N72-29194

THE USE OF LINEAR PROGRAMMING TECHNIQUES TO DESIGN OPTIMAL DIGITAL FILTERS FOR PULSE SHAPING AND CHANNEL EQUALIZATION

by

RONALD C. HOUTS
Project Director

and

DONALD W. BURLAGE
Research Associate

April, 1972

TECHNICAL REPORT NUMBER 142-102

COMMUNICATION SYSTEMS GROUP

BUREAU OF ENGINEERING RESEARCH

UNIVERSITY OF ALABAMA UNIVERSITY, ALABAMA

THE UNIVERSITY OF ALABAMA 1831

THE USE OF LINEAR PROGRAMMING TECHNIQUES TO DESIGN OPTIMAL DIGITAL FILTERS FOR PULSE SHAPING AND CHANNEL EQUALIZATION

by

RONALD C. HOUTS
Project Director

and

DONALD W. BURLAGE
Research Associate

APRIL, 1972

TECHNICAL REPORT NUMBER 142-102

Prepared for

National Aeronautics and Space Administration
Marshall Space Flight Center
Huntsville, Alabama

Under Contract Number

NAS8-20172

Communication Systems Group
Bureau of Engineering Research
University of Alabama

ABSTRACT

A time-domain technique is developed to design finite-duration impulse-response digital filters using linear programming. Two related applications of this technique in data transmission systems are considered. The first is the design of pulse-shaping digital filters to generate or detect signaling waveforms transmitted over bandlimited channels that are assumed to have ideal low-pass or band-pass characteristics. The second is the design of digital filters to be used as preset equalizers in cascade with channels that have known impulse-response characteristics.

A straightforward design procedure which can be used for both applications is established. Specifications for the design are expressed in terms of the desired impulse response output from either a pulse-shaping filter or an equalized transmission channel. Equations which are linear functions of the unknown filter multiplier coefficients are then derived corresponding to the appropriate impulse response. Finally, these equations are constrained to have the specified values at appropriate critical time points so that a linear programming routine can be applied to solve for the multiplier coefficients.

Example designs are presented which illustrate that excellent waveforms can be generated with frequency-sampling filters and the ease with which digital transversal filters can be designed for preset equalization. Other results are included to depict the relationships and tradeoffs between such digital filter parameters as sampling interval and number of multipliers required to meet such equalization specifications as amount of residual intersymbol interference which can be tolerated and input energy which must be transmitted.

ACKNOWLEDGEMENT

The authors would like to express their appreciation to the Telemetry and Data Technology Branch of the Astrionics Laboratory, Marshall Space Flight Center, for the support of this research. We would also like to thank Messrs. Roy Curl, Norman Harris, and Brooks Lawrence for their valuable programming assistance in the development of the computer routines and illustrations used in this report.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Recent developments in digital-circuit technology have indicated that the processing of signals with digital hardware, i.e., digital filtering, has some very significant advantages over the conventional analog signal processor; particularly, greater flexibility and accuracy, smaller size, and potentially lower implementation cost [1]. These advantages have resulted in a great upsurge in the research and development of digital systems to perform functions previously implemented only with analog hardware, and even more important, functions that were considered impossible due to the limitations of analog components. Emphasis has been placed on the design of software as well as digital hardware. Algorithms are being developed and programmed on digital computers to perform operations that previously required complete systems of conventional analog design.

This study will concentrate on one of these areas of research that is becoming increasingly important, namely the application of digital filters to solve problems of baseband data communications. Transmission of pulse-amplitude-modulated (PAM) signals over any realistic channel is impeded both by additive noise and time dispersion resulting from the channel's nonlinear phase characteristic. Consequently, emphasis will be given to the design and implementation of pulse-shaping digital filters for the purpose of reducing these two effects. Although filters have been designed to compensate for these two effects, most designs have required a complicated analog implementation. Hence, the intent of this study is to develop a simple-to-use algorithm for the time-domain design of digital filters so that the resulting filter can be implemented easily and will have the desired compensating effect on a data transmission channel.

A background for the study is provided in Section 1.1 by the discussion of problems inherent in baseband data transmission and the review in Section 1.2 of previous efforts to solve these problems. The content of this study is outlined in Section 1.3.

1.1 BASEBAND TRANSMISSION PROBLEMS

The transmitted signal for PAM waveforms is given by the expression

$$s(t) = \sum_{k} a_{jk}\, g(t - kt_b) \ , \tag{1.1}$$

where the coefficient a_{jk} assumes one of M possible values, denoted by the subscript $j = 1,2,\ldots,M$, for the k^{th} interval. The time interval between transmitted pulses is t_b, and the set of coefficients $\{a_{jk}\}$, which are called "input symbols", are generated by a transmitter data source at the rate of $1/t_b$ symbols per second. For example, transmission of binary data would require a coefficient set of at least two values, e.g., $a_{1k} = A$, $a_{2k} = -A$.

This investigation will emphasize the use of digital filters for the generation and detection of the individual pulses g(t) that make up the transmitted signal. For example, if there is some sort of encoder-decoder device in the communication system, so as to increase transmission efficiency, the symbols $\{a_{jk}\}$ will be defined to be the encoder output. The pulses will be assumed to have a base-band frequency spectrum, i.e., frequency components are present near, but not necessarily including, zero frequency. Any frequency translation operation such as amplitude or frequency modulation of a carrier will be assumed to be part of the transmission channel.

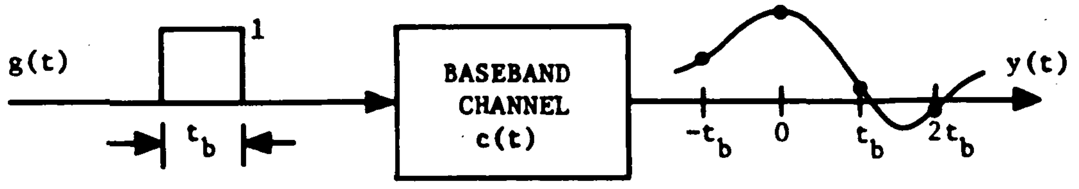
1.1.1 Intersymbol Interference

A fundamental impairment to the transmission of PAM waveforms is the frequency distortion and bandwidth limitations of realistic channels, e.g., consider the transmitted and received pulses sketched in Fig. 1-1(a). For this system, g(t) is a rectangular pulse of unit magnitude and width t_b seconds. The transmitted signal would thus consist of a sum of these pulses, each multiplied by a_{jk}. However, as indicated, a channel with impulse response c(t) produces a time spread for each pulse due to phase shifting of the spectral components. Each received pulse y(t) results from the convolution of g(t) and c(t), denoted hereafter as g(t) * c(t). Therefore, the received waveform r(t) consists of overlapping pulses, i.e.,

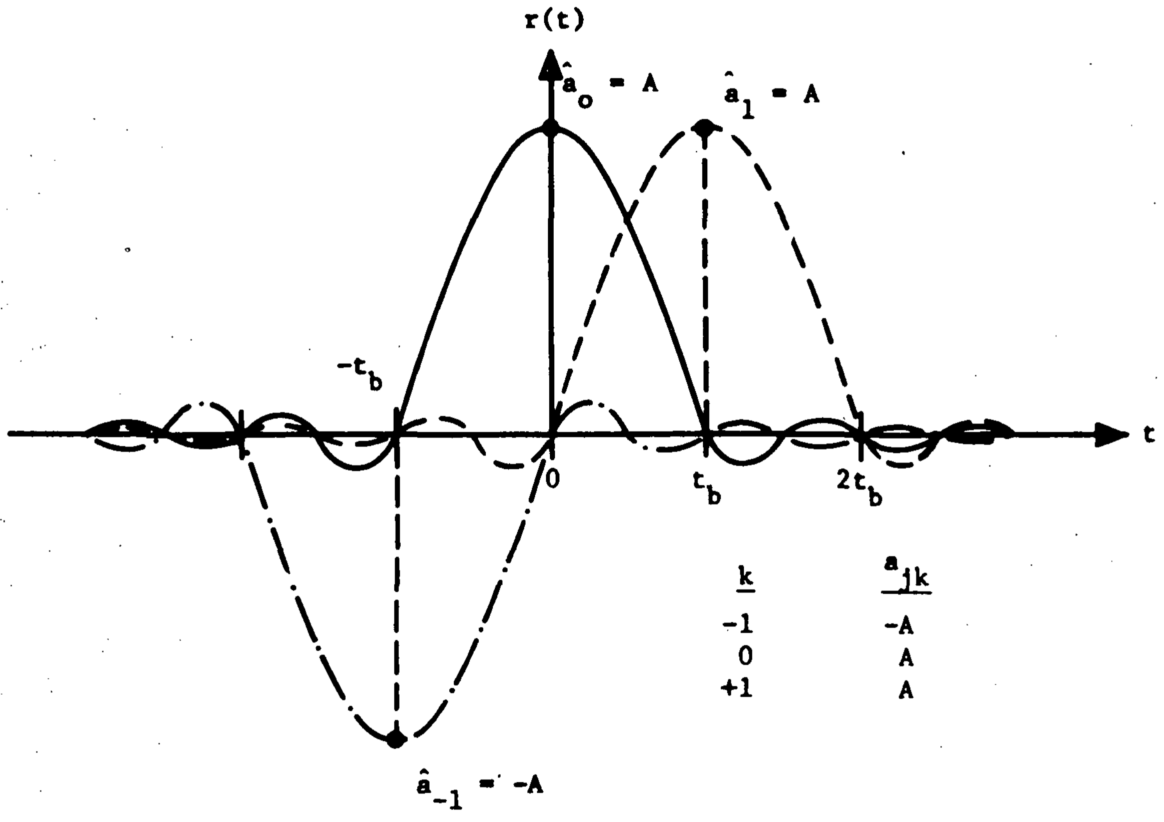
$$r(t) = \sum_{k} a_{jk} \, y(t - kt_b) \ . \qquad (1.2)$$

