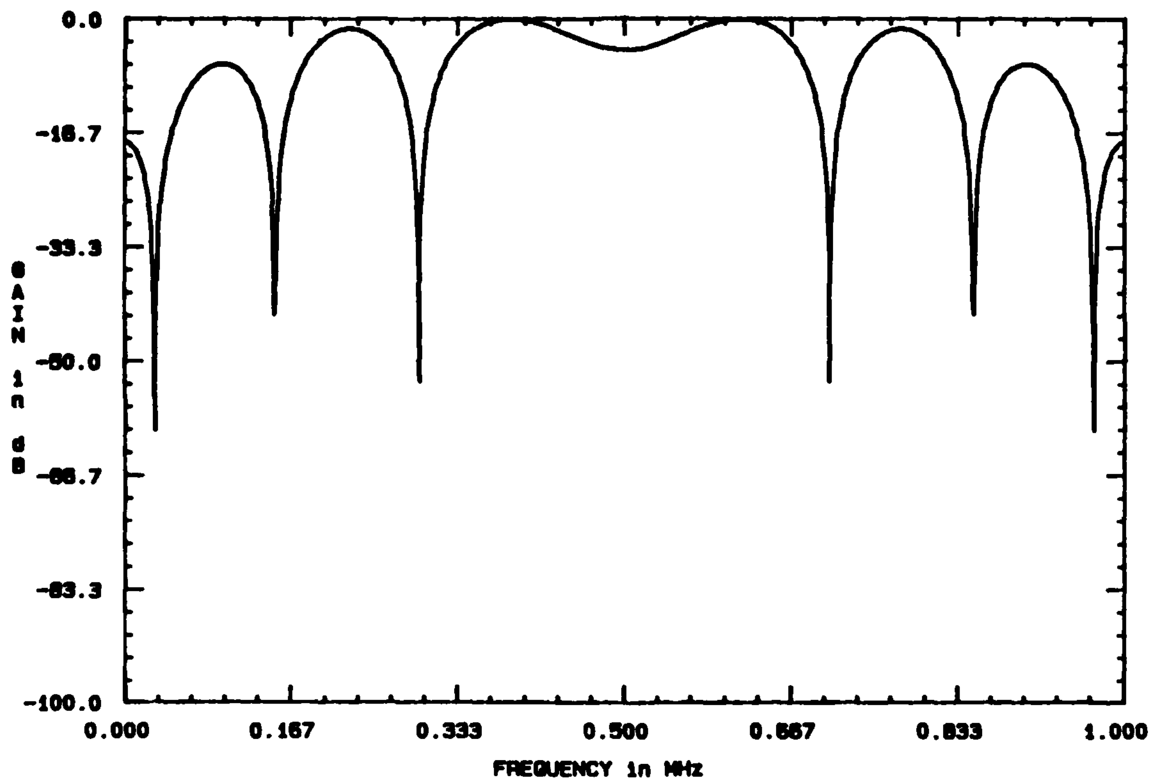


Figure 18. Transducer 2 Frequency Response (Fully Optimized Design).

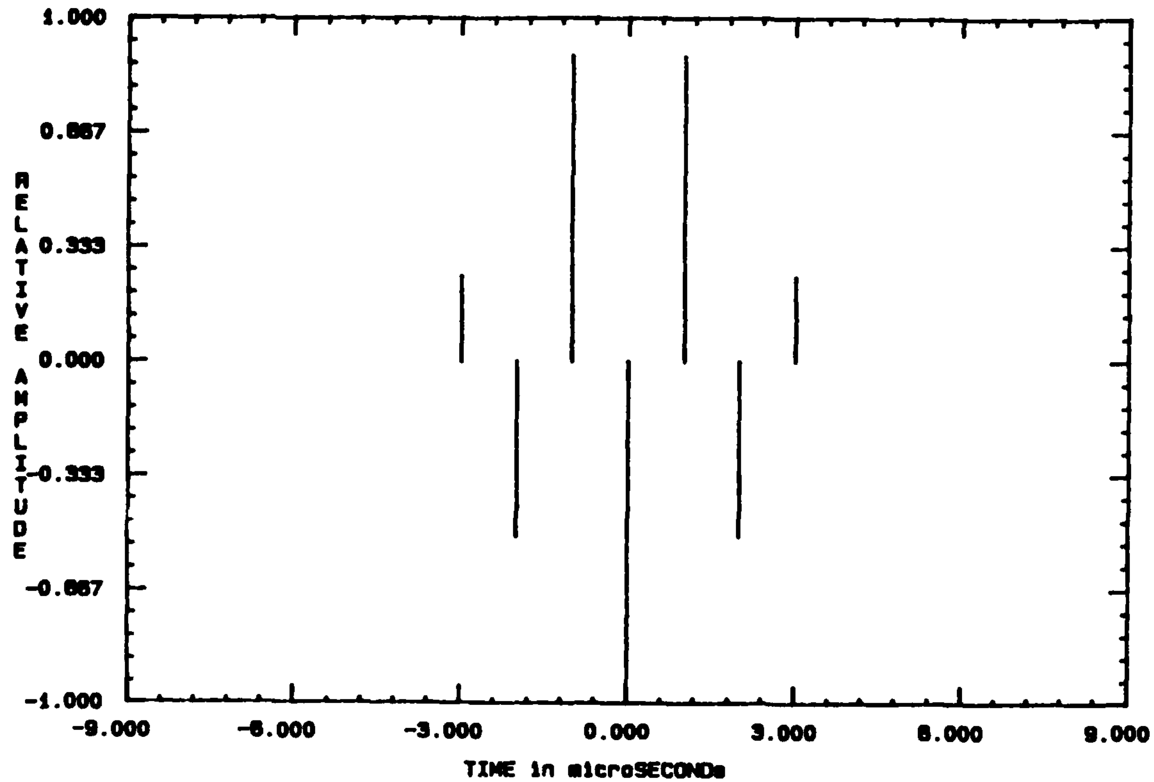


Figure 19. Transducer 1 Impulse Response (Fully Optimized Design).

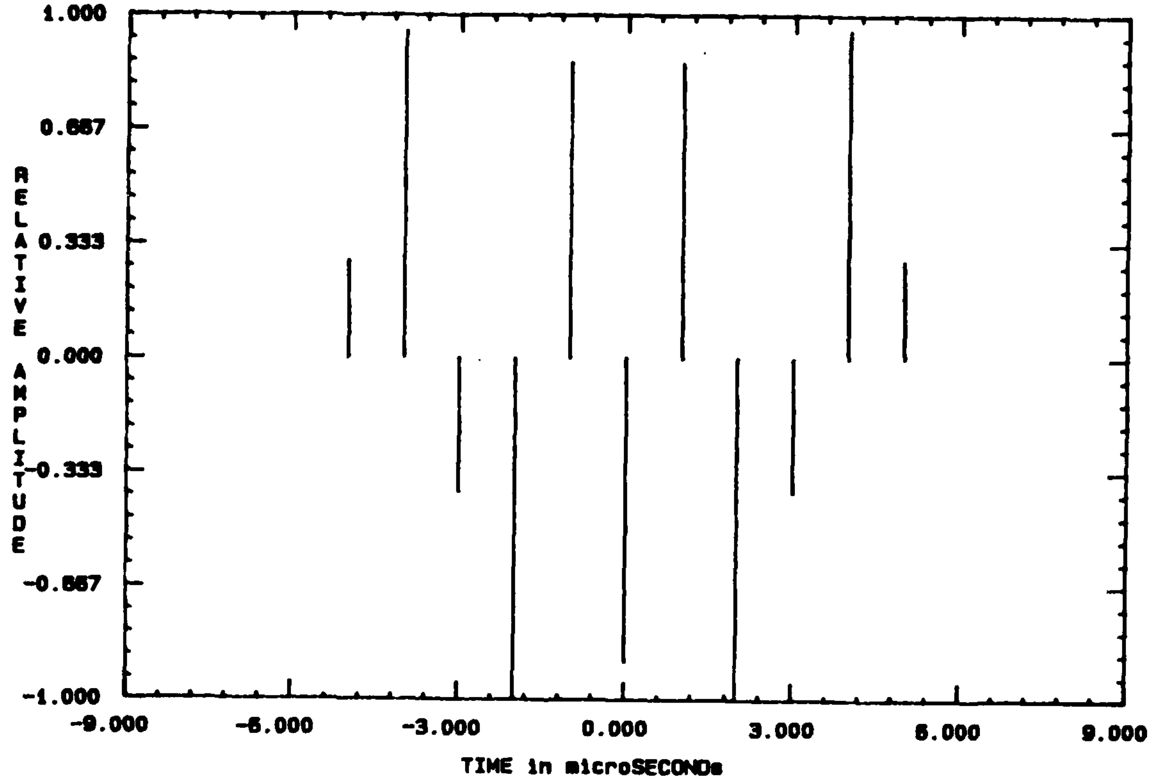


Figure 20. Transducer 2 Impulse Response (Fully Optimized Design).

CHAPTER VIII

CONCLUSIONS

The technique presented in this thesis of optimally splitting the zeros of the transfer function shows considerable immediate promise for low order (N less than 40) filter designs. An algorithm permitting the split design of high order filters has not become readily apparent, although the presented "all-combinations" technique may still be used if limits are placed upon the transducer sizes. For example, specifying a fifty-fifty split design significantly reduces the number of combinations which must be tested. Future efforts in this area must focus upon a more efficient search criteria or be content to accept less than optimal results, as in the "alternating zero" approach. The concepts presented here may be extended to generate multi-transducer splits, with primary utility in large ordered digital filter implementations.

In spite of the limitations of the technique used here, it is obvious that a "best split" design of an FIR filter transfer function exists based upon the average size, variance and range of the resulting impulse response coefficients. The "alternating zero" split approach (Morimoto et al. 1980 and Ruppel et al. 1984), the only other method published to date, exhibited a lower FOM when compared to the split found by testing all possible combinations, based upon the stated criteria. Careful study of figures 15, 16, 19 and 20 (shown in Chapter VII) demonstrates that the presented technique does, indeed, produce larger tap sizes for a given transfer function than the "alternating zero" approach. Therefore, for critical, low-ordered design cases, the technique presented here will provide best results.

The Jenkins and Traub factoring algorithm is a very accurate means of obtaining the zeros of polynomials within an order of one-hundred or so. However, high ordered SAW filter designs will require that the zeros of polynomials of a thousand order, or more, will need to be extracted. Based upon observation of the programs and polynomial characteristics encountered in this thesis, several alternative approaches to locating the zeros seem feasible.

One technique that merits further investigation is the use of the Fourier transform to expose the stopband zero locations. This can be done by noting where the stopband nulls occur and reference these points to the corresponding angles about the z-plane unit circle. The cosines and sines of these angles are the real and imaginary components of the stopband zeros which, of course, have corresponding complex conjugates in the lower half of the unit circle. Once all of the stopband zeros have been found, the z-polynomial may be deflated in one step, yielding a polynomial consisting of only the passband zeros. Next, the Jenkins and Traub algorithm may be applied to the greatly reduced polynomial to locate the passband zeros. The Fourier transform technique could be applied to any FIR obtained by any design technique.

Another zero location technique which may be explored is unique to the Parks and McClellan program. One of the outputs of the program is a listing of all of the Chebyshev polynomial extremal frequencies. Since the limits of each stopband are known (specified apriori), and since the extremal frequencies are fairly evenly spaced in the stopband, an interpolation between adjacent extremal frequencies will provide a good approximation (or, at least a good initial guess) to a function zero in the stopband. If the interpolation method is deemed sufficient, then all of the zeros found may be used to reduce the response down to a polynomial containing only the passband zeros, to which the Jenkins and Traub algorithm may be applied. A better approximation of the stopband zeros would be obtained by applying the interpolation approximation as an initial guess to Bairstow's technique to obtain the best estimate of the zero. Again, the passband zeros could be found using the Jenkins and Traub method.