This time overlap results in a distortion called intersymbol interference (ISI) which can cause errors to be made in the detection of pulses, even in the absence of noise in the channel.

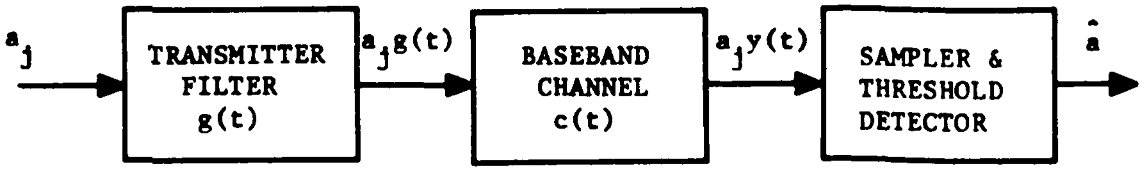
If the input symbols are to be determined by sampling the received signal at intervals of t_b seconds and comparing the resulting samples to threshold values, as is often done in simple PAM receivers,



(a) Received Pulse with Intersymbol Interference



(b) Ideal Raised-Cosine Pulses at Receiver



(c) Block Diagram of Noise-Free Baseband System

Fig. 1-1. Baseband Data Transmission.

the nonzero magnitudes at $\pm t_b$ and $2t_b$ in Fig. 1-1(a) represent ISI distortion introduced in the sample values of adjacent pulses. In this manner, the total peak ISI distortion caused by the k^{th} pulse, which appears distributed in all other sample values, is defined as

$$D_k = \sum_{\substack{\ell=-\infty \\ \ell \neq k}}^{\infty} a_{jk} |y(\ell t_b - kt_b)| \quad . \tag{1.3}$$

The same pulse waveshape is used to transmit all symbols over the same channel, so the distortion value

$$D = \frac{D_k}{a_{jk}} \tag{1.4}$$

is independent of the symbol being transmitted. Hence, reduction of intersymbol interference is equivalent to reducing the value of D.

Complete elimination of ISI distortion, i.e., D = 0, would require a waveform y(t) at the channel output with zero magnitude at the sampling times of all other pulses, i.e., at integer multiples of t_b seconds measured with reference to the pulse peak. Such a situation is illustrated by the overlapping pulses of Fig. 1-1(b). These pulses, which correspond to the sequence of transmitted symbols -A, A, and A, must be summed to give the actual signal r(t) at the channel output. However, it is obvious from Fig. 1-1(b) that the magnitude of r(t) would be identically -A, A, and A at the respective receiver sampling instants $-t_b$, 0, and t_b. Thus, the receiver decisions of the transmitted symbols, where the detector output is denoted by $\hat{a}_k$, are error free in the absence of channel noise. The received pulse shape of Fig. 1-1(b) is called a raised-cosine pulse and is of a class that

Nyquist [2] showed would have zero intersymbol interference and require a bandwidth of approximately $1/(2t_b)$ Hz. Pulses of this class are used as optimum design goals and can only be approximated in practice. Furthermore, it is difficult to approach very closely this type of pulse with analog filters; however, excellent results can be obtained with digital filters.

A technique will be developed to design pulse-shaping digital-filters that can be used to reduce the distortion D. As shown in Fig. 1-1(c), the pulse g(t) is generated with a digital transmitter filter and applied to a known channel so that the channel output y(t) will closely approximate the raised-cosine pulse. Note that the k subscript for a_{jk} and $\hat{a}_k$ as used in Fig. 1-1(b) is dropped when the transmission of a single pulse is being considered. The analog equivalent of the transmitter filter has been called an "equalizer" rather than a pulse-shaping filter, since it is usually thought of as a device which alters the composite frequency characteristic in order to reduce ISI distortion. For some applications the channel characteristic c(t) will be unknown, in which case it will be assumed that the channel is distortion free. Consequently, the pulse applied to the channel, rather than the channel output, is specified to be of the Nyquist class. A technique for the design of pulse-shaping digital filters for this application will be developed in Chapter 3 and the design technique using known channel specifications in Chapter 4.

1.1.2 Channel Noise

The received signal as defined in (1.2) was assumed to be noise free. For many channels which employ coaxial cable or shielded, twisted pairs of wires, the signal-to-noise ratio is high and the noise-free assumption is valid. Under these circumstances reduction of intersymbol interference is of prime importance. However, for low signal-to-noise ratios the emphasis must be placed on optimum detection of the transmitted pulse in the presence of noise. The channel noise is usually modeled by adding a term n(t) to the received signal, i.e.,

$$r(t) = \sum_k a_{jk}\, y(t - kt_b) + n(t) \ , \tag{1.5}$$

where n(t) is assumed to be a white Gaussian noise signal.

It has been shown [3] that a matched-filter detector will give the optimum (minimum probability of detector error) reception of signals in an additive white Gaussian noise background. The block diagram of such a detector is given in Fig. 1-2 for the detection of two equally likely equal-energy signals, $y_1(t)$ and $y_2(t)$. This detector includes two filters, each of which is "matched" to one of the possible signals, i.e., the filter unit-impulse responses are

$$h_i(t) = y_i(t_b - t) \qquad i = 1,2 \ . \tag{1.6}$$

A sampler and decision circuit form the remaining portions of the detector. During each time interval of t_b seconds r(t) will consist of the additive noise plus either $y_1(t)$ or $y_2(t)$. At the end of each interval, the filter outputs are sampled to obtain the decision statistics S_1 and S_2. The best estimate of which signal is present, i.e., which symbol has been transmitted, is then obtained by comparing the two statistics as indicated in the block diagram.

The matched-filter detector can be simplified further for signal sets where all possible signals are multiples of the same unique function. For example, if it is assumed that the two signals imbedded in the noise are $y_1(t) = A\ y(t)$ and $y_2(t) = -A\ y(t)$, where the function $y(t)$ is assumed to be identically zero outside the interval $0 \leq t \leq t_b$, then $S_1 = -S_2$ at time $t = t_b$. Hence, the decision rule can be modified so that all that is required is to determine whether S_1 is positive or negative, which eliminates the need for one of the matched filters. The PAM baseband received signal (1.5) also consists of multiples of a single function $y(t)$ plus the noise interference. Hence, if $y(t)$ satisfies the time-limited constraint, the optimum receiver for detection of the PAM pulses is a filter, sampler, and threshold detector as illustrated in Fig. 1-3. Of course, it is impossible for the function $y(t)$ to be truly time limited at the output of a bandlimited channel. Nevertheless, optimum detection of PAM baseband signals can be approximated with this type of matched-filter technique. The same technique used to design digital filters for transmission of pulses over unspecified channels will also be used to design digital filters to implement the matched-filter portion of this receiver.

1.2 HISTORICAL REVIEW

Efforts to improve the quality of PAM data transmission over the baseband channel range from Nyquist's work [4] on telegraph transmission theory in the 1920's to the present day efforts to design all digital PAM transmitters and receivers. Summaries of these studies have been published [5,6] and some of the more pertinent results, namely theoretical developments in PAM data transmission, digitally

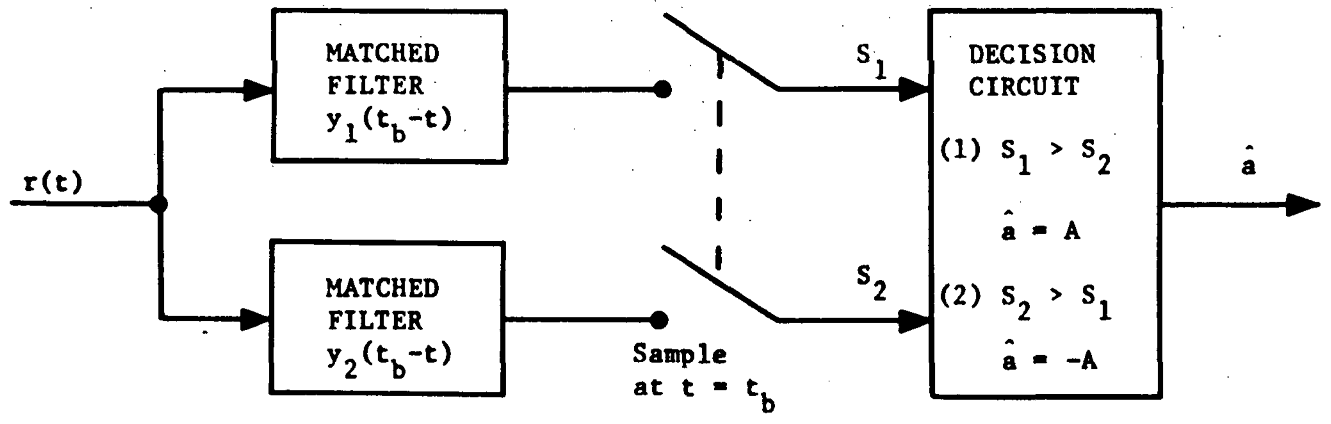


Fig. 1-2. Matched-Filter Detector.

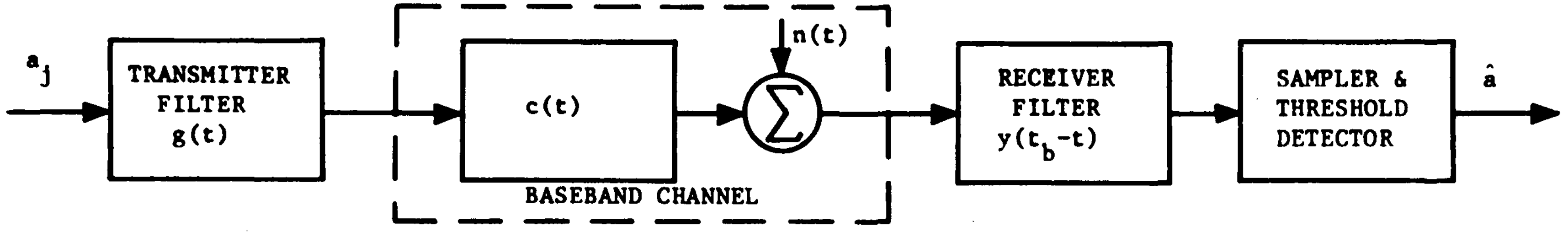


Fig. 1-3. Block Diagram of Baseband System with Noise.

synthesized PAM systems, coding-decoding techniques, and adaptive equalization will be reviewed in this section.

1.2.1 Theoretical Developments

In 1928 Nyquist published a classic paper [2] which has formed the basis for the study of PAM data transmission. He developed criteria for the required overall frequency characteristic of a finite-bandwidth transmission channel so that distortionless transmission could be achieved. He also illustrated that the minimum bandwidth required for distortionless baseband data transmission is approximately equal to the reciprocal of twice the symbol interval, i.e., $1/2t_b$. The time interval, t_b, is called the "Nyquist interval" and the corresponding bandwidth, $1/2t_b$, the "Nyquist bandwidth". However, Nyquist's technique required the solution of complicated Fourier integrals; consequently, an approximation technique called the "method of paired echoes" was developed by Wheeler [7]. He showed that small sinusoidal perturbations in either the channel amplitude or phase characteristic would result in echoes in the channel impulse-response; consequently, the location and amplitude of the echoes could be predicted by inspection of either frequency characteristic. In 1954 Sunde published a comprehensive summary [8,9] of the theoretical work of these early researchers.

More recently, emphasis has been placed on techniques to reduce intersymbol interference by placing an equalizing device at the transmitter and/or receiver, or even in the center of the transmission channel. Bogert [10] developed what he called a "time reversal technique" which in theory would eliminate the phase distortion effects of any bisectional channel. He illustrated that the channel output could be made independent of channel phase characteristics by placing an equalizer at the channel midpoint to reverse the time base of signal segments. However, the impractical technique he used for time reversal was to record signal segments at the channel midpoint on magnetic tape, and then run the recorded signal backwards past the reproduce heads during playback into the second half of the channel.

Gerst & Diamond [11] developed a technique to specify the required impulse response of a transmitter filter so that the output of a known channel would be time-limited to the Nyquist interval of t_b seconds. Consequently, ISI distortion would be completely eliminated. Unfortunately, their technique is only applicable to channels that are not bandlimited, and requires maximum energy transfer at frequencies where the channel attenuation is greatest.

Aein & Hancock [12] considered the problems of both background channel noise and ISI distortion. They assumed that the received pulse would only overlap in the following Nyquist interval, and then derived the impulse response for the matched-filter receiver which would be optimum under this assumption. Their calculated probability-of-error curves illustrated the performance improvement of their receiver over the standard matched-filter detector of Fig. 1-3. George [13] removed the restriction on the location of ISI distortion and performed a similar analysis to define an optimum matched-filter receiver. He assumed the noise to be a sum of ISI distortion and channel noise.

Gibby & Smith [14] extended the distortionless transmission criteria of Nyquist by removing the bandwidth restrictions. Their results are generalized to include channels with either no low frequency response or bandwidths that exceed the Nyquist bandwidth. Constraints were derived on the real and imaginary parts of the channel transfer characteristic to ensure distortionless transmission.

The results of a number of efforts to jointly optimize both the transmitter and receiver filters have been published. Tufts [15] studied the problem of joint transmitter and receiver filter optimization to minimize the mean-square-difference between the channel input symbols and the receiver output estimates for a finite sequence of data. The difference was assumed to result from the combined effects of intersymbol interference and channel noise. He showed that the optimum receiver was the cascade of a matched filter and a tapped delay line with time-varying coefficients. The corresponding optimum transmitted pulse waveform was required to be time limited to one Nyquist interval and to be a time-varying function. Aaron & Tufts [16] proved that this structure for minimizing the mean-square-difference also minimizes the average probability of detector error over a finite sequence of symbols. Coefficients of the tapped delay line calculated by both minimization techniques agree closely. However, calculation of these tap coefficients requires the difficult solution of simultaneous nonlinear equations, and thus severely limits the technique for practical system design. Smith [17] removed the restriction that the transmitted pulse be time limited and also obtained a transmitter-receiver filter combination which was time invariant. Furthermore, his approach did not require the solution of nonlinear equations, but he could only minimize the mean-square-difference to an approximation. Berger & Tufts [18] extended the optimization study to include the effects of timing jitter and also compared performance characteristics of the jointly optimized PAM systems to the theoretically attainable channel capacity as derived by Shannon.

1.2.2 Digital Synthesis Developments

The results of theoretical attempts to optimize PAM system performance as described in the previous subsection usually require the accurate generation of complicated signaling waveforms and the equally accurate synthesis of matched filters with specified impulse responses. Technical and economical problems associated with conventional analog techniques for waveform generation and matched filter realization have resulted in the recent attempts to design digital PAM transmitters and receivers.

Voelcker [19] was one of the first to illustrate the use of digital hardware to generate PAM signals with his binary transversal filter (BTF). His technique, which was actually a hybrid realization, was to replace the tapped delay line of the conventional transversal filter with a shift register and to use resistive summing networks to form the tap coefficients. He didn't consider the design problem of specifying filter coefficients but did analyze in detail the sources of distortion inherent in the BTF and how to reduce their effects. Since the summing portion of the filter is analog, the problems of resistor precision, loading effects, and voltage variations in the shift-register stages are still disadvantages to this type of signal generation.

Van Gerwen & Van Der Wurf [20] incorporated the BTF and a digital modulator into an experimental vestigial-sideband data transmitter.

By using digital circuitry, they were able to implement the transmitter entirely with resistors and transistors. Consequently, the complete transmitter, consisting of 203 transistors and 172 resistors, was integrated into one 2.1 x 2.7 mm^2 chip. Satisfactory transmission was obtained up to a signaling rate of 1 Mbps. They also discussed the digital implementation of other modulation techniques such as frequency-shift keying and orthogonal modulation as well as the use of digital filters in receivers.