In conclusion, an optimal split of zeros does exist, based upon the stated criteria. The major limiting factor of the presented technique is the requirement that all possible split combinations must be devised and rated with respect to each other. This is a very time-consuming process and future efforts may lead to a more efficient means of finding the optimal combination. The use of more powerful computing machines may make the split-design of somewhat higher order filters practical, but a reasonable upper limit is rapidly approached with each increase in filter order. Perhaps the best tradeoff between this method and the "alternating zero" approach is to obtain the best fifty-fifty split by trying all possible combinations for this case only (i.e., N zeros taken N/2 at a time). This approach will, at the very least, provide as good a design as the "alternating zero" approach, without the serious high order computational drawbacks of the method presented here. Additionally, some design considerations may favor a fifty-fifty split, as in the case of two digital signal processors being required to evenly share the processing tasks. Nevertheless, the purpose of this thesis was to prove the existence of one such optimal combination, not the optimal means of obtaining it. With a sigh of relief, and a hint of moderate surprise, such proof has been presented.

APPENDICES

APPENDIX A

FILTER DESIGN PROGRAM

```
      SUBROUTINE REMEZDES
C
C
C PROGRAM FOR THE DESIGN OF LINEAR PHASE FINITE IMPULSE
C RESPONSE (FIR) FILTERS USING THE REMEZ EXCHANGE
ALGORITHM
C JIM MCCLELLAN, RICE UNIVERSITY, APRIL 13, 1973
C
C MODIFIED BY KEITH V. LINDSAY, UNIVERSITY OF
C CENTRAL FLORIDA, 1 MARCH 86, FOR USE WITH UCF'S
C SAWCAD PROGRAM.
C
C THREE TYPES OF FILTERS ARE INCLUDED--BANDPASS FILTERS
C DIFFERENTIATORS, AND HILBERT TRANSFORM FILTERS
C
C THE INPUT DATA CONSISTS OF 5 SECTIONS
C
C SECTION 1--FILTER LENGTH, TYPE OF FILTER, 1-MULTIPLE
C PASSBAND/STOPBAND, 2-DIFFERENTIATOR, 3-HILBERT TRANSFORM
C FILTER, NUMBER OF BANDS, CARD PUNCH DESIRED, AND GRID
C DENSITY
C
C SECTION 2--BANDEDGES, LOWER AND UPPER EDGES FOR EACH
BAND
C WITH A MAXIMUM OF 10 BANDS.
C
C SECTION 3--DESIRED FUNCTION (OR DESIRED SLOPE IF A
C DIFFERENTIATOR ) FOR EACH BAND.
C
C SECTION 4--WEIGHT FUNCTION IN EACH BAND. FOR A
C DIFFERENTIATOR, THE WEIGHT FUNCTION IS INVERSELY
C PROPORTIONAL TO F
C
      COMMON/FILE/ AMP(4096),PHASE(4096),NFFT,ITYPE
      COMMON/DAT/ FO,TFLO,TFHI,NUM
      COMMON
PI2,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
      DIMENSION IEXT(514),AD(514),ALPHA(514),X(514),Y(514)
      DIMENSION H(514)
      DIMENSION DES(8224),GRID(8224),WT(8224)
      DIMENSION EDGE(20),FX(10),WTX(10),DEVIAT(10)
      DOUBLE PRECISION PI2,PI
      DOUBLE PRECISION AD,DEV,X,Y
      DOUBLE PRECISION HH
      LOGICAL FAIL, MOREN
      INTEGER PB, SB
      PI2=6.283185307179586
      PI=3.141592653589793
C
C THE PROGRAM IS SET UP FOR A MAXIMUM LENGTH OF 1024, BUT
C THIS UPPER LIMIT CAN BE CHANGED BY REDIMENSIONING THE
C ARRAYS IEXT, AD, ALPHA, X, Y, H TO BE NFMAX/2+2.
C THE ARRAYS DES, GRID, AND WT MUST BE DIMENSIONED
```

```
C 16(NFMAX/2+2).
C
      NFMAX=1024
100 CONTINUE
      JTYPE=0
C
C PROGRAM INPUT SECTION
C
      PRINT *,' DIGITAL FILTER DESIGN (FIR) VIA THE
     1 REMEZ EXCHANGE ALGORITHM.'
      PRINT *,' '
      PRINT *,'ENTER TYPE OF FILTER:'
      PRINT *,'      (1)-MULTIPLE PASSBAND/STOPBAND'
      PRINT *,'      (2)-DIFFERENTIATOR'
      PRINT *,'      (3)-HILBERT TRANSFORM FILTER'
      PRINT *,' '
      READ *,JTYPE
      PRINT *,' '
      PRINT *,'ENTER THE NUMBER OF BANDS '
      READ *,NBANDS
      PRINT *,' '
C     PRINT *,'OUTPUT THE IMPULSE RESPONSE (1=YES, 0=NO)'
C     READ *,JPUNCH
      JPUNCH=1
C     PRINT *,'ENTER THE GRID DENSITY'
C     READ *,LGRID
      LGRID=16
      IF(NBANDS.LE.0) NBANDS=1
C
C GRID DENSITY IS ASSUMED TO BE 16 UNLESS SPECIFIED
OTHERWISE.
C
      IF(LGRID.LE.0) LGRID=16
      DO 888 J=1,NBANDS
      PRINT *,' '
      PRINT *,'BAND ',J,':'
      PRINT *,'      LOWER EDGE:'
      READ *,EDGE(2*J-1)
      PRINT *,' '
      PRINT *,'      UPPER EDGE:'
      READ *,EDGE(2*J)
888 CONTINUE
      IF(JTYPE.EQ.2) GO TO 890
      PRINT *,' '
      PRINT *,'ENTER THE DESIRED FUNCTION OF EACH BAND'
      PRINT *,' (0=NOPASS, 1=PASSBAND)'
      DO 889 J=1,NBANDS
      PRINT *,' '
      PRINT *,'      BAND ',J,':'
      READ *,FX(J)
889 CONTINUE
     GO TO 893
890 DO 892 J=1,NBANDS
```

```
    PRINT *,'ENTER THE SLOPE OF BAND ',J
    READ *,FX(J)
892 CONTINUE
893 CONTINUE
    PRINT *,' '
    PRINT *,'ENTER THE MAX dB RIPPLE IN THE PASSBAND'
    READ *,DP
    DP=10.E0**(DP/20)-1.0E0
    DS=1.0E0
    DO 898 J=1,NBANDS
    PRINT *,' '
    PRINT *,'ENTER THE MAXIMUM dB LEVEL IN BAND ',J
    READ *,WTX(J)
    IF (WTX(J).EQ.0.0E0) THEN
            WTX(J)=1.0E0
            PB=J
            GO TO 898
    END IF
c   WTX(J)=AINT(10.0E0**((20.0E0*ALOG10(DP) -
WTX(J))/20.0E0) +1)
    WTX(J)=(10.0E0**((20.0E0*ALOG10(DP)-WTX(J))/20.0E0))
    IF (WTX(J).LT.1.0E0) WTX(J)=1.0E0
    PRINT *,'STOPBAND WEIGHTING =',WTX(J)
2477        IF (DS.GT.DP/WTX(J)) THEN
            DS=DP/WTX(J)
            SB=J
            IF (J.EQ.1) THEN
                    DELTAF=EDGE(3)-EDGE(2)
                    GO TO 898
            END IF
            IF (J.LT.NBANDS) THEN
                    IF (DP/WTX(J-1).EQ.DP) THEN
                            DELTAF=EDGE(J*2-1)-EDGE(J*2-2)
                            GO TO 898
                    END IF
                    DELTAF=EDGE(J*2+1)-EDGE(J*2)
                    GO TO 898
            END IF
            IF (J.EQ.NBANDS) THEN
                    DELTAF=EDGE(J*2-1)-EDGE(J*2-2)
            END IF
    END IF
898 CONTINUE
C
C  TAKE A GUESS AT AN INITIAL VALUE FOR NFILT
C  BASED UPON VAIDYANATHAN'S FORMULATION
C
    NFILT=INT((-10.0E0*ALOG10(DP*DS) - 13.0E0)/(14.6E0 *
DELTAF))
2126        PRINT *,'WORKING ON FILTER OF ORDER ',NFILT-1
    IF(NFILT.GT.NFMAX.OR.NFILT.LT.3) CALL ERROR
    IF(JTYPE.EQ.0) CALL ERROR
    NEG=1
```

```
      IF(JTYPE.EQ.1) NEG=0
      NODD=NFILT/2
      NODD=NFILT-2*NODD
      NFCNS=NFILT/2
      IF(NODD.EQ.1.AND.NEG.EQ.0) NFCNS=NFCNS+1
C
C SET UP THE DENSE GRID. THE NUMBER OF POINTS IN THE GRID
C IS (FILTER LENGTH + 1)*GRID DENSITY / 2
C
      GRID(1)=EDGE(1)
      DELF=LGRID*NFCNS
      DELF=0.5/DELF
      IF(NEG.EQ.0) GO TO 135
      IF(EDGE(1).LT.DELF) GRID(1)=DELF
135   CONTINUE
      J=1
      L=1
      LBAND=1
140   FUP=EDGE(L+1)
145   TEMP=GRID(J)
C
C CALCULATE THE DESIRED MAGNITUDE RESPONSE AND THE WEIGHT
C FUNCTION ON THE GRID
C
      DES(J)=EFF(TEMP,FX,WTX,LBAND,JTYPE)
      WT(J)=WATE(TEMP,FX,WTX,LBAND,JTYPE)
      J=J+1
      GRID(J)=TEMP+DELF
      IF(GRID(J).GT.FUP) GO TO 150
      GO TO 145
150   GRID(J-1)=FUP
      DES(J-1)=EFF(FUP,FX,WTX,LBAND,JTYPE)
      WT(J-1)=WATE(FUP,FX,WTX,LBAND,JTYPE)
      LBAND=LBAND+1
      L=L+2
      IF(LBAND.GT.NBANDS) GO TO 160
      GRID(J)=EDGE(L)
      GO TO 140
160   NGRID=J-1
      IF(NEG.NE.NODD) GO TO 165
      IF(GRID(NGRID).GT.(0.5-DELF)) NGRID=NGRID-1
165   CONTINUE
C
C SET UP A NEW APPROXIMATION PROBLEM WHICH IS EQUIVALENT
C TO THE ORIGINAL PROBLEM
C
      IF(NEG) 170,170,180
170   IF(NODD.EQ.1) GO TO 200
      DO 175 J=1,NGRID
      CHANGE=DCOS(PI*GRID(J))
      DES(J)=DES(J)/CHANGE
175   WT(J)=WT(J)*CHANGE
      GO TO 200
```

```
180 IF(NODD.EQ.1) GO TO 190
    DO 185 J=1,NGRID
    CHANGE=DSIN(PI*GRID(J))
    DES(J)=DES(J)/CHANGE
185 WT(J)=WT(J)*CHANGE
    GO TO 200
190 DO 195 J=1,NGRID
    CHANGE=DSIN(PI2*GRID(J))
    DES(J)=DES(J)/CHANGE
195 WT(J)=WT(J)*CHANGE
C
C INITIAL GUESS FOR THE EXTREMAL FREQUENCIES--EQUALLY
C SPACED ALONG THE GRID
C
200 TEMP=FLOAT(NGRID-1)/FLOAT(NFCNS)
    DO 210 J=1,NFCNS
210 IEXT(J)=(J-1)*TEMP+1
    IEXT(NFCNS+1)=NGRID
    NM1=NFCNS-1
    NZ=NFCNS+1
C
C CALL THE REMEZ EXCHANGE ALGORITHM TO DO THE
APPROXIMATION
C PROBLEM.
C
    CALL REMEZ(EDGE,NBANDS,MOREN)
    IF (MOREN.EQ..TRUE.) THEN
            NFILT=NFILT+1
            GO TO 2126
    END IF
    IF (DEV/WTX(PB).GT.DP .OR. DEV/WTX(SB).GT.DS) THEN
            NFILT=NFILT+1
C           PRINT *,'DEVIATION PB =',DEV/WTX(PB),'PB=',PB
C           PRINT *,'DEVIATION SB =',DEV/WTX(SB),'SB=',SB
            GO TO 2126
    END IF
C
C CALCULATE THE IMPULSE RESPONSE.
C
    IF(NEG) 300,300,320
300 IF(NODD.EQ.0) GO TO 310
    DO 305 J=1,NM1
305 H(J)=0.5*ALPHA(NZ-J)
    H(NFCNS)=ALPHA(1)
    GO TO 350
310 H(1)=0.25*ALPHA(NFCNS)
    DO 315 J=2,NM1
315 H(J)=0.25*(ALPHA(NZ-J)+ALPHA(NFCNS+2-J))
    H(NFCNS)=0.5*ALPHA(1)+0.25*ALPHA(2)
    GO TO 350
320 IF(NODD.EQ.0) GO TO 330
    H(1)=0.25*ALPHA(NFCNS)
    H(2)=0.25*ALPHA(NM1)
```

```
      DO 325 J=3,NM1
325   H(J)=0.25*(ALPHA(NZ-J)-ALPHA(NFCNS+3-J))
      H(NFCNS)=0.5*ALPHA(1)-0.25*ALPHA(3)
      H(NZ)=0.0
      GO TO 350
330   H(1)=0.25*ALPHA(NFCNS)
      DO 335 J=2,NM1
335   H(J)=0.25*(ALPHA(NZ-J)-ALPHA(NFCNS+2-J))
      H(NFCNS)=0.5*ALPHA(1)-0.25*ALPHA(2)
C
C SET UP IMPULSE RESPONSE/POLYNOMIAL ARRAY  **
C  ARRAY IN VARIABLE AMP                    **
C
350   DO 342 I=1,NFCNS
      AMP(I)=H(I)/H(NFCNS)
      IF(NEG.EQ.0) AMP(NFILT-I+1)=H(I)/H(NFCNS)
      IF(NEG.EQ.1) AMP(NFILT+1-I)=-H(I)/H(NFCNS)
342   CONTINUE
      IF(NEG.EQ.1 .AND. NODD.EQ.1) AMP(NZ)=0.D0
C
C ADD SAWCAD PARAMETERS NORMALIZED TO 1 MHZ
C 2 Fo SAMPLING
C
      NUM=NFILT
      NFFT=NFILT
      ITYPE=-1
      FO=0.5
      TFLO=-(NFILT-1)/2
      TFHI=(NFILT-1)/2
C
C PROGRAM OUTPUT SECTION.
C
C
      PRINT 360
360   FORMAT(//70(1H*)//25X,'FINITE IMPULSE RESPONSE (FIR)'/
     1        25X,'LINEAR PHASE DIGITAL FILTER DESIGN'/
     2        25X,'REMEZ EXCHANGE ALGORITHM'/)
      IF(JTYPE.EQ.1) PRINT 365
365   FORMAT(25X,'BANDPASS FILTER'/)
      IF(JTYPE.EQ.2) PRINT 370
370   FORMAT(25X,'DIFFERENTIATOR'/)
      IF(JTYPE.EQ.3) PRINT 375
375   FORMAT(25X,'HILBERT TRANSFORMER'/)
      PRINT 378,NFILT
378   FORMAT(20X,'FILTER LENGTH = ',I3/)
      PRINT 380
380   FORMAT(20X,'***** IMPULSE RESPONSE *****')
      DO 381 J=1,NFCNS
      K=NFILT+1-J
      IF(NEG.EQ.0) PRINT 382,J,H(J),K
      IF(NEG.EQ.1) PRINT 383,J,H(J),K
381   CONTINUE
382   FORMAT(20X,'H(',I3,') = ',E15.8,' = H(',I4,')')
```

```
383 FORMAT(20X,'H(',I3,') = ',E15.8,'= -H(',I4,')')
    IF(NEG.EQ.1.AND.NODD.EQ.1) PRINT 384,NZ
384 FORMAT(20X,'H(',I3,') = 0.0')
    DO 450 K=1,NBANDS,4
    KUP=K+3
    IF(KUP.GT.NBANDS) KUP=NBANDS
    PRINT 385,(J,J=K,KUP)
385 FORMAT(/24X,4('BAND',I3,8X))
    PRINT 390,(EDGE(2*J-1),J=K,KUP)
390 FORMAT(2X,'LOWER BAND EDGE',5F15.9)
    PRINT 395,(EDGE(2*J),J=K,KUP)
395 FORMAT(2X,'UPPER BAND EDGE',5F15.9)
    IF(JTYPE.NE.2) PRINT 400,(FX(J),J=K,KUP)
400 FORMAT(2X,'DESIRED VALUE',2X,5F15.9)
    IF(JTYPE.EQ.2) PRINT 405,(FX(J),J=K,KUP)
405 FORMAT(2X,'DESIRED SLOPE',2X,5F15.9)
    PRINT 410,(WTX(J),J=K,KUP)
410 FORMAT(2X,'WEIGHTING',6X,5F15.9)
    DO 420 J=K,KUP
420 DEVIAT(J)=DEV/WTX(J)
    PRINT 425,(DEVIAT(J),J=K,KUP)
425 FORMAT(2X,'DEVIATION',6X,5F15.9)
    IF(JTYPE.NE.1) GO TO 450
    DO 430 J=K,KUP
430 DEVIAT(J)=20.0*ALOG10(DEVIAT(J))
    PRINT 435,(DEVIAT(J),J=K,KUP)
435 FORMAT(2X,'DEVIATION IN DB',5F15.9)
450 CONTINUE
    PRINT 455,(GRID(IEXT(J)),J=1,NZ)
455 FORMAT(/2X,'EXTREMAL FREQUENCIES'/(2X,5F12.7))
    PRINT 460
460 FORMAT(/1X,70(1H*)/1H1)
C
    RETURN
    END
C
    FUNCTION EFF(TEMP,FX,WTX,LBAND,JTYPE)
c
c function to calculate the desired magnitude response
c as a function of frequency.
c
    DIMENSION FX(5),WTX(5)
    IF(JTYPE.EQ.2) GO TO 1
    EFF=FX(LBAND)
    RETURN
1   EFF=FX(LBAND)*TEMP
    RETURN
    END
c
c
c
    FUNCTION WATE(TEMP,FX,WTX,LBAND,JTYPE)
c
```

```
c function to calculate the weight function as a function
c of frequency.
c
      DIMENSION FX(5),WTX(5)
      IF(JTYPE.EQ.2) GO TO 1
      WATE=WTX(IBAND)
      RETURN
1     IF(FX(IBAND).LT.0.0001) GO TO 2
      WATE=WTX(IBAND)/TEMP
      RETURN
2     WATE=WTX(IBAND)
      RETURN
      END
c
c
c
      SUBROUTINE ERROR
      PRINT 1
1     FORMAT(' ********** ERROR IN INPUT DATA **********')
      STOP
      END
c
c
c
      SUBROUTINE REMEZ(EDGE,NBANDS,MOREN)
c
c this subroutine implements the remez exchange algorithm
c for the weighted chebychev approximation of a continuous
c function with a sum of cosines.  inputs to the
subroutine
c are a dense grid which replaces the frequency axis, the
c desired function on this grid. the weight function on
this
c grid, the number of cosines, and an initial guess of the
c extremal frequencies. the program minimizes the
chebychev
c error by determining the best location of the extremal
c frequencies (points of maximum error) and then
calculates
c the coefficients of the best approximation.
      COMMON
PI2,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
      DIMENSION EDGE(20)
      DIMENSION IEXT(514),AD(514),ALPHA(514),X(514),Y(514)
      DIMENSION DES(8224),GRID(8224),WT(8224)
      DIMENSION A(514),P(513),Q(513)
      DOUBLE PRECISION PI2,DNUM,DDEN,DTEMP,A,P,Q
      DOUBLE PRECISION AD,DEV,X,Y
      LOGICAL MOREN
C
C THE PROGRAM ALLOWS A MAXIMUM NUMBER OF ITERATIONS OF 25
C
      MOREN=.FALSE.
```

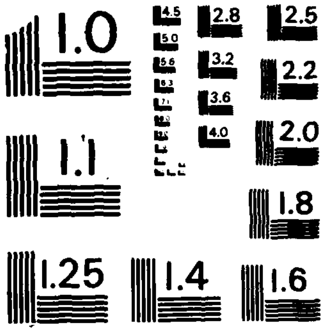
```
      ITRMAX=25
      DEVL=-1.0
      NZ=NFCNS+1
      NZZ=NFCNS+2
      NITER=0
  100 CONTINUE
      IEXT(NZZ)=NGRID+1
      NITER=NITER+1
      IF(NITER.GT.ITRMAX) GO TO 400
      DO 110 J=1,NZ
      DTEMP=GRID(IEXT(J))
      DTEMP=DCOS(DTEMP*PI2)
  110 X(J)=DTEMP
      JET=(NFCNS-1)/15+1
      DO 120 J=1,NZ
  120 AD(J)=D(J,NZ,JET)
      DNUM=0.0
      DDEN=0.0
      K=1
      DO 130 J=1,NZ
      L=IEXT(J)
      DTEMP=AD(J)*DES(L)
      DNUM=DNUM+DTEMP
      DTEMP=K*AD(J)/WT(L)
      DDEN=DDEN+DTEMP
  130 K=-K
      DEV=DNUM/DDEN
      NU=1
      IF(DEV.GT.0.0) NU=-1
      DEV=-NU*DEV
      K=NU
      DO 140 J=1,NZ
      L=IEXT(J)
      DTEMP=K*DEV/WT(L)
      Y(J)=DES(L)+DTEMP
  140 K=-K
      IF(DEV.GE.DEVL) GO TO 150
      MOREN=.TRUE.
      RETURN
  150 DEVL=DEV
      JCHANGE=0
      K1=IEXT(1)
      KNZ=IEXT(NZ)
      KLOW=0
      NUT=-NU
      J=1
C
C SEARCH FOR THE EXTREMAL FREQUENCIES OF THE BEST
C APPROXIMATION
C
  200 IF(J.EQ.NZZ) YNZ=COMP
      IF(J.GE.NZZ) GO TO 300
      KUP=IEXT(J+1)
```

```
      L=IEXT(J)+1
      NUT=-NUT
      IF(J.EQ.2) Y1=COMP
      COMP=DEV
      IF(L.GE.KUP) GO TO 220
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=NUT*ERR-COMP
      IF(DTEMP.LE.0.0) GO TO 220
      COMP=NUT*ERR
  210 L=L+1
      IF(L.GE.KUP) GO TO 215
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=NUT*ERR-COMP
      IF(DTEMP.LE.0.0) GO TO 215
      COMP=NUT*ERR
      GO TO 210
  215 IEXT(J)=L-1
      J=J+1
      KLOW=L-1
      JCHNGE=JCHNGE+1
      GO TO 200
  220 L=L-1
  225 L=L-1
      IF(L.LE.KLOW) GO TO 250
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=NUT*ERR-COMP
      IF(DTEMP.GT.0.0) GO TO 230
      IF(JCHNGE.LE.0) GO TO 225
      GO TO 260
  230 COMP=NUT*ERR
  235 L=L-1
      IF(L.LE.KLOW) GO TO 240
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=NUT*ERR-COMP
      IF(DTEMP.LE.0.0) GO TO 240
      COMP=NUT*ERR
      GO TO 235
  240 KLOW=IEXT(J)
      IEXT(J)=L+1
      J=J+1
      JCHNGE=JCHNGE+1
      GO TO 200
  250 L=IEXT(J)+1
      IF(JCHNGE.GT.0) GO TO 215
  255 L=L+1
      IF(L.GE.KUP) GO TO 260
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=NUT*ERR-COMP
```

```
      IF(DTEMP.LE.0.0) GO TO 255
      COMP=NUT*ERR
      GO TO 210
  260 KLOW=IEXT(J)
      J=J+1
      GO TO 200
  300 IF(J.GT.NZZ) GO TO 320
      IF(K1.GT.IEXT(1)) K1=IEXT(1)
      IF(KNZ.LT.IEXT(NZ)) KNZ=IEXT(NZ)
      NUT1=NUT
      NUT=-NU
      L=0
      KUP=K1
      COMP=YNZ*(1.00001)
      LUCK=1
  310 L=L+1
      IF(L.GE.KUP) GO TO 315
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=NUT*ERR-COMP
      IF(DTEMP.LE.0.0) GO TO 310
      COMP=NUT*ERR
      J=NZZ
      GO TO 210
  315 LUCK=6
      GO TO 325
  320 IF(LUCK.GT.9) GO TO 350
      IF(COMP.GT.Y1) Y1=COMP
      K1=IEXT(NZZ)
  325 L=NGRID+1
      KLOW=KNZ
      NUT=-NUT1
      COMP=Y1*(1.00001)
  330 L=L-1
      IF(L.LE.KLOW) GO TO 340
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=NUT*ERR-COMP
      IF(DTEMP.LE.0.0) GO TO 330
      J=NZZ
      COMP=NUT*ERR
      LUCK=LUCK+10
      GO TO 235
  340 IF(LUCK.EQ.6) GO TO 370
      DO 345 J=1,NFCNS
  345 IEXT(NZZ-J)=IEXT(NZ-J)
      IEXT(1)=K1
      GO TO 100
  350 KN=IEXT(NZZ)
      DO 360 J=1,NFCNS
  360 IEXT(J)=IEXT(J+1)
      IEXT(NZ)=KN
      GO TO 100
```