Nowak & Schmid [21] studied the problem of designing nonrecursive digital filters* to have the raised-cosine frequency characteristic specified by Nyquist. They used a Fourier series expansion of the raised-cosine magnitude characteristic to determine the nonrecursive filter coefficients. A major portion of their study was devoted to measuring the filter's minimum stop-band attenuation and impulse-response shape as the filter clock frequency, number of terms in the Fourier series, and type of raised-cosine characteristic were varied.

Croisier & Pierret [22] have shown theoretically that a digital modulator can be used to generate vestigial-sideband and phase-modulated signals which are identical to those obtained from analog techniques. They reported four types of digital modulators which have been produced, plus one experimental model. The production units generate either two- or four-level pulses with carrier frequencies of 2.8 kHz to 64 kHz, while the experimental unit has eight-level pulses with a carrier frequency of 2.8 kHz.

---

*The nonrecursive digital filter and the corresponding Fourier series expansion technique for its design will be described in Section 2.1.

Even though emphasis is being placed on the use of digital hardware in PAM systems, it can be seen from recent literature [23-26] that researchers are still investigating analog pulse-shaping network techniques.

1.2.3 Related Developments

Developments discussed previously were concerned primarily with the analysis and design of the PAM transmitter and receiver filters, since these are the areas of emphasis in this investigation. Two other areas which have become quite important in PAM system design are the use of encoding-decoding devices and adaptive equalization.

Several authors [27-33] have discussed the application of coding techniques to increase the data transmission rate of binary data without a corresponding increase in the required bandwidth. Signaling at a rate higher than the theoretical Nyquist limit is accomplished by allowing controlled amounts of intersymbol interference. The encoding device for this method converts the original binary source sequence into a sequence of symbols $\{a_{jk}\}$ which are applied to the transmitter filter. Encoding eliminates the interdependence of detected symbols at the receiver, so that a single detector error will not be propagated and result in an error burst. The method requires that the equivalent impulse response of the cascade network consisting of transmitter filter, channel, and receiver filter be of a special shape in order to control the allowable intersymbol interference. Kretzmer [31] has tabulated a number of these impulse responses which he calls partial-response pulses. By allowing intersymbol interference in the system impulse response, more than two possible sample levels exist at the receiver. Consequently, a decoding device is used to generate binary estimates from the multilevel samples. Since more than two levels must be detected at the receiver, this technique is more sensitive to noise. Nevertheless, the possibility of an increased data rate without increased bandwidth is a definite advantage for certain low-noise environments.

Lucky, et al. [34-47] have developed the technique of adaptive equalization for use with time-varying channels such as are found in the switched-line telephone network. High-speed data transmission over channels with only preset equalizers at the transmitter and/or receiver cannot be sustained for widely time-varying channel characteristics. Consequently, an automatic equalizer, consisting of an adjustable transversal filter and associated control circuitry, has been developed to modify automatically the filter tap coefficients. Various algorithms exist for determining the proper tap settings. One technique is to transmit a test signal through the cascaded channel and automatic equalizer during a so-called "training period". Typical test signals include a sum of sinusoids equally spaced throughout the channel bandwidth, a sequence of isolated pulses, and a wide-band pseudo-noise signal. The equalizer output response is compared to a locally generated reference response and the corresponding error signal used to increment the tap settings so as to reduce error. A more recent technique [37], called the "decision-directed method", is to continually adjust the tap settings during the transmission of actual data. In this scheme the detected symbols at the receiver are assumed to correctly represent the input symbols (which should be a valid assumption if the equalizer is in near-optimum adjustment) thus eliminating the need for a locally-generated reference. The cross correlation between the regenerated input symbols and the actual received waveform is used to determine the error signal for tap-setting purposes. Problems exist in the initialization of either of these tap-setting algorithms. The test-signal technique operates on the principle of peak-distortion minimization and the algorithm will not converge if the initial peak distortion is greater than some critical value. The decision-directed technique, which minimizes mean-square distortion, takes an abnormally long time to converge if the initial tap settings do not place the equalizer in near-optimum adjustment.

1.3 OUTLINE OF STUDY

Problems associated with PAM baseband data transmission were reviewed in Chapter 1. The remaining chapters are devoted to the design of pulse-shaping digital filters to solve some of these problems.

Chapter 2 concentrates on the class of digital filters to be employed as pulse-shaping devices, i.e., the finite-duration impulse-response filter. This class, which consists of the digital transversal filter and the frequency-sampling digital filter, can be implemented in hardware or by a general-purpose computer algorithm. Block diagrams, that would be used for a hard-wired digital-logic implementation, and difference equations, that would be programmed for a general-purpose computer implementation, are included for each filter. The design techniques reported in the literature, which are mainly frequency-domain procedures, are reviewed and design examples are included.

An optimization algorithm is developed in Chapter 3 to design the frequency-sampling digital filter from impulse-response specifications. This technique is applicable to design digital filters to shape pulses for data transmission over unknown channels or for matched-filter detection of pulses in noise. Equations are derived for the filter impulse response as a linear function of unknown multiplier coefficients, which must be determined to complete the filter design. It is then shown how these equations can be employed to constrain the filter impulse response to meet certain time-domain specifications. A linear-programming optimization algorithm is used to solve the constraint equations for the unknown filter coefficients. Details regarding linear programming and its computer implementation for digital filter design are found in the appendices. The procedures for designing digital PAM transmitter filters and digital matched filters are outlined by the use of examples.

Application of the optimization algorithm to the design of digital filters for equalization of data channels that have known characteristics is developed in Chapter 4. Equations which are linear functions of the equalizer multiplier-coefficients are derived for the equivalent impulse response of the cascaded digital-filter equalizer, D/A converter, channel, and receiver filter combination. These equations are then used in the manner of Chapter 3 to constrain the received pulse shape so that the linear-programming algorithm can be used to specify the digital equalizer coefficients. Examples are included for the design of both transversal filter and frequency-sampling filter equalizers.

Finally, in Chapter 5 conclusions are drawn from the results of this investigation and problems for future study are recommended.

CHAPTER 2

FINITE-DURATION IMPULSE-RESPONSE FILTERS

A class of digital filters characterized by a finite number of nonzero values in the impulse-response will be considered in this chapter. This class of filters has particular advantages for time-domain design techniques since the impulse response need be specified for only a finite time interval. Specifically, the impulse-response, which is denoted as h(mT), is defined as a set of N values $\{h_0, h_1, \ldots, h_{N-1}\}$ with $h_m = 0$ for all other values of m. Two methods exist for the realization of this filter type, viz., the digital transversal filter and the frequency-sampling digital filter, which are nonrecursive and recursive implementations respectively.

2.1 DIGITAL TRANSVERSAL FILTERS

The digital transversal filter is a straightforward digital implementation of the analog transversal filter that was originally introduced by Kallman [48]. The use of digital logic affords more precise filtering at a lower cost and smaller size [19,49] than is possible with the tapped delay line and resistive summing network of the conventional analog transversal filter.

2.1.1 Transversal Filter Characteristics

A block diagram of the digital transversal filter is illustrated in Fig. 2-1. Each square signifies a delay element of T seconds, so

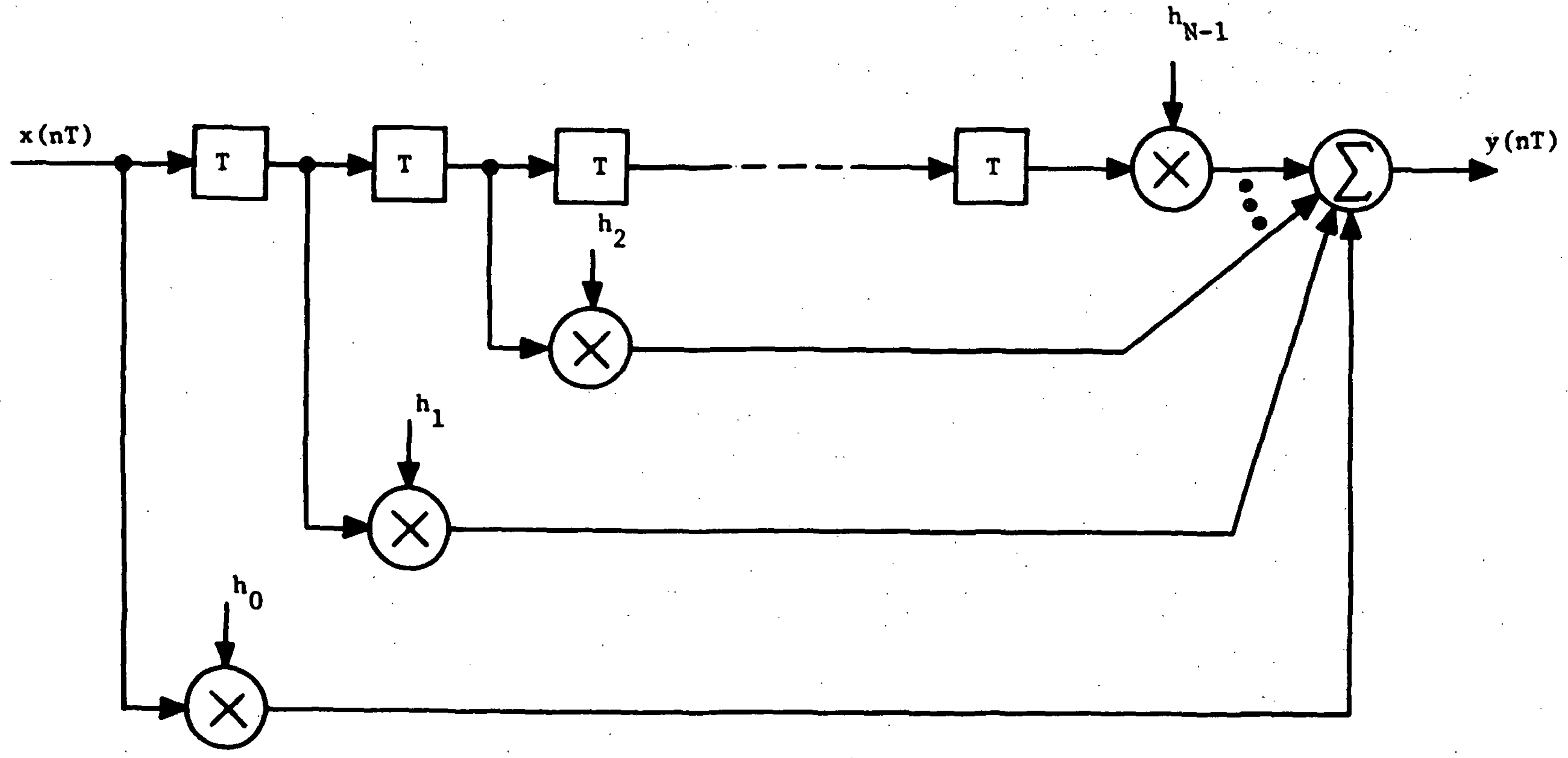


Fig. 2-1. Digital Transversal Filter.

a collective delay of NT seconds can be realized by an N-stage shift register. The multiplication and summation elements are constructed from digital logic circuits. It should be noted that each of the signal samples, x(nT) and y(nT), is actually an M-bit digital word, where 2^M is the number of quantization levels used in the digital representation. Consequently, each set of time delays is realized by a parallel bank of M shift registers and the arithmetic elements multiply or add M-bit digital words. It can be observed from inspection of Fig. 2-1 that the difference equation for the transversal filter is written as

$$y(nT) = \sum_{m=0}^{N-1} h_m x(nT - mT) \ . \tag{2.1}$$

Taking the z-transform of (2.1) gives the corresponding transfer function

$$\frac{Y(z)}{X(z)} \triangleq H(z) = \sum_{m=0}^{N-1} h_m z^{-m} \ . \tag{2.2}$$

The sequence

$$x(nT) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \ , \end{cases} \tag{2.3}$$

applied to a digital filter corresponds to the unit impulse which is applied to a continuous filter, because the response to this particular sequence is the inverse z-transform of the filter transfer function. Hence, the reason for setting the transversal-filter multiplier coefficients equal to the desired unit-impulse response $h_0,\ldots,h_{N-1}$

can be seen by applying the sequence {1,0,0,...} to the network of Fig. 2-1. It will be assumed that there are no delay elements prior to the first multiplier tap. Hence, the first of the N impulse-response values will occur at time zero, i.e., $h_m = h(mT)$. It will also be assumed that $x(nT) = 0$ for $n < 0$.

Since (2.1) is the discrete equivalent of the convolution integral as applied to continuous linear systems, the nonrecursive realization is sometimes called a "direct-convolution filter". Stockham [50] has shown that by using the fast-Fourier-transform (FFT) algorithm to calculate $y(nT)$, the required computation time can be reduced significantly for sufficiently large N. When the FFT is used in this manner, the realization is called a "fast-convolution filter".

In addition to having a finite impulse response, other distinctive characteristics of the transversal filter include the absence of stability problems in the realization, a piecewise-linear phase characteristic which generates a uniform time delay and the need for a large number of coefficient multipliers to obtain sharp amplitude characteristic cutoffs. Since the transfer function (2.2) has only zeros there is no possibility of poles falling outside the z-plane unit circle; hence, the filter is inherently stable. The amplitude and phase characteristics will be illustrated shortly.

2.1.2 Fourier-Series Design Technique

Several conventional methods for the design of nonrecursive filters have been reviewed by Kaiser [51]. These techniques, which are all frequency-domain procedures, have in the past formed the basis for transversal filter design. Perhaps the most useful has been the Fourier series expansion technique. Since the desired digital filter frequency characteristic, H(f), must always be periodic, it can be expanded in a Fourier series over the range $|f| \leq 1/2T$, i.e.,

$$H(f) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(2\pi nTf) + b_n \sin(2\pi nTf)] , \qquad (2.4a)$$

where

$$a_n = 2T \int_{-1/2T}^{1/2T} H(f) \cos(2\pi nTf) \, df \qquad (2.4b)$$

and

$$b_n = 2T \int_{-1/2T}^{1/2T} H(f) \sin(2\pi nTf) \, df . \qquad (2.4c)$$

The Fourier-series coefficients can be shown to represent the desired impulse-response values. Consequently, the desired filter coefficients are obtained by: (1) replacing the sinusoidal functions with their Euler expansion equivalents, (2) making the standard substitution

$$z = e^{j2\pi fT} , \qquad (2.5)$$

and (3) comparing the resulting transfer function with (2.2). Of course, the series must be truncated to obtain a finite number of coefficients, and in general the filter will have complex-valued coefficients and will require future values of the input sequence, i.e., the summation in (2.2) will include negative values of m.

As an example, Kaiser has considered the problem of designing an ideal digital transversal differentiator. The Fourier-series coefficients which correspond to the desired frequency characteristic

$$H(f) = j2\pi f \qquad |f| \leq \frac{1}{2T} , \tag{2.6}$$

can be found by substituting (2.6) into (2.4b) and (2.4c), giving

$$a_n = 0 \qquad n = 0,1,\ldots,N$$

and (2.7)

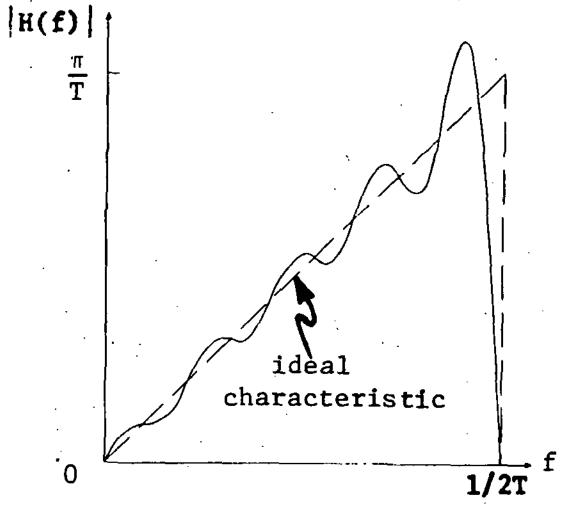
$$b_n = \frac{j2}{nT} (-1)^{n+1} \qquad n = 1,2,\ldots,N .$$

The transfer function is obtained by substituting (2.7) into the exponential form of (2.4a) with the series truncated to N = 9. Then the transfer function is multiplied by z^{-9} to eliminate the need for future input values which gives a transfer function with 18 non-zero coefficients, i.e.,

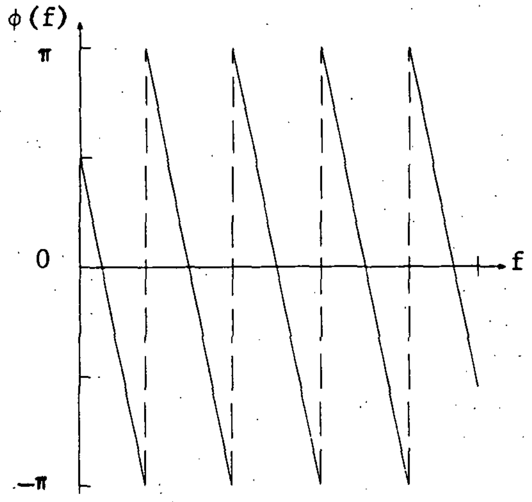
$$H(z) = \sum_{n=1}^{9} \frac{(-1)^{n+1}}{nT} [z^{n-9} - z^{-n-9}] . \tag{2.8}$$