AD-A170 855 A ZERO EXTRACTION AND SEPARATION TECHNIQUE FOR SURFACE ACOUSTIC WAVE AND..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH K V LINDSAY 1986 2/2

UNCLASSIFIED AFIT/CI/NR-86-93T F/G 9/1 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
370 IF(JCHNGE.GT.0) GO TO 100
C
C CALCULATION OF THE COEFFICIENTS OF THE BEST
APPROXIMATION
C USING THE INVERSE DISCRETE FOURIER TRANSFORM
C
400 CONTINUE
    NM1=NFCNS-1
    FSH=1.0E-6
    GTEMP=GRID(1)
    X(NZZ)=-2.0
    CN=2*NFCNS-1
    DELF=1.0/CN
    L=1
    KKK=0
    IF(EDGE(1).EQ.0.0.AND.EDGE(2*NBANDS).EQ.5) KKK=1
    IF(NFCNS.LE.3) KKK=1
    IF(KKK.EQ.1) GO TO 405
    DTEMP=DCOS(PI2*GRID(1))
    DNUM=DCOS(PI2*GRID(NGRID))
    AA=2.0/(DTEMP-DNUM)
    BB=-(DTEMP+DNUM)/(DTEMP-DNUM)
405 CONTINUE
    DO 430 J=1,NFCNS
    FT=(J-1)*DELF
    XT=DCOS(PI2*FT)
    IF(KKK.EQ.1) GO TO 410
    XT=(XT-BB)/AA
    FT=ACOS(XT)/PI2
410 XE=X(L)
    IF(XT.GT.XE) GO TO 420
    IF((XE-XT).LT.FSH) GO TO 415
    L=L+1
    GO TO 410
415 A(J)=Y(L)
    GO TO 425
420 IF((XT-XE).LT.FSH) GO TO 415
    GRID(1)=FT
    A(J)=GEE(1,NZ)
425 CONTINUE
    IF(L.GT.1) L=L-1
430 CONTINUE
    GRID(1)=GTEMP
    DDEN=PI2/CN
    DO 510 J=1,NFCNS
    DTEMP=0.0
    DNUM=(J-1)*DDEN
    IF(NM1.LT.1) GO TO 505
    DO 500 K=1,NM1
500 DTEMP=DTEMP+A(K+1)*DCOS(DNUM*K)
505 DTEMP=2.0*DTEMP+A(1)
510 ALPHA(J)=DTEMP
    DO 550 J=2,NFCNS
```

```
550 ALPHA(J)=2*ALPHA(J)/CN
    ALPHA(1)=ALPHA(1)/CN
    IF(KKK.EQ.1) GO TO 545
    P(1)=2.0*ALPHA(NFCNS)*BB+ALPHA(NM1)
    P(2)=2.0*AA*ALPHA(NFCNS)
    Q(1)=ALPHA(NFCNS-2)-ALPHA(NFCNS)
    DO 540 J=2,NM1
    IF(J.LT.NM1) GO TO 515
    AA=0.5*AA
    BB=0.5*BB
515 CONTINUE
    P(J+1)=0.0
    DO 520 K=1,J
    A(K)=P(K)
520 P(K)=2.0*BB*A(K)
    P(2)=P(2)+A(1)*2.0*AA
    JM1=J-1
    DO 525 K=1,JM1
525 P(K)=P(K)+Q(K)+AA*A(K+1)
    JP1=J+1
    DO 530 K=3,JP1
530 P(K)=P(K)+AA*A(K-1)
    IF(J.EQ.NM1) GO TO 540
    DO 535 K=1,J
535 Q(K)=-A(K)
    Q(1)=Q(1)+ALPHA(NFCNS-1-J)
540 CONTINUE
    DO 543 J=1,NFCNS
543 ALPHA(J)=P(J)
545 CONTINUE
    IF(NFCNS.GT.3) RETURN
    ALPHA(NFCNS+1)=0.0
    ALPHA(NFCNS+2)=0.0
    RETURN
    END
C
C
C
    DOUBLE PRECISION FUNCTION D(K,N,M)
C
C FUNCTION TO CALCULATE THE LAGRANGE INTERPOLATION
C COEFFICIENTS FOR USE IN THE FUNCTION GEE.
C
    COMMON
PI2,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
    DIMENSION IEXT(514),AD(514),ALPHA(514),X(514),Y(514)
    DIMENSION DES(8224),GRID(8224),WT(8224)
    DOUBLE PRECISION AD,DEV,X,Y
    DOUBLE PRECISION Q
    DOUBLE PRECISION PI2
    D=1.0
    Q=X(K)
    DO 3 L=1,M
```

```
      DO 2 J=L,N,M
      IF(J-K)1,2,1
1     D=2.0*D*(Q-X(J))
2     CONTINUE
3     CONTINUE
      D=1.0/D
      RETURN
      END
C
C
C
C
      DOUBLE PRECISION FUNCTION GEE(K,N)
C
C FUNCTION TO EVALUATE THE FREQUENCY RESPONSE USING THE
C LAGRANGE INTERPOLATION FORMULA IN THE BARYCENTRIC FORM
C
      COMMON
PI2,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
      DIMENSION IEXT(514),AD(514),ALPHA(514),X(514),Y(514)
      DIMENSION DES(8224),GRID(8224),WT(8224)
      DOUBLE PRECISION P,C,D,XF
      DOUBLE PRECISION PI2
      DOUBLE PRECISION AD,DEV,X,Y
      P=0.0
      XF=GRID(K)
      XF=DCOS(PI2*XF)
      D=0.0
      DO 1 J=1,N
      C=XF-X(J)
      C=AD(J)/C
      D=D+C
1     P=P+C*Y(J)
      GEE=P/D
      RETURN
      END
C
C
C
C
      SUBROUTINE OUCH
      PRINT 1
1     FORMAT(' ************ FAILURE TO CONVERGE
*************'/
      1        '0PROBABLE CAUSE IS MACHINE ROUNDING ERROR'/
      2        '0THE IMPULSE RESPONSE MAY BE CORRECT'/
      3        '0CHECK WITH A FREQUENCY RESPONSE')
      RETURN
      END
```

APPENDIX B

ZERO EXTRACTION PROGRAM

```
C
C  RPOLY ROUTINE BY M.A. JENKINS
C  ACM TOMS VOL 1 NO 2 JUNE 1975
C
      SUBROUTINE RPOLY
C FINDS THE ZEROS OF A REAL POLYNOMIAL
C OP - DOUBLE PRECISION VECTOR OF COEFFICIENTS IN
C      ORDER OF DECREASING POWERS.
C DEGREE - INTEGER DEGREE OF THE POLYNOMIAL.
C ZEROR, ZEROI - OUTPUT DOUBLE PRECISION VECTORS OF
C                REAL AND IMAGINARY PARTS OF THE
C                ZEROS.
C FAIL - OUTPUT LOGICAL PARAMETER, TRUE ONLY IF
C        LEADING COEFFICIENT IS ZERO OR IF RPOLY
C        HAS FOUND FEWER THAN DEGREE ZEROS.
C        IN THE LATTER CASE, DEGREE IS RESET TO
C        THE NUMBER OF ZEROS FOUND.
C TO CHANGE THE SIZE OF POLYNOMIALS WHICH CAN BE
C SOLVED, RESET THE DIMENSIONS OF THE ARRAYS IN THE
C COMMON AREA AND IN THE FOLLOWING DECLARATIONS
C FOR SCALING, BOUNDS AND ERROR CALCULATIONS. ALL
C CALCULATIONS FOR THE ITERATIONS ARE DONE IN
C DOUBLE PRECISION.
      COMMON/FILE/ AMP(4096),PHASE(4096),NFFT,ITYPE
      COMMON/DAT/ FO,TFLO,TFHI,NUM
      COMMON /RPOLY/ P, QP, K, QK, SVK, SR, SI, U,
     1 V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,
     2 H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN
      DOUBLE PRECISION P(1024), QP(1024), K(1024),
     1 QK(1024), SVK(1024), SR, SI, U, V, A, B, C, D,
     2 A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,
     3 LZR, LZI
      REAL ETA, ARE, MRE
      INTEGER N, NN
C OP IS DIMENSIONED TO 4096 SINCE IT TAKES ITS ARRAY
C FROM SAWCAD'S AMP ARRAY.
      DOUBLE PRECISION OP(4096), TEMP(1024),
     1 ZEROR(1024), ZEROI(1024), T, AA, BB, CC, DABS,
     2 FACTOR
      REAL PT(1024), LO, MAX, MIN, XX, YY, COSR,
     1 SINR, XXX, X, SC, BND, XM, FF, DF, DX, INFIN,
     2 SMALNO, BASE
      INTEGER DEGREE, CNT, NZ, I, J, JJ, NM1
      LOGICAL FAIL, ZEROK
C THE FOLLOWING STATEMENTS SET MACHINE CONSTANTS USED
C IN THE VARIOUS PARTS OF THE PROGRAM. THE MEANING OF THE
C FOUR CONSTANTS ARE...
C ETA    THE MAXIMUM RELATIVE REPRESENTATION ERROR
C        WHICH CAN BE DESCRIBED AS THE SMALLEST
C        POSITIVE FLOATING POINT NUMBER SUCH THAT
C        1.D0+ETA IS GREATER THAN 1.
C INFIN  THE LARGEST FLOATING POINT NUMBER.
C SMALNO THE SMALLEST POSITIVE FLOATING POINT
```

```
C        NUMBER IF THE EXPONENT RANGE DIFFERS IN SINGLE
C        AND DOUBLE PRECISION THEN SMALNO AND INFIN
C        SHOULD INDICATE THE SMALLER RANGE.
C BASE   THE BASE OF THE FLOATING POINT NUMBER
C        SYSTEM USED.
C THE VALUES BELOW CORRESPOND TO THE VAX 11-750
      BASE=2.
      ETA=1.387779E-17
      INFIN=1.7E38
      SMALNO=5.9E-39
C ARE AND MRE REFER TO THE UNIT ERROR IN + AND *
C RESPECTIVELY. THEY ARE ASSUMED TO BE THE SAME AS
C ETA
      ARE=ETA
      MRE=ETA
      LO=SMALNO/ETA
C
C INITIALIZE ARRAYS
C
      IF (ITYPE.EQ.1) THEN
              PRINT *,'THIS IS A FREQUENCY FILE!'
              PRINT *,'EXPECTED AN IMPULSE RESPONSE FILE'
              RETURN
      END IF
      DEGREE=NUM-1
      DO 233 I=1,DEGREE+1
      OP(I)=AMP(I)
      AMP(I)=0.0D0
      PHASE(I)=0.0D0
      ZEROR(I)=0.0D0
      ZEROI(I)=0.0D0
233 CONTINUE
C
C INITIALIZATION OF CONSTANTS FOR SHIFT ROTATION
C
      XX=0.70710678
      YY=-XX
      COSR=-.069756474
      SINR=.99756405
      FAIL=.FALSE.
      N=DEGREE
      NN=N+1
C ALGORITHM FAILS IF THE LEADING COEFFICIENT IS ZERO.
      IF (OP(1).NE.0.D0) GO TO 10
      FAIL=.TRUE.
      DEGREE=0
      PRINT *,'LEADING COEFFICIENT IS ZERO -- NOT ALLOWED'
      RETURN
C REMOVE THE ZEROS AT THE ORIGIN IF ANY
10    IF(OP(NN).NE.0.0D0) GO TO 20
      J=DEGREE-N+1
      ZEROR(J)=0.0D0
      amp(j)=0.0d0
```

```
      ZEROI(J)=0.0D0
      phase(j)=0.0d0
      NN=NN-1
      N=N-1
      GO TO 10
C MAKE A COPY OF THE COEFFICIENTS
20    DO 30 I=1,NN
      P(I)=OP(I)
30    CONTINUE
C START THE ALGORITHM FOR ONE ZERO
40    IF(N.GT.2)GO TO 60
      IF(N.LT.1)RETURN
C CALCULATE THE FINAL ZERO OR PAIR OF ZEROS.
      IF(N.EQ.2)GO TO 50
      ZEROR(DEGREE)=-P(2)/P(1)
      ZEROI(DEGREE)=0.0D0
      AMP(DEGREE)=ZEROR(DEGREE)
      PHASE(DEGREE)=ZEROI(DEGREE)
      RETURN
50    CALL QUAD(P(1), P(2), P(3), ZEROR(DEGREE-1),
     1 ZEROI(DEGREE-1), ZEROR(DEGREE), ZEROI(DEGREE))
      amp(degree-1)=zeror(degree-1)
      phase(degree-1)=zeroi(degree-1)
      AMP(DEGREE)=ZEROR(DEGREE)
      PHASE(DEGREE)=ZEROI(DEGREE)
      RETURN
C FIND THE LARGEST AND SMALLEST MODULI OF COEFFICIENTS
60    MAX=0.
      MIN=INFIN
      DO 70 I=1,NN
      X=ABS(SNGL(P(I)))
      IF(X.GT.MAX) MAX=X
      IF(X.NE.0. .AND. X.LT.MIN) MIN=X
70    CONTINUE
C SCALE IF THERE ARE LARGE OR VERY SMALL COEFFICIENTS
C COMPUTES A SCALE FACTOR TO MULTIPLY THE
C COEFFICIENTS OF THE POLYNOMIAL. THE SCALING IS DONE
C TO AVOID OVERFLOW AND TO AVOID UNDETECTED UNDERFLOW
C INTERFERING WITH THE CONVERGENCE CRITERION.
C THE FACTOR IS A POWER OF THE BASE.
      SC=LO/MIN
      IF(SC.GT.1.0)GO TO 80
      IF(MAX.LT.10.)GO TO 110
      IF(SC.EQ.0.) SC=SMALNO
      GO TO 90
80    IF(INFIN/SC.LT.MAX)GO TO 110
90    L=ALOG(SC)/ALOG(BASE)+0.5
      FACTOR=(BASE*1.0D0)**L
      IF(FACTOR.EQ.1.D0)GO TO 110
      DO 100 I=1,NN
      P(I)=FACTOR*P(I)
100   CONTINUE
C COMPUTE LOWER BOUND ON MODULI OF ZEROS.
```

```
110 DO 120 I=1,NN
    PT(I)=ABS(SNGL(P(I)))
120 CONTINUE
    PT(NN)=-PT(NN)
C COMPUTE UPPER ESTIMATE OF BOUND
    X=EXP((ALOG(-PT(NN))-ALOG(PT(1)))/FLOAT(N))
    IF(PT(N).EQ.0.)GO TO 130
C IF NEWTON STEP AT THE ORIGIN IS BETTER, USE IT!
    XM=-PT(NN)/PT(N)
    IF(XM.LT.X)X=XM
C CHOP THE INTERVAL (0,X) UNTIL FF .LE. 0
130 XM=X*.1
    FF=PT(1)
    DO 140 I=2,NN
    FF=FF*XM+PT(I)
140 CONTINUE
    IF(FF.LE.0.)GO TO 150
    X=XM
    GO TO 130
150 DX=X
C DO NEWTON ITERATION UNTIL X CONVERGES TO TWO
C DECIMAL PLACES
160 IF(ABS(DX/X).LE..005)GO TO 180
    FF=PT(1)
    DF=FF
    DO 170 I=2,N
    FF=FF*X+PT(I)
    DF=DF*X+FF
170 CONTINUE
    FF=FF*X+PT(NN)
    DX=FF/DF
    X=X-DX
    GO TO 160
180 BND=X
C COMPUTE THE DERIVATIVE AS THE INITIAL K POLYNOMIAL
C AND DO 5 STEPS WITH NO SHIFT
    NM1=N-1
    DO 190 I=2,N
    K(I)=FLOAT(NN-I)*P(I)/FLOAT(N)
190 CONTINUE
    K(1)=P(1)
    AA=P(NN)
    BB=P(N)
    ZEROK=K(N).EQ.0.D0
    DO 230 JJ=1,5
    CC=K(N)
    IF(ZEROK)GO TO 210
C USE SCALED FORM OF RECURRENCE IF VALUE OF K AT 0 IS
C NONZERO
    T=-AA/CC
    DO 200 I=1,NM1
    J=NN-I
    K(J)=T*K(J-1)+P(J)
```

```
200 CONTINUE
    K(1)=P(1)
    ZEROK=DABS(K(N)).LE.DABS(BB)*ETA*10.
    GO TO 230
C USE THE UNSCALED FORM OF RECURRENCE
210 DO 220 I=1,NM1
    J=NN-I
    K(J)=K(J-1)
220 CONTINUE
    K(1)=0.D0
    ZEROK=K(N).EQ.0.D0
230 CONTINUE
C SAVE K FOR RESTARTS WITH NEW SHIFTS
    DO 240 I=1,N
    TEMP(I)=K(I)
240 CONTINUE
C LOOP TO SELECT THE QUADRATIC CORRESPONDING TO EACH
C NEW SHIFT
         DO 280 CNT=1,20
C QUADRATIC RESPONDS TO A DOUBLE SHIFT TO A
C NON-REAL POINT AND ITS COMPLEX CONJUGATE. THE POINT
C HAS MODULUS BND AND AMPLITUDE ROTATED BY 94 DEGREES
C FROM THE PREVIOUS SHIFT
    XXX=COSR*XX-SINR*YY
    YY=SINR*XX+COSR*YY
    XX=XXX
    SR=BND*XX
    SI=BND*YY
    U=-2.0D0*SR
    V=BND
C SECOND STAGE CALCULATION, FIXED QUADRATIC
    CALL FXSHFR(20*CNT,NZ)
    IF(NZ.EQ.0)GO TO 260
C THE SECOND STAGE JUMPS DIRECTLY TO ONE OF THE THIRD
C STAGE ITERATIONS AND RETURNS HERE IF SUCCESSFUL.
C DEFLATE THE POLYNOMIAL, STORE THE ZERO OR ZEROS AND
C RETURN TO THE MAIN ALGORITHM.
    J=DEGREE-N+1
    ZEROR(J)=SZR
    AMP(J)=SZR
    ZEROI(J)=SZI
    PHASE(J)=SZI
    NN=NN-NZ
    N=NN-1
    DO 250 I=1,NN
    P(I)=QP(I)
250 CONTINUE
    IF(NZ.EQ.1) GO TO 40
    ZEROR(J+1)=LZR
    ZEROI(J+1)=LZI
    AMP(J+1)=LZR
    PHASE(J+1)=LZI
    GO TO 40
```

```
C IF THE ITERATION IS UNSUCCESSFUL ANOTHER QUADRATIC
C IS CHOSEN AFTER RESTORING K
260 DO 270 I=1,N
    K(I)=TEMP(I)
270 CONTINUE
280 CONTINUE
C RETURN WITH FAILURE IF NO CONVERGENCE WITH 20
C SHIFTS.
    FAIL=.TRUE.
    DEGREE=DEGREE-N
    RETURN
    END
        SUBROUTINE FXSHFR(L2, NZ)
C COMPUTES UP TO L2 FIXED SHIFT K-POLYNOMIALS,
C TESTING FOR CONVERGENCE IN THE LINEAR OR QUADRATIC
C CASE. INITIATES ONE OF THE VARIABLE SHIFT
C ITERATIONS AND RETURNS WITH THE NUMBER OF ZEROS
C FOUND.
C L2 - LIMIT OF FIXED SHIFT STEPS.
C NZ - NUMBER OF ZEROS FOUND.
    COMMON /RPOLY/ P, QP, K, QK, SVK, SR, SI, U,
    1 V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,
    2 H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN
    DOUBLE PRECISION P(1024), QP(1024), K(1024),
    1 QK(1024), SVK(1024), SR, SI, U, V, A, B, C, D,
    2 A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,
    3 LZR, LZI
    REAL ETA, ARE, MRE
    INTEGER N, NN
    DOUBLE PRECISION SVU, SVV, UI, VI, S
    REAL BETAS, BETAV, OSS, OVV, SS, VV, TS, TV,
    1 OTS, OTV, TVV, TSS
    INTEGER L2, NZ, TYPE, I, J, IFLAG
    LOGICAL VPASS, SPASS, VTRY, STRY
    NZ=0
    BETAV=.25
    BETAS=.25
    OSS=SR
    OVV=V
C EVALUATE POLYNOMIAL BY SYNTHETIC DIVISION
        CALL QUADSD(NN,U,V,P,QP,A,B)
        CALL CALCSC(TYPE)
        DO 80 J=1,L2
C CALCULATE NEXT K POLYNOMIAL AND ESTIMATE V
        CALL NEXTK(TYPE)
        CALL CALCSC(TYPE)
        CALL NEWEST(TYPE, UI,VI)
        VV=VI
C ESTIMATE S
        SS=0.
        IF(K(N).NE.0.D0) SS=-P(NN)/K(N)
        TV=1.
        TS=1.
```

```
      IF(J.EQ.1 .OR. TYPE.EQ.3) GO TO 70
C COMPUTE RELATIVE MEASURES OF CONVERGENCE OF S AND V
C SEQUENCES
      IF(VV.NE.0.)TV=ABS((VV-OVV)/VV)
      IF(SS.NE.0.)TS=ABS((SS-OSS)/SS)
C IF DECREASING, MULTIPLY TWO MOST RECENT
C CONVERGENCE MEASURES
      TVV=1.
      IF(TV.LT.OTV)TVV=TV*OTV
      TSS=1.
      IF(TS.LT.OTS)TSS=TS*OTS
C COMPARE WITH CONVERGENCE CRITERIA
      VPASS=TVV.LT.BETAV
      SPASS=TSS.LT.BETAS
      IF(.NOT.(SPASS .OR. VPASS)) GO TO 70
C AT LEAST ONE SEQUENCE HAS PASSED THE CONVERGENCE
C TEST. STORE VARIABLES BEFORE ITERATING.
      SVU=U
      SVV=V
      DO 10 I=1,N
      SVK(I)=K(I)
10    CONTINUE
      S=SS
C CHOOSE ITERATION ACCORDING TO THE FASTEST
C CONVERGING SEQUENCE.
      VTRY=.FALSE.
      STRY=.FALSE.
      IF(SPASS .AND. ((.NOT.VPASS) .OR.
    1 TSS.LT.TVV)) GO TO 40
20    CALL QUADIT(UI,VI,NZ)
      IF(NZ.GT.0) RETURN
C QUADRATIC ITERATION HAS FAILED. FLAG THAT IT HAS
C BEEN TRIED AND DECREASE THE CONVERGENCE CRITERION.
      VTRY=.TRUE.
      BETAV=BETAV*.25
C TRY LINEAR ITERATION IF IT HAS NOT BEEN TRIED AND
C THE S SEQUENCE IS CONVERGING.
      IF(STRY .OR. (.NOT.SPASS)) GO TO 50
      DO 30 I=1,N
      K(I)=SVK(I)
30    CONTINUE
40    CALL REALIT(S, NZ, IFLAG)
      IF(NZ.GT.0) RETURN
C LINEAR ITERATION HAS FAILED. FLAG THAT IT HAS BEEN
C TRIED AND DECREASE THE CONVERGENCE CRITERION.
      STRY=.TRUE.
      BETAS=BETAS*.25
      IF(IFLAG.EQ.0) GO TO 50
C IF LINEAR ITERATION SIGNALS AN ALMOST DOUBLE REAL
C ZERO ATTEMPT QUADRATIC ITERATION.
      UI=-(S+S)
      VI=S*S
      GO TO 20
```