Characteristics of the transversal filter corresponding to (2.8) with T = 1s are given in Fig. 2-2.* The Gibbs phenomenon evident in the amplitude characteristic is due to the series truncation, which illustrates the transversal-filter property that a large number of taps are required for sharp amplitude cutoffs, i.e., more terms are needed in the series expansion. As expected, the phase characteristic is exactly linear over the entire bandwidth. The linear phase shift which has been subtracted from the ideal 90° phase shift results from the z^{-9} term in (2.8). It can be seen

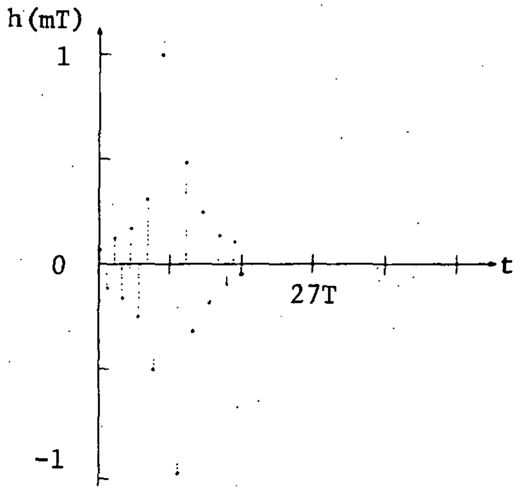
*All illustrations of digital filter frequency- and time-domain characteristics were generated with the ZFRAPP and ZTRAMP subroutines which are described in the technical report by Houts & Burlage [52].



(a) Amplitude Characteristic



(b) Phase Characteristic



(c) Impulse Response

Fig. 2-2. Frequency- and Time-Domain Characteristics of Digital Differentiator Designed by Fourier Series Technique.

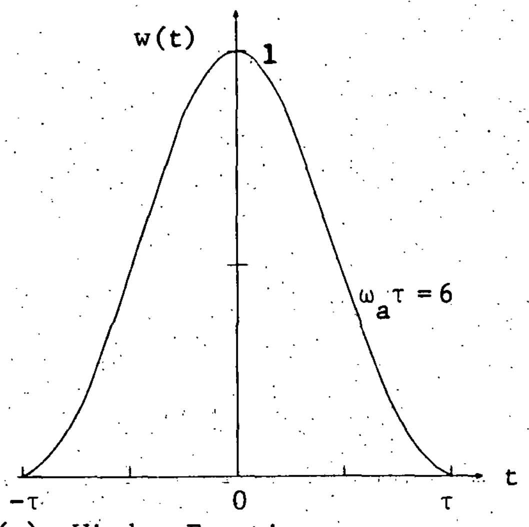
from inspection of Fig. 2-2(c) that the impulse response contains the same 18 values predicted by (2.8) plus the zero value at 9T corresponding to $a_0 = 0$.

2.1.3 Window-Function Design Technique

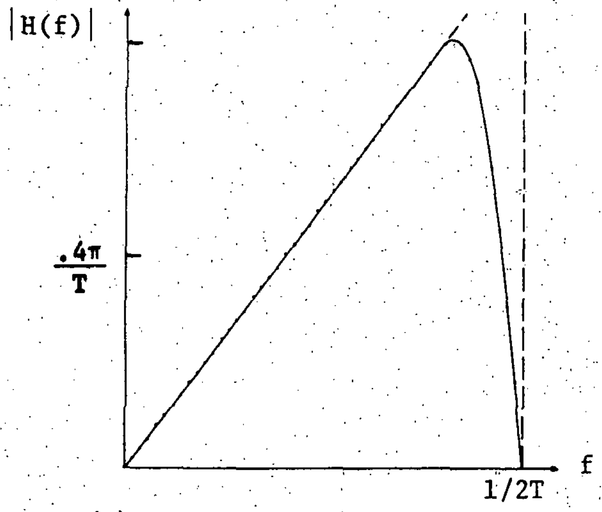
Reduction of the Gibbs phenomenom has been studied by Kaiser and others [53-56] resulting in the development of the so-called "window function" design technique for transversal filters. First, a Fourier series expansion of the desired frequency characteristic is obtained. This series is then multiplied by the window function, which is time limited and is used to modify the series coefficients so as to reduce the normal truncation error. Since multiplication in the time domain is equivalent to convolution in the frequency domain, multiplication of the time-domain coefficients by a proper window function results in a smoothing of the sharp transitions found in the desired frequency characteristic, which then can be more closely approximated. The modified Bessel window function discovered by Kaiser, i.e.,

$$w(t) = \begin{cases} \dfrac{I_0[\omega_a\sqrt{\tau^2 - t^2}]}{I_0(\omega_a\tau)} & |t| \leq \tau \\ 0 & |t| > \tau \end{cases} \qquad (2.9)$$

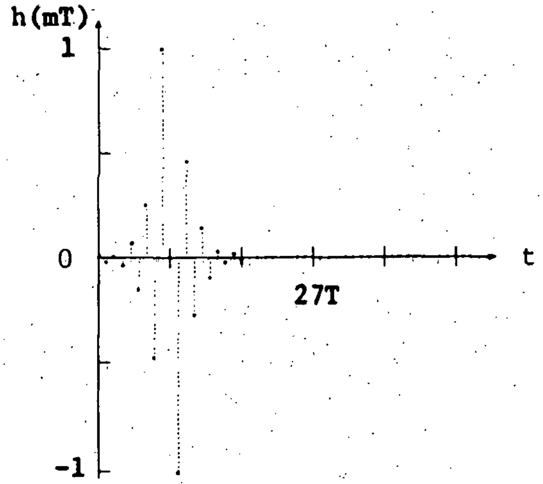
is shown in Fig. 2-3(a) and has been used to redesign the ideal differentiator. Multiplying each coefficient of (2.7) by a corresponding window-function value w(nT), with $\tau = 9T$, results in a transversal filter which has the characteristics shown in Fig.



(a) Window Function



(b) Amplitude Characteristic



(c) Impulse Response

Fig. 2-3. Frequency- and Time-Domain Characteristics of Transversal Filter Designed by Window Function Technique.

2-3. It is obvious that the ripple effect, so evident in Fig. 2-2(a), has been reduced significantly. Discussion of the advantages, disadvantages, and tradeoff possibilities of various window functions are found in the previously cited references.

Frequency-domain design techniques developed subsequent to Kaiser [51] have been primarily applications of optimization algorithms to minimize errors between the desired and actual filter frequency characteristic. Since these techniques are applicable to both the transversal and frequency-sampling filter realizations, discussion will be deferred until the frequency-sampling filter has been described.

2.1.4 Time-Domain Design Techniques

Techniques for designing digital transversal filters in the time domain have not received much emphasis in recent years nor have they been developed to the degree of the frequency-domain methods. The majority of the work that has been done is mainly applications of classical numerical analysis procedures for prediction, smoothing, and differentiation. Monroe [57], in summarizing these procedures, has derived equations to define the N coefficients, h_m, of (2.1) so that the designed filter has a maximum noise reduction capability while simultaneously having desired differentiating and/or predicting characteristics. The input to the filter is assumed to be a polynomial of finite degree with additive white noise. He has also developed a method to specify the coefficients $\{h_m\}$ so that the filter step response will meet desired criteria, such as overshoot, and still have noise rejection capabilities.

Cavin, Ray, and Rhyne [58] have considered the time-domain design of an optimal transversal filter which would give an impulse output when an expected seismic waveform, called a Ricker wavelet, is applied to its input. They studied three error criteria for indicating the difference between the actual filter response and the ideal impulse, namely, a weighted-mean-square-error, a weighted-absolute-error, and a minimax error. Filter coefficients that would minimize each of these error functions were determined with a linear programming algorithm. One problem with this technique was that all of the resulting designs were of the high-pass frequency characteristic, and could thus amplify noise components that might be present in the seismic waveform.

2.2 FREQUENCY-SAMPLING DIGITAL FILTERS

The frequency-sampling filter, a recursive realization of the finite-duration impulse response filter, was first introduced by Rader & Gold [59]. The name of the filter results from their frequency-domain design technique rather than an implied description of the filter realization or operation.

2.2.1 Frequency-Sampling Filter Characteristics

A block diagram of a frequency-sampling filter is illustrated in Fig. 2-4. The network consists of a comb filter in series with a parallel combination of 1 + [N/2]* digital resonators each

* The notation I[N/2] will be used to indicate the integer portion of N/2.