```
C RESTORE VARIABLES
50      U=SVU
        V=SVV
        DO 60 I=1,N
        K(I)=SVK(I)
60      CONTINUE
C TRY QUADRATIC ITERATION IF IT HAS NOT BEEN TRIED
C AND THE V SEQUENCE IS CONVERGING.
        IF(VPASS .AND. (.NOT.VTRY)) GO TO 20
C RECOMPUTE QP AND SCALAR VALUES TO CONTINUE THE
C SECOND STAGE.
        CALL QUADSD(NN, U, V, P, QP, A, B)
        CALL CALCSC(TYPE)
70      OVV=VV
        OSS=SS
        OTV=TV
    OTS=TS
80      CONTINUE
        RETURN
        END
        SUBROUTINE QUADIT(UU,VV,NZ)
C VARIABLE-SHIFT K-POLYNOMIAL ITERATION FOR A
C QUADRATIC FACTOR CONVERGES ONLY IF THE ZEROS ARE
C EQUIMODULAR OR NEARLY SO.
C UU, VV  - COEFFICIENTS OF STARTING QUADRATIC
C NZ      - NUMBER OF ZEROS FOUND
    COMMON /RPOLY/ P, QP, K, QK, SVK, SR, SI, U,
    1 V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,
    2 H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN
    DOUBLE PRECISION P(1024), QP(1024), K(1024),
    1 QK(1024), SVK(1024), SR, SI, U, V, A, B, C, D,
    2 A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,
    3 LZR, LZI
    REAL ETA, ARE, MRE
    INTEGER N, NN
    DOUBLE PRECISION UI, VI, UU, VV, DABS
    REAL MS, MP, OMP, EE, RELSTP, T, ZM
    INTEGER NZ, TYPE, I, J
    LOGICAL TRIED
        NZ=0
        TRIED=.FALSE.
        U=UU
        V=VV
        J=0
C MAIN LOOP
10      CALL QUAD(1.D0, U, V, SZR, SZI, LZR, LZI)
C RETURN IF ROOTS OF THE QUADRATIC ARE REAL AND NOT
C CLOSE TO MULTIPLE OR NEARLY EQUAL AND OF OPPOSITE
C SIGN.
        IF(DABS(DABS(SZR)-DABS(LZR)).GT..01D0*
    1 DABS(LZR)) RETURN
C EVALUATE POLYNOMIAL BY QUADRATIC SYNTHETIC DIVISION.
        CALL QUADSD(NN, U, V, P, QP, A, B)
```

```
      MP=DABS(A-SZR*B)+DABS(SZI*B)
C COMPUTE A RIGOROUS BOUND ON THE ROUNDING ERROR IN
C EVALUATING P
      ZM=SQRT(ABS(SNGL(V)))
      EE=2.*ABS(SNGL(QP(1)))
      T=-SZR*B
      DO 20 I=2,N
      EE=EE*ZM+ABS(SNGL(QP(I)))
20    CONTINUE
      EE=EE*ZM+ABS(SNGL(A)+T)
      EE=(5.*MRE+4.*ARE)*EE-(5.*MRE+2.*ARE)*
     1 (ABS(SNGL(A)+T)+ABS(SNGL(B))*ZM)+
     2 2.*ARE*ABS(T)
C ITERATION HAS CONVERGED SUFFICIENTLY IF THE
C POLYNOMIAL VALUE IS LESS THAN 20 TIMES THIS BOUND
      IF(MP.GT.20.*EE)GO TO 30
      NZ=2
      RETURN
30    J=J+1
C STOP ITERATION AFTER 20 STEPS
      IF(J.GT.20)RETURN
      IF(J.LT.2)GO TO 50
      IF(RELSTP.GT..01 .OR. MP.LT.OMP .OR. TRIED)
     1 GO TO 50
C ACLUSTER APPEARS TO BE STALLING THE CONVERGENCE.
C FIVE FIXED SHIFT STEPS ARE TAKEN WITH A U,V CLOSE
C TO THE CLUSTER.
      IF(RELSTP.LT.ETA) RELSTP=ETA
      RELSTP=SQRT(RELSTP)
      U=U-U*RELSTP
      V=V+V*RELSTP
      CALL QUADSD(NN,U,V,P,QP,A,B)
      DO 40 I=1,5
      CALL CALCSC(TYPE)
      CALL NEXTK(TYPE)
40    CONTINUE
      TRIED=.TRUE.
      J=0
50    OMP=MP
C CALCULATE NEXT K POLYNOMIAL AND NEW U AND V
      CALL CALCSC(TYPE)
      CALL NEXTK(TYPE)
      CALL CALCSC(TYPE)
      CALL NEWEST(TYPE, UI, VI)
C IF VI IS ZERO THE ITERATION IS NOT CONVERGING.
      IF(VI.EQ.0.D0) RETURN
      RELSTP=DABS((VI-V)/VI)
      U=UI
      V=VI
      GO TO 10
      END
      SUBROUTINE REALIT(SSS, NZ, IFLAG)
C VARIABLE SHIFT H POLYNOMIAL ITERATION FOR A REAL
```

```
C ZERO.
C SSS   - STARTING ITERATE
C NZ    - NUMBER OF ZERO FOUND
C IFLAG- FLAG TO INDICATE A PAIR OF ZEROS NEAR REAL
C        AXIS.
      COMMON /RPOLY/ P, QP, K, QK, SVK, SR, SI, U,
      1 V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,
      2 H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN
      DOUBLE PRECISION P(1024), QP(1024), K(1024),
      1 QK(1024), SVK(1024), SR, SI, U, V, A, B, C, D,
      2 A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,
      3 LZR, LZI
      REAL ETA, ARE, MRE
      INTEGER N, NN
      DOUBLE PRECISION PV, KV, T, S, SSS, DABS
      REAL MS, MP, OMP, EE
      INTEGER NZ, IFLAG, I, J, NM1
      NM1=N-1
          NZ=0
          S=SSS
          IFLAG=0
          J=0
C MAIN LOOP
10        PV=P(1)
C EVALUATE P AT S
          QP(1)=PV
          DO 20 I=2,NN
          PV=PV*S+P(I)
          QP(I)=PV
20        CONTINUE
          MP=DABS(PV)
C COMPUTE A RIGOROUS BOUND ON THE ERROR IN EVALUATING
C P
          MS=DABS(S)
          EE=(MRE/(ARE+MRE))*ABS(SNGL(QP(1)))
          DO 30 I=2,NN
          EE=EE*MS+ABS(SNGL(QP(I)))
30        CONTINUE
C ITERATION HAS CONVERGED SUFFICIENTLY IF THE
C POLYNOMIAL VALUE IS LESS THAN 20 TIMES THIS BOUND.
          IF(MP.GT.20.*((ARE+MRE)*EE-MRE*MP))GO TO 40
          NZ=1
          SZR=S
          SZI=0.D0
          RETURN
40        J=J+1
C STOP ITERATION AFTER 10 STEPS
          IF(J.GT.10) RETURN
          IF(J.LT.2) GO TO 50
          IF(DABS(T).GT..001*DABS(S-T) .OR. MP.LE.OMP)
      1 GO TO 50
C A CLUSTER OF ZEROS NEAR THE REAL AXIS HAS BEEN
C ENCOUNTERED. RETURN WITH IFLAG SET TO INITIATE
```

```
C QUADRATIC ITERATION.
        IFLAG=1
        SSS=S
        RETURN
C RETURN IF THE POLYNOMIAL VALUE HAS INCREASED
C SIGNIFICANTLY.
50      OMP=MP
C COMPUTE T, THE NEXT POLYNOMIAL, AND THE NEW ITERATE
        KV=K(1)
        QK(1)=KV
        DO 60 I=2,N
        KV=KV*S+K(I)
        QK(I)=KV
60      CONTINUE
        IF(DABS(KV).LE.DABS(K(N))*10.*ETA)GO TO 80
C USE THE SCALED FORM OF THE RECURRENCE IF THE VALUE
C OF K AT S IS NONZERO
        T=-PV/KV
        K(1)=QP(1)
        DO 70 I=2,N
        K(I)=T*QK(I-1)+QP(I)
70      CONTINUE
        GO TO 100
C USE SCALED FORM
80      K(1)=0.0D0
        DO 90 I=2,N
        K(I)=QK(I-1)
90      CONTINUE
100     KV=K(1)
        DO 110 I=2,N
        KV=KV*S+K(I)
110     CONTINUE
        T=0.D0
        IF(DABS(KV).GT.DABS(K(N))*10.*ETA) T=-PV/KV
        S=S+T
        GO TO 10
        END
        SUBROUTINE CALCSC(TYPE)
C THIS ROUTINE CALCULATES SCALAR QUANTITIES USED TO
C COMPUTE THE NEXT K POLYNOMIAL AND NEW ESTIMATES OF
C THE QUADRATIC COEFFICIENTS.
C TYPE - INTEGER VARIABLE SET HERE INDICATING HOW THE
C         CALCULATIONS ARE NORMALIZED TO AVOID
C         OVERFLOW.
    COMMON /RPOLY/ P, QP, K, QK, SVK, SR, SI, U,
    1 V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,
    2 H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN
    DOUBLE PRECISION P(1024), QP(1024), K(1024),
    1 QK(1024), SVK(1024), SR, SI, U, V, A, B, C, D,
    2 A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,
    3 LZR, LZI
        REAL ETA, ARE, MRE
        INTEGER N, NN
```

```
      DOUBLE PRECISION DABS
      INTEGER TYPE
C SYNTHETIC DIVISION OF K BY THE QUADRATIC 1,U,V
      CALL QUADSD(N, U, V, K, QK, C, D)
      IF(DABS(C).GT.DABS(K(N))*100.*ETA)GO TO 10
      IF(DABS(D).GT.DABS(K(N-1))*100.*ETA) GO TO 10
      TYPE=3
C TYPE=3 INDICATES THE QUADRATIC IS ALMOST A FACTOR
C OF K.
      RETURN
10    IF(DABS(D).LT.DABS(C))GO TO 20
      TYPE=2
C TYPE=2 INDICATES THAT ALL FORMULAS ARE DIVIDED BY D
      E=A/D
      F=C/D
      G=U*B
      H=V*B
      A3=(A+G)*E+H*(B/D)
      A1=B*F-A
      A7=(F+U)*A+H
      RETURN
20    TYPE=1
C TYPE=1 INDICATES THAT ALL FORMULAS ARE DIVIDED BY C.
      E=A/C
      F=D/C
      G=U*E
      H=V*B
      A3=A*E+(H/C+G)*B
      A1=B-A*(D/C)
      A7=A+G*D+H*F
      RETURN
      END
      SUBROUTINE NEXTK(TYPE)
C COMPUTES THE NEXT K POLYNOMIALS USING SCALARS
C COMPUTED IN CALCSC.
   COMMON /RPOLY/ P, QP, K, QK, SVK, SR, SI, U,
   1 V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,
   2 H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN
   DOUBLE PRECISION P(1024), QP(1024), K(1024),
   1 QK(1024), SVK(1024), SR, SI, U, V, A, B, C, D,
   2 A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,
   3 LZR, LZI
      REAL ETA, ARE, MRE
      INTEGER N, NN
      DOUBLE PRECISION TEMP, DABS
      INTEGER TYPE
      IF(TYPE.EQ.3)GO TO 40
      TEMP=A
      IF(TYPE.EQ.1) TEMP=B
      IF(DABS(A1).GT.DABS(TEMP)*ETA*10.) GO TO 20
C IF A1 IS NEARLY ZERO THEN USE A SPECIAL FORM OF THE
C RECURRENCE.
      K(1)=0.D0
```