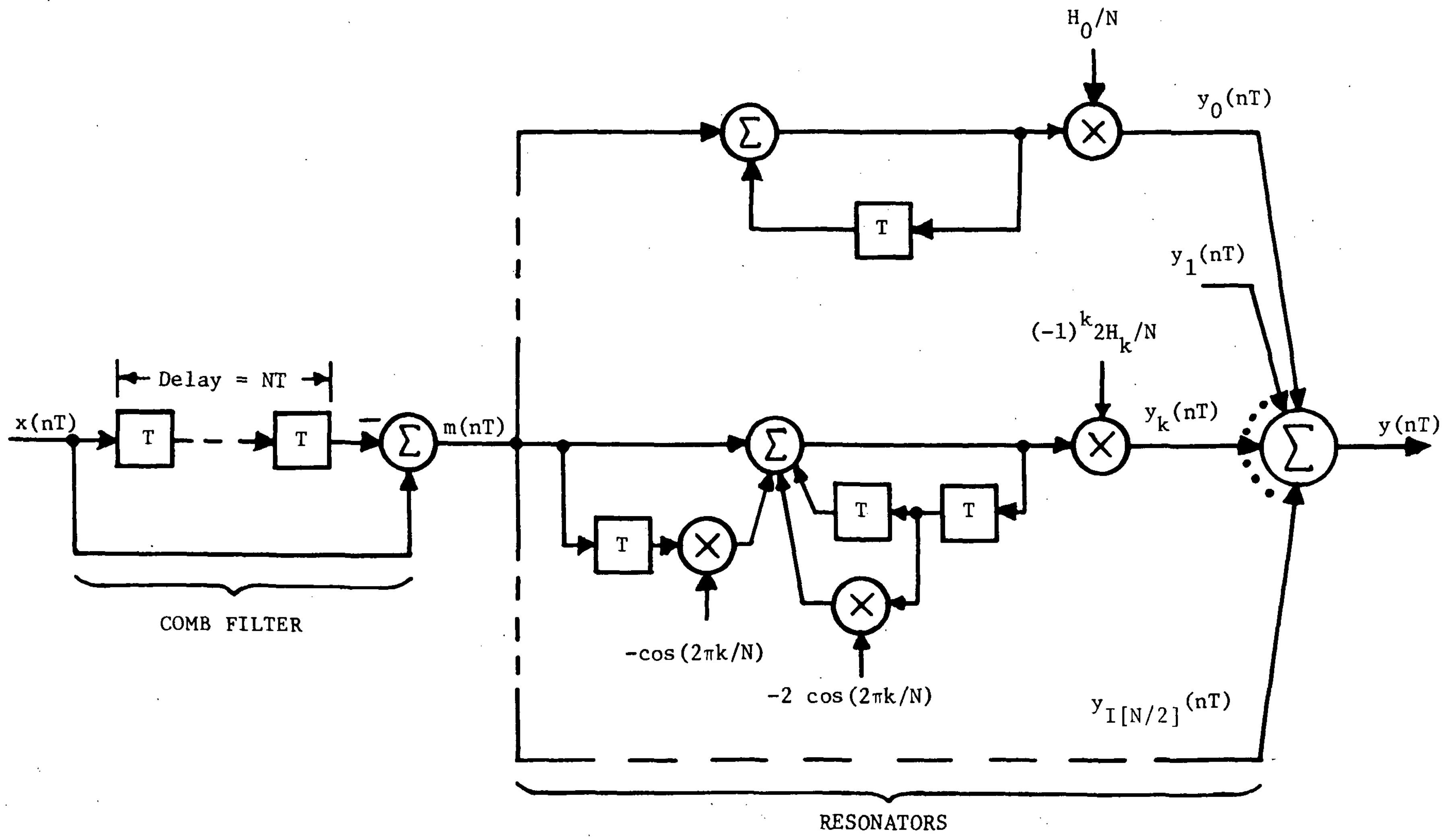


Fig. 2-4. Frequency-Sampling Digital Filter.

"tuned" to a different frequency. The reasons for this terminology will become evident when the filter frequency characteristics are considered. First, however, the filter difference equations and transfer functions must be derived.

The frequency-sampling digital filter shown in Fig. 2-4 is described by the following difference equations:

$$m(nT) = x(nT) - x(nT - NT) \ , \tag{2.10a}$$

$$y_0(nT) = \frac{H_0}{N} m(nT) + y_0(nT - T) \ , \tag{2.10b}$$

$$y_k(nT) = \frac{(-1)^k}{N} 2H_k [m(nT) - \cos(2\pi k/N)\, m(nT - T)]$$

$$- 2\cos(2\pi k/N)\, y_k(nT - T) + y_k(nT - 2T)$$

$$k = 1,2,\ldots, I[N/2] \ , \tag{2.10c}$$

and

$$y(nT) = \sum_{k=0}^{I[N/2]} y_k(nT) \ . \tag{2.10d}$$

An overall transfer function could be derived by taking the z-transform of each of these equations, as was done for the transversal filter in (2.2), and then combining the resulting expressions. Alternatively, the approach of Gold & Jordan [60] will be followed in order to better illustrate certain properties of the filter. The discrete Fourier transform (DFT) of the finite impulse response sequence $\{h_0,\ldots,h_{N-1}\}$ yields

$$H(k\Omega) = \sum_{m=0}^{N-1} h_m \, e^{-jmTk\Omega} \qquad k = 0,1,\ldots,N-1 \, , \qquad (2.11)$$

where $\Omega = 2\pi/NT$ is the increment between samples in the frequency domain. Those samples, $H(k\Omega)$, correspond to evaluation of $H(z)$ at N equally-spaced points around the unit circle in the z-plane, and consequently are values of the continuous frequency response for the filter. Taking the inverse DFT of these frequency samples yields an expression for the impulse response as a function of the samples, namely

$$h_m = \frac{1}{N} \sum_{k=0}^{N-1} H(k\Omega) \, e^{jk\Omega mT} \, . \qquad (2.12)$$

Hence, by substituting (2.12) into (2.2), interchanging summations, and using the closed-form expression for the geometric series, the following transfer function is obtained which is also a function of the frequency samples.

$$H(z) = \frac{1 - z^{-N}}{N} \sum_{k=0}^{N-1} \frac{H(k\Omega)}{1-e^{jTk\Omega} z^{-1}} \, . \qquad (2.13)$$

This function could be realized with a network very similar to that of Fig. 2-4. The term $1 - z^{-N}$ represents the transfer function for the comb filter and the terms $H(k\Omega)/N$, which may be complex valued, represent the output multiplier coefficients for the resonators. The primary difference is that each resonator would be of first-order with a single complex-coefficient multiplier. By assuming that

$$H(k\Omega) = H^*(N\Omega - k\Omega) \ , \tag{2.14}$$

complex conjugate terms in (2.13) can be combined in the manner

$$\frac{H(k\Omega)}{1 - e^{jTk\Omega}z^{-1}} + \frac{H(N\Omega - k\Omega)}{1 - e^{jT(N-k)\Omega}z^{-1}} = \tag{2.15}$$

$$\frac{2\,\mathrm{Re}\{H(k\Omega)\} - 2[\mathrm{Re}\{H(k\Omega)\}\cos(Tk\Omega) + \mathrm{Im}\{H(k\Omega)\}\sin(Tk\Omega)]\,z^{-1}}{1 - 2\cos(Tk\Omega)\,z^{-1} + z^{-2}}$$

For real H(kΩ), (2.15) becomes the transfer function corresponding to Fig. 2-4, i.e.,

$$H(z) = \frac{1 - z^{-N}}{N}\left[\frac{H_0}{1 - z^{-1}} + \sum_{k=1}^{I[N/2]} \frac{(-1)^k 2H_k[1 - \cos(2\pi k/N)z^{-1}]}{1 - 2\cos(2\pi k/N)z^{-1} + z^{-2}}\right] \tag{2.16}$$

where real H(kΩ) are denoted by H_k. The real frequency samples have been multiplied by $(-1)^k$ since Rader & Gold [59] have shown that this multiplication is necessary to have a smooth interpolation through the frequency samples by the continuous frequency response of the filter. Although the transfer function (2.16) holds for all odd N, it is only valid for even N if $H_{N/2} = 0$. If this is not the case, the function can be corrected by replacing the $H_{N/2}$ term of the summation with $(-1)^{N/2} H_{N/2}/(1 + z^{-1})$.

2.2.2 Conventional Frequency-Domain Design Techniques

All techniques that have been used to design frequency-sampling filters have been frequency-domain methods. The original technique of Rader & Gold [59] was to specify the filter multiplier coefficients $(-1)^k H_k/N$ directly from the samples H_k of the desired continuous amplitude frequency characteristic taken at N equally-spaced frequencies. An insight into this design technique can be obtained by inspection of Figs. 2-5 and 2-6. Significant poles and zeros of (2.16) are illustrated in the z-plane of Fig. 2-5 for a three-resonator filter with N = 20. The comb filter results in 20 equally spaced zeros around the unit circle. Therefore, as frequency is increased from zero to the half-sampling frequency, 1/2T, corresponding to movement midway around the unit circle, the comb-filter magnitude will appear as a series of 10 equal-magnitude lobes or "teeth" with the zero value between adjacent lobes resulting from a z-plane zero. On this semicircle, each resonator is represented by a complex pole which cancels the effect of the comb-filter zero at that resonant frequency. Consequently, those frequency components contained within a bandwidth centered on the resonant frequency are passed by that resonator stage, and the overall filter frequency characteristic is determined by the sum of these individual passbands.