```
        K(2)=-A7*QP(1)
        DO 10 I=3,N
        K(I)=A3*QK(I-2)-A7*QP(I-1)
10      CONTINUE
        RETURN
C USE SCALED FORM OF THE RECURRENCE.
20      A7=A7/A1
        A3=A3/A1
        K(1)=QP(1)
        K(2)=QP(2)-A7*QP(1)
        DO 30 I=3,N
        K(I)=A3*QK(I-2)-A7*QP(I-1)+QP(I)
30      CONTINUE
        RETURN
C USE UNSCALED FORM OF THE RECURRENCE IF TYPE IS 3
40      K(1)=0.D0
        K(2)=0.D0
        DO 50 I=3,N
        K(I)=QK(I-2)
50      CONTINUE
        RETURN
        END
        SUBROUTINE NEWEST(TYPE, UU, VV)
C COMPUTES NEW ESTIMATES OF THE QUADRATIC COEFFICIENTS
C USING THE SCALARS COMPUTED IN CALCSC.
     COMMON /RPOLY/ P, QP, K, QK, SVK, SR, SI, U,
     1 V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,
     2 H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN
     DOUBLE PRECISION P(1024), QP(1024), K(1024),
     1 QK(1024), SVK(1024), SR, SI, U, V, A, B, C, D,
     2 A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,
     3 LZR, LZI
        REAL ETA, ARE, MRE
        INTEGER N, NN
        DOUBLE PRECISION A4,A5,B1,B2,C1,C2,C3,
     1 C4, TEMP, UU, VV
        INTEGER TYPE
C USE FORMULAS APPROPRIATE TO SETTING TYPE.
        IF(TYPE.EQ.3) GO TO 30
        IF(TYPE.EQ.2) GO TO 10
        A4=A+U*B+H*F
        A5=C+(U+V*F)*D
        GO TO 20
10      A4=(A+G)*F+H
        A5=(F+U)*C+V*D
C EVALUATE NEW QUADRATIC COEFFICIENTS.
20      B1=-K(N)/P(NN)
        B2=-(K(N-1)+B1*P(N))/P(NN)
        C1=V*B2*A1
        C2=B1*A7
        C3=B1*B1*A3
        C4=C1-C2-C3
        TEMP=A5+B1*A4-C4
```

```
      IF(TEMP.EQ.0.D0) GO TO 30
      UU=U-(U*(C3+C2)+V*(B1*A1+B2*A7))/TEMP
      VV=V*(1.+C4/TEMP)
      RETURN
C IF TYPE=3 THE QUADRATIC IS ZEROED.
30    UU=0.D0
      VV=0.D0
      RETURN
      END
      SUBROUTINE QUADSD(NN, U, V, P, Q, A, B)
C DIVIDES P BY THE QUADRATIC 1,U,V PLACING THE
C QUOTIENT IN Q AND THE REMAINDER IN A,B
      DOUBLE PRECISION P(NN), Q(NN), U, V, A, B, C
      INTEGER I
      B=P(1)
      Q(1)=B
      A=P(2)-U*B
      Q(2)=A
      DO 10 I=3,NN
      C=P(I)-U*A-V*B
      Q(I)=C
      B=A
      A=C
10    CONTINUE
      RETURN
      END
C
      SUBROUTINE QUAD(A, B1, C, SR, SI, LR, LI)
C
C CALCULATE THE ZEROS OF THE QUADRATIC A*Z**2+B1*Z+C.
C THE QUADRATIC FORMULA, MODIFIED TO AVOID
C OVERFLOW, IS USED TO FIND THE LARGER ZERO IF THE
C ZEROS ARE REAL AND BOTH ZEROS ARE COMPLEX.
C THE SMALLER REAL ZERO IS FOUND DIRECTLY FROM THE
C PRODUCT OF THE ZEROS C/A.
      DOUBLE PRECISION A, B1, C, SR, SI, LR, LI, B,
     1 D, E, DABS, DSQRT
      IF(A.NE.0.D0) GO TO 20
      SR=0.D0
      IF(B1.NE.0.D0) SR=-C/B1
      LR=0.D0
10    SI=0.D0
      LI=0.D0
      RETURN
20    IF(C.NE.0.D0) GO TO 30
      SR=0.D0
      LR=-B1/A
      GO TO 10
C COMPUTE DISCRIMINATE AVOIDING OVERFLOW.
30    B=B1/2.D0
      IF(DABS(B).LT.DABS(C)) GO TO 40
      E=1.D0-(A/B)*(C/B)
      D=DSQRT(DABS(E))*DABS(B)
```

```
      GO TO 50
40    E=A
      IF(C.LT.0.D0) E=-A
      E=B*(B/DABS(C))-E
      D=DSQRT(DABS(E))*DSQRT(DABS(C))
50    IF(E.LT.0.D0) GO TO 60
C REAL ZEROS.
   IF(B.GE.0.D0) D=-D
      LR=(-B+D)/A
      SR=0.D0
      IF(LR.NE.0.D0) SR=(C/LR)/A
   GO TO 10
C COMPLEX CONJUGATE ZEROS.
60    SR=-B/A
      LR=SR
      SI=DABS(D/A)
      LI=-SI
      RETURN
      END
```

APPENDIX C

OPTIMAL COMBINATION AND RECONSTITUTION PROGRAM

```
C
      SUBROUTINE COMBO
C
C THIS ROUTINE GENERATES ALL POSSIBLE, NON REPEATING
C COMBINATIONS OF N SAMPLES TAKEN K AT A TIME.
C ALGORITHM ADAPTED FROM
C        APPLIED COMBINATORIAL MATHEMATICS
C           PAGE 24
C           POLYA ET AL      1964
C           JOHN WILEY AND SONS, INC, NEW YORK
      COMMON/FILE/ AMP(4096),PHASE(4096),NFFT,ITYPE
      COMMON/DAT/FO,TFLO,TFHI,NUM
      DOUBLE PRECISION ZEROR(1024), ZEROI(1024)
      DOUBLE PRECISION HBESTONE(1024), HBESTTWO(1024)
      INTEGER C(1024), G(1024), A(1024)
      INTEGER I,K,N,T
      INTEGER DEGREE,TOTAL,MONEBEST,MTWOBEST
C
      IF (ITYPE.NE.0) THEN
              PRINT *,'*** NEED TO OBTAIN ZEROS FIRST ***'
              RETURN
      END IF
      DEGREE=NUM-1
      IF (DEGREE.GT.35) THEN
              PRINT *,' '
              PRINT *,'         ****** WARNING ******'
              PRINT *,' COMPUTATION TIME WILL EXCEED 2
HOURS'
              PRINT *,' '
      END IF
      DO 289 I=1,DEGREE
      ZEROR(I)=AMP(I)
      ZEROI(I)=PHASE(I)
289   CONTINUE
      TOTAL=0
      N=DEGREE/2+2
      IF ((FLOAT(N)-FLOAT(DEGREE)/2.0E0).NE.0) THEN
              N=N+1
              ZEROR(DEGREE+1)=0.0D0
              ZEROI(DEGREE+1)=0.0D0
      END IF
C
C INITIALIZE ARRAY C
C
      DO 140 I=1,N
      C(I)=I
140   CONTINUE
C
      DO 5 K=0,N/2
      PRINT *,K,' AT A TIME FOR H1'
      PRINT *,TOTAL,' COMBINATIONS TESTED SO FAR'
      IF (K.EQ.0) GO TO 1200
C
```

```
C INITIALIZE TO BEGIN
C
      T=1
      A(1)=1
300 G(T)=C(A(T))
      IF (T.EQ.K) GO TO 1200
      A(T+1)=A(T)+1
      IF (A(T+1).EQ.(1+N)) GO TO 900
      T=T+1
      GO TO 300
900 T=T-1
      IF (T.EQ.0) GO TO 5
      GO TO 1300
1200        CALL
POLYRECON(ZEROR,ZEROI,G,DEGREE,K,HBESTONE,
      1        HBESTTWO,MONEBEST,MTWOBEST)
      TOTAL=TOTAL+1
      IF (K.EQ.0) GO TO 5
1300        A(T)=A(T)+1
      IF (A(T).EQ.(1+N)) GO TO 900
      GO TO 300
5     CONTINUE
      PRINT *,'   *** ALL ITERATIONS COMPLETE ***'
      PRINT *,'        TOTAL COMBINATIONS = ',TOTAL
      PRINT *,'BEST DESIGN FOUND:'
      PRINT *,' '
      PRINT *,'TRANSDUCER 1:'
      PRINT *,' '
      DO 888 L=1,MONEBEST
      PRINT *,'H(',L,') =',HBESTONE(L)
      AMP(L)=HBESTONE(L)
      PHASE(L)=0.0
888 CONTINUE
      ITYPE=-1
      NUM=MONEBEST
      NFFT=MONEBEST
      FO=0.5
      TFLO=-(NUM-1)/2.0
      TFHI=(NUM-1)/2.0
      CALL WRITEO
      PRINT *,' '
      PRINT *,'TRANSDUCER 2:'
      PRINT *,' '
      DO 889 L=1,MTWOBEST
      PRINT *,'H(',L,') =',HBESTTWO(L)
      AMP(L)=HBESTTWO(L)
      PHASE(L)=0.0
889 CONTINUE
      ITYPE=-1
      NUM=MTWOBEST
      NFFT=MTWOBEST
      FO=0.5
      TFLO=-(NUM-1)/2.0
```

```
      TPHI=(NUM-1)/2.0
      CALL WRITEO
      PRINT *,' '
      RETURN
      END
C
      SUBROUTINE POLYRECON(ZEROR,ZEROI,G,DEGREE,K,HBESTONE,
     1        HBESTTWO,MONEBEST,MTWOBEST)
C
C SUBROUTINE TO FORM H1 AND H2
C
      DOUBLE PRECISION ZEROR(1024),ZEROI(1024)
      DOUBLE PRECISION AONE(1024), ATWO(1024), B(3)
      DOUBLE PRECISION CONE(1024), CTWO(1024)
      DOUBLE PRECISION HONE(1024), HTWO(1024), HMAX, HMIN
      DOUBLE PRECISION HBESTONE(1024), HBESTTWO(1024)
      DOUBLE PRECISION FOM, FOMOLD, FOMONE, FOMTWO, AA, BB
      DOUBLE PRECISION FRAT, HBOTH(1024)
      INTEGER G(1024)
      INTEGER DEGREE, K, ZCOUNT, GCOUNT, II, JJ, JJJ
      INTEGER MONE, MTWO, MONEBEST, MTWOBEST
C
C OBTAIN h1 AND h2
C
      MONE=0
      MTWO=0
      DO 11 L=1,1025
      AONE(L)=0.0D0
      ATWO(L)=0.0D0
11    CONTINUE
C
C SELECT A ZERO
C
      ZCOUNT=1
      GCOUNT=1
      DO 5000 JJ=1,DEGREE
      AA=ZEROR(JJ)
      BB=ZEROI(JJ)
C
C IF IMAG PART IS VERY SMALL, LET THIS BE A LINEAR FACTOR
C
      IF (DABS(BB).LT.1.0D-10) BB=0.0D0
C
C IF IT IS THE COMPLEX CONJUGATE ROOT (I.E. IMAG PART
NEGATIVE)
C THEN SKIP IT, SINCE IT WILL GET PICKED UP BY THE
POSITIVE
C IMAG QUADRATIC FACTOR.
C
      IF (BB.LT.0.0D0) GO TO 5000
C
C CREATE A QUADRATIC FACTOR FROM A COMPLEX SET OF ROOTS,
OR
```

```
C A LINEAR FACTOR FROM A REAL ROOT.
C
      B(1)=AA*AA+BB*BB
      IF (BB.EQ.0.0D0) B(1)=-AA
      B(2)=-2.0D0*AA
      IF (BB.EQ.0.0D0) B(2)=1.0D0
      B(3)=1.0D0
      IF (BB.EQ.0.0D0) B(3)=0.0D0
C
C FOR THE SPECIAL CASE OF THE ENTIRE TRANSFER FUNCTION ON
C ONE TRANSDUCER ONLY, LET h1 BE AN IMPULSE (=1).
C
      IF (K.EQ.0) THEN
              MONE=1
              HONE(1)=1.0D0
      END IF
C
      IF (GCOUNT.GT.K) GO TO 232
C
C INCORPORATE THE LINEAR OR QUADRATIC FACTOR INTO THE
C h1 POLYNOMIAL
C
      IF (ZCOUNT.EQ.G(GCOUNT)) THEN
              CALL
POLYMULT(MONE,AONE,B,CONE,ZCOUNT,GCOUNT,BB,HONE)
              GCOUNT=GCOUNT+1
              GO TO 5000
      END IF
C
C INCORPORATE THE LINEAR OR QUADRATIC FACTOR INTO THE
C h2 POLYNOMIAL
C
232 IF (ZCOUNT.NE.G(GCOUNT)) THEN
              CALL
POLYMULT(MTWO,ATWO,B,CTWO,ZCOUNT,GCOUNT,BB,HTWO)
      END IF
C
5000          CONTINUE
C
C SCALE THE COEFFICIENTS TO MAXIMUM OF 1
C
      HMAX=0.0D0
      DO 96 L=1,MONE
      IF (DABS(HONE(L)).GT.HMAX) HMAX=DABS(HONE(L))
96    CONTINUE
      DO 97 L=1,MONE
      HONE(L)=HONE(L)/HMAX
97    CONTINUE
      HMAX=0.0D0
      DO 98 L=1,MTWO
      IF (DABS(HTWO(L)).GT.HMAX) HMAX=DABS(HTWO(L))
98    CONTINUE
      DO 99 L=1,MTWO
```

```
      HTWO(L)=HTWO(L)/HMAX
99    CONTINUE
C
C DETERMINE THE FIGURE OF MERIT FOR THE DESIGN
C
      IF (K.EQ.0) THEN
              HONE(1)=1.0D0
              FOMOLD=0.0E0
      END IF
      DO 348 L=1,MONE
      HBOTH(L)=HONE(L)
348   CONTINUE
      DO 349 L=1+MONE,MONE+MTWO
      HBOTH(L)=HTWO(L-MONE)
349   CONTINUE
      CALL HSTATS(HBOTH,MONE+MTWO,FOM)
C
C IF THIS IS THE BEST DESIGN RELATIVE TO ALL PAST ONES,
C SAVE THE RESULTS.
C
      IF (FOM.GT.FOMOLD) THEN
              PRINT *,'** BEST YET FOLLOWS **'
              FOMOLD=FOM
              MONEBEST=MONE
              MTWOBEST=MTWO
              PRINT *,'FOM=',FOM,' K=',K
              DO 111 II=1,MONE
              HBESTONE(II)=HONE(II)
111           CONTINUE
              DO 2222 II=1,MTWO
              HBESTTWO(II)=HTWO(II)
2222                  CONTINUE
C
C OPTIONAL CODE TO PROVIDE SPECIFIC INFORMATION CONCERNING
C THE NATURE OF EACH SELECTED ROOT (I.E., PASSBAND OR
C STOPBAND, REAL OR COMPLEX)
C
c             ZCOUNT=1
c             GCOUNT=1
c             DO 1234 II=1,DEGREE
c             AA=ZEROR(II)
c             BB=ZEROI(II)
c             IF (DABS(BB).LT.1.0D-10) BB=0.0D0
c             IF (BB.LT.0.0D0) GO TO 1234
c             IF (GCOUNT.GT.K) GO TO 217
c             IF (ZCOUNT.EQ.G(GCOUNT)) THEN
c                     PRINT *,'H1 ZERO:',AA,' +/-',BB
c                     AA=DABS(1.0D0-DSQRT(AA*AA+BB*BB))
c                     IF (AA.LT.1.0D-4) PRINT *,'STOPBAND'
c                     IF (AA.GE.1.0D-4) PRINT *,'PASSBAND'
c                     IF (BB.EQ.0.0D0) PRINT *,'REAL'
c                     PRINT *,' '
c                     ZCOUNT=ZCOUNT+1
```

```
c                       GCOUNT=GCOUNT+1
c                       GO TO 1234
c               END IF
c 217                   IF (ZCOUNT.NE.G(GCOUNT)) THEN
c                       PRINT *,'H2 ZERO:',AA,' +/-',BB
c                       AA=DABS(1.0D0-DSQRT(AA*AA+BB*BB))
c                       IF (AA.LT.1.0D-4) PRINT *,'STOPBAND'
c                       IF (AA.GE.1.0D-4) PRINT *,'PASSBAND'
c                       IF (BB.EQ.0.0D0) PRINT *,'REAL'
c                       PRINT *,' '
c                       ZCOUNT=ZCOUNT+1
c               END IF
c 1234                  CONTINUE
      END IF
C
      RETURN
C
      END
C
      SUBROUTINE HSTATS(X, M, FOM)
C
C THIS SUBROUTINE DETERMINES THE STATISTICS OF THE SAMPLES
C AND RETURNS THE FIGURE OF MERIT (FOM)
C
      DOUBLE PRECISION X(1024), XMAX, XMIN, XAVG, XVAR
      DOUBLE PRECISION FOM, XSUM, XDUM, XRANGE
      INTEGER M
      XSUM=0.0D0
      XMAX=0.0D0
      XMIN=1.0D20
      XDUM=0.0D0
      DO 10 I=1,M
      XDUM=XDUM+X(I)*X(I)
      XSUM=XSUM+DABS(X(I))
      IF (DABS(X(I)).GT.XMAX) XMAX=DABS(X(I))
      IF (X(I).EQ.0.0D0) GO TO 10
      IF (DABS(X(I)).LT.XMIN) XMIN=DABS(X(I))
10    CONTINUE
      XAVG=XSUM/DBLE(M)
      XVAR=(DBLE(M)*XDUM-XSUM*XSUM)/(DBLE(M) * (DBLE(M) -
1.0D0))
      XRANGE=XMAX-XMIN
      IF (XVAR.EQ.0.0D0) XVAR=1.0D-10
      FOM=XAVG/(XRANGE*XVAR)
35    RETURN
      END
C
C POLYNOMIAL RECONSTITUTION SUBROUTINE. THIS ROUTINE TAKES
C THE LINEAR OR QUADRATIC FACTOR AND MULTIPLIES IT BY THE
CURRENT
C POLYNOMIAL.
C
      SUBROUTINE POLYMULT(M,A,B,C,ZCOUNT,GCOUNT,BB,H)
```

```
      DOUBLE PRECISION A(1024), B(3),H(1024)
      DOUBLE PRECISION C(1024), BB
      INTEGER ZCOUNT, M
C
C IF NO CURRENT POLYNOMIAL YET EXISTS, THEN THE FACTOR
BECOMES
C THE CURRENT POLYNOMIAL.
C
      IF (M.EQ.0) THEN
              DO 1002 L=1,3
              A(L)=B(L)
              C(L)=B(L)
1002                  CONTINUE
              M=1
              GO TO 1421
      END IF
C
C IF THE CURRENT POLYNOMIAL IS OF ORDER GREATER THAN
THREE, THE
C RECURSIVE RELATIONSHIP APPLIES.
C
      IF (M.GT.3) THEN
              DO 1001 L=3,M
              C(L)=B(3)*A(L-2)
              C(L)=C(L)+B(2)*A(L-1)
              C(L)=C(L)+B(1)*A(L)
1001                  CONTINUE
      END IF
C
C DUE TO VARIABLE ADDRESSING LIMITATIONS, TAKE CARE OF THE
TWO
C HIGH ORDER AND THREE LOWEST ORDER COEFFICIENTS MANUALLY
C
      C(M+2)=B(3)*A(M)
      C(M+1)=B(3)*A(M-1)+B(2)*A(M)
      C(3)=B(3)*A(1)+B(2)*A(2)+B(1)*A(3)
      C(2)=B(2)*A(1)+B(1)*A(2)
      C(1)=B(1)*A(1)
C
C UPDATE THE NUMBER OF ZEROS USED.
C
1421          ZCOUNT=ZCOUNT+1
C
C ORDER INCREASES BY TWO FOR A QUADRATIC FACTOR, ONE FOR A
C LINEAR FACTOR.
C
      IF (BB.NE.0.0D0) M=M+2
      IF (BB.EQ.0.0D0) M=M+1
C
C ESTABLISH THE NEW CURRENT POLYNOMIAL
C AND THE NEW IMPULSE RESPONSE
C
      DO 131 L=1,M
```

```
      A(L)=C(L)
      H(L)=C(L)
  131 CONTINUE
      RETURN
      END
```

REFERENCES

Balanis, Constatine A. Antenna Theory. New York: Harper and Row, 1982.

Beckenbach, Edwin F., ed. Applied Combinatorial Mathematics. New York: John Wiley and Sons, Inc., 1964.

Bishop, C.D., and Malocha, D.C. "Non-Iterative Design of SAW Bandpass Filters." IEEE Ultrasonics Symposium Proceedings (1984): 18-21.

Blomquist, Carl W. "An Analysis of Methods of Solving Roots of Real Polynomials." Master's Thesis, Florida Institute of Technology, Melbourne, 1968.

DeVries, A.J. "A Design Method for SAW Filters Using Simple Structures as Building Blocks." IEEE 1973 Ultrasonics Symposium Proceedings (November 1973): 441-444.

Jenkins, M.A. "Algorithm 493 Zeros of a Real Polynomial (C2)." ACM Transactions on Mathematical Software 1 (June 1975): 178-189.

Jenkins, M.A., and Traub, J.F. "A Three-Stage Algorithm for Real Polynomials Using Quadratic Iteration." SIAM Journal of Numerical Analysis 7, 4 (December 1970): 545-566.

Kahan, W. "Laguerre's Method of a Circle Which Contains at Least One Zero of a Polynomial." SIAM Journal of Numerical Analysis 4, 3 (1967): 474-482.

McClellan, James H., and Parks, Thomas W. "A Unified Approach to the Design of Optimum FIR Linear-Phase Digital Filters." IEEE Transactions on Circuit Theory CT-20, 6 (November 1973): 697-701.

McClellan, James H.; Parks, Thomas W.; and Rabiner, Lawrence R. "A Computer Program for Designing Optimum FIR Linear Phase Digital Filters." IEEE Transactions on Audio and Electroacoustics 21 (December 1973): 506-526.

Moore, J.B. "A Convergent Algorithm for Solving Polynomial Equations." Journal of the Association for Computing Machinery 14, 2 (April 1967): 311-315.

Morimoto, M.; Kobayashi, Y.; and Hibino, M. "An Optimal SAW Filter Design Using FIR Design Techniques." IEEE Ultrasonics Symposium Proceedings (1980): 298-302.

Rabiner, Lawrence R. "Linear Programming Design of Finite Impulse Response (FIR) Digital Filters." IEEE Transactions on Audio Electroacoustics AU-20 (October 1972): 280-288.

Rabiner, Lawrence R. "The Design of Finite Impulse Response Digital Filters Using Linear Programming Techniques." Bell System Technical Journal 51 (July/August 1972): 1117-1198.

Rabiner, Lawrence R., and Gold, Bernard. Theory and Application of Digital Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.

Rabiner, Lawrence R.; McClellan, James H.; and Parks, Thomas W. "FIR Digital Filter Design Techniques Using Weighted Chebyshev Approximation." Proceedings of the IEEE 63 (April 1975): 595-610.

Ralston, Anthony, and Wilf, Herbert S. Mathematical Methods for Digital Computers. New York: John Wiley and Sons, Inc., 1960.

Remez, E. Ya. "General Computational Methods of Chebyshev Approximation." Kiev, USSR: Atomic Energy Translation 4491, 1957.

Richie, Samuel M. "Three-Phase Unidirectional Surface Acoustic Wave Transducer Model and Computer Aided Design Implementation." Master's Thesis, University of Central Florida, Orlando, 1983.

Ruppel, C.; Ehrmann-Falkenau, E.; and Stocker, H.R. "Compensation Algorithm for SAW Second Order Effects in Multistrip Coupler Filters." IEEE Ultrasonics Symposium Proceedings (1985).

Ruppel, C.; Ehrmann-Falkenau, E.; Stocker, H.R.; and Mader, W. "A Design for SAW Filters with Multistrip Couplers." IEEE Ultrasonics Symposium Proceedings (1984): 13-17.

Schelin, Charles W. "Counting Zeros of Real Polynomials within the Unit Disk." SIAM Journal of Numerical Analysis 20, 5 (October 1983): 1023-1031.

Simons, Fred O.; Weeks, Art; and Kotick, David M. "An Optimized Program for Factoring Higher Order Polynomials on the Atari Home Computer." Southeastern IEEE Proceedings (1983): 250-251.

Smith, B.T.; Boyle, J.M.; Dongarra, J.J.; Garbow, B.S.; Ikebe, Y.; Klema, V.C.; and Moler, C.B. "Matrix Eigensystem Routines - EISPACK Guide." Lecture Notes in Computer Science 6 (1976): iii-iv.

Vaidyanathan, P.P. "Optimal Design of Linear-Phase FIR Digital Filters with Very Flat Passbands and Equiripple Stopbands." IEEE Transactions on Circuits and Systems cas-32 (September 1985): 904-917.

END
DTIC
9 — 86