A typical set of frequency samples which would correspond to the pole-zero plot of Fig. 2-5 are given in Fig. 2-6. This design of a filter with three resonator stages and a comb filter with 20 stages is an attempt to obtain an ideal low-pass frequency characteristic. Frequency- and time-domain characteristics for the filter are plotted in Fig. 2-7. It is evident from the amplitude characteristic that the continuous response passes through each of the specified frequency samples, even though the ideal characteristic could not be obtained. The piecewise linear phase shift over the entire frequency range and the finite impulse response

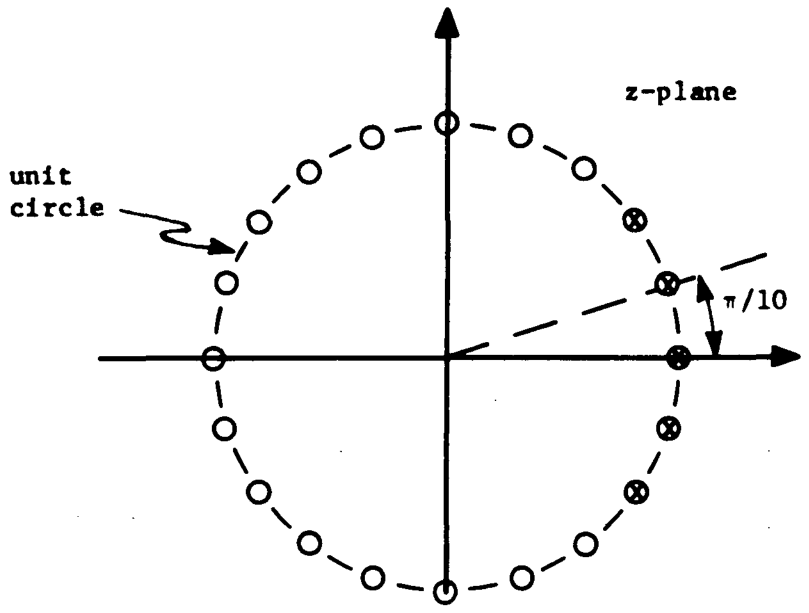


Fig. 2-5. Pole-Zero Plot for Low-Pass Filter.

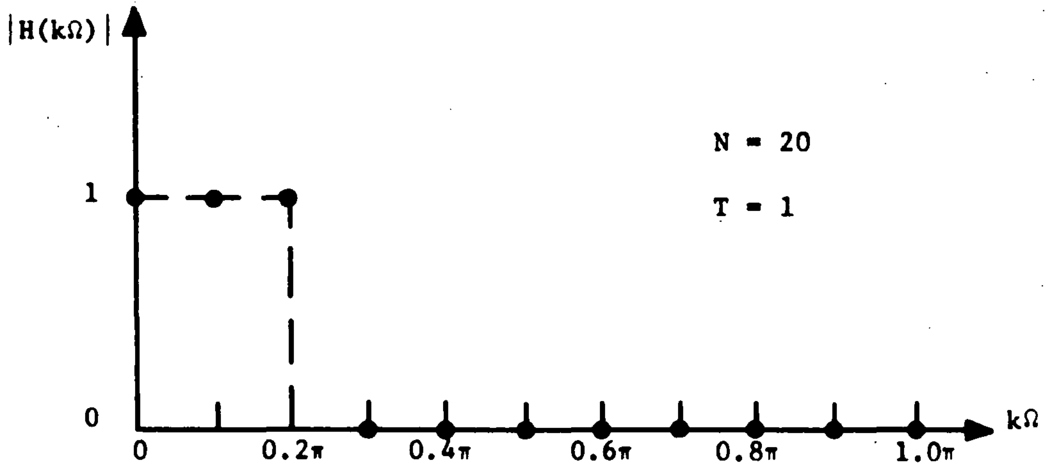
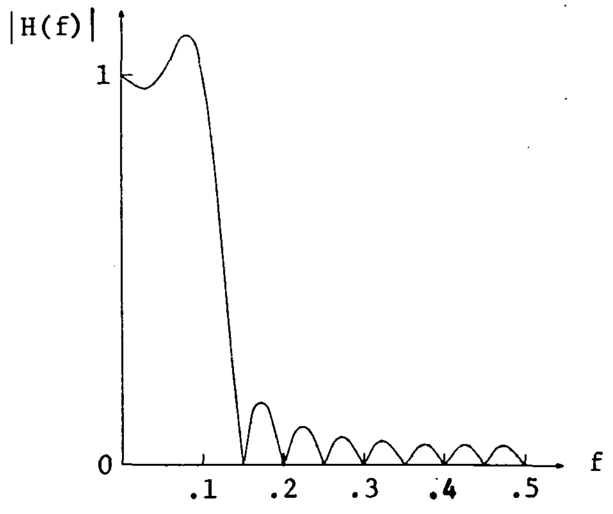
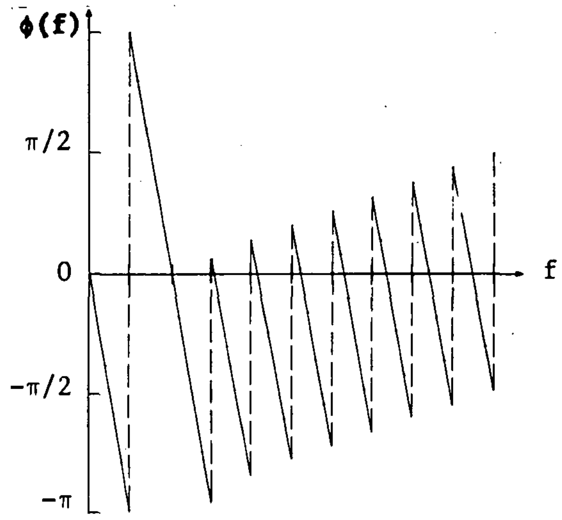


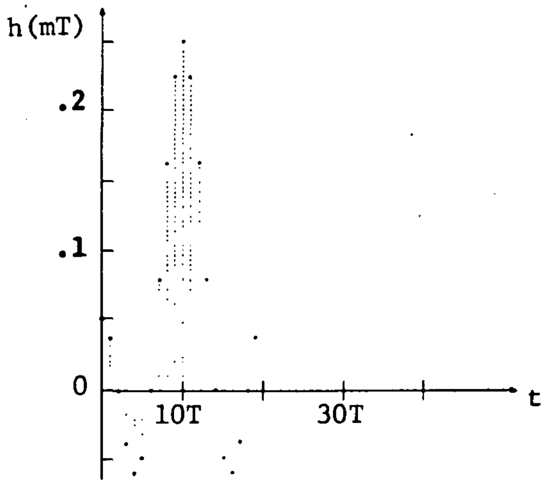
Fig. 2-6. Frequency Samples for Low-Pass Filter.



(a) Amplitude Characteristic



(b) Phase Characteristic



(c) Impulse Response

Fig. 2-7. Frequency- and Time-Domain Characteristics of Frequency-Sampling Filter.

are distinctive characteristics of this type of filter as they were for the transversal filter.

The reason for the impulse response being finite can be seen by applying the input sequence {1,0,0...} to the network of Fig. 2.4. The comb filter causes two sequences to be applied to each resonator, i.e., the original sequence and a delayed negative version NT seconds later. Then by taking the inverse z-transform of a resonator transfer function to obtain

$$H_k(mT) = (-1)^k \frac{2H_k}{N} \cos(2\pi km/N) \ , \qquad (2.17)$$

it is obvious that the resonator impulse response repeats at NT seconds. Hence, the delayed negative impulse cancels the effect of the original impulse giving a zero output for all times after NT seconds.

Any implemented digital filter requires multiplier coefficients to be represented by digital words having a finite number of bits. This coefficient quantization error can have a serious effect for the frequency-sampling filter which requires poles to be placed directly on the unit-circle stability boundary. Weinstein [61] and others who have studied this problem have shown that acceptable performances can be obtained by moving all the critical poles and zeros slightly inside the unit circle, i.e., at a radius of $r = 1 - \delta$. The value for δ can usually be in the range of 2^{-10} to 2^{-12} with little noticeable change in the filter characteristics.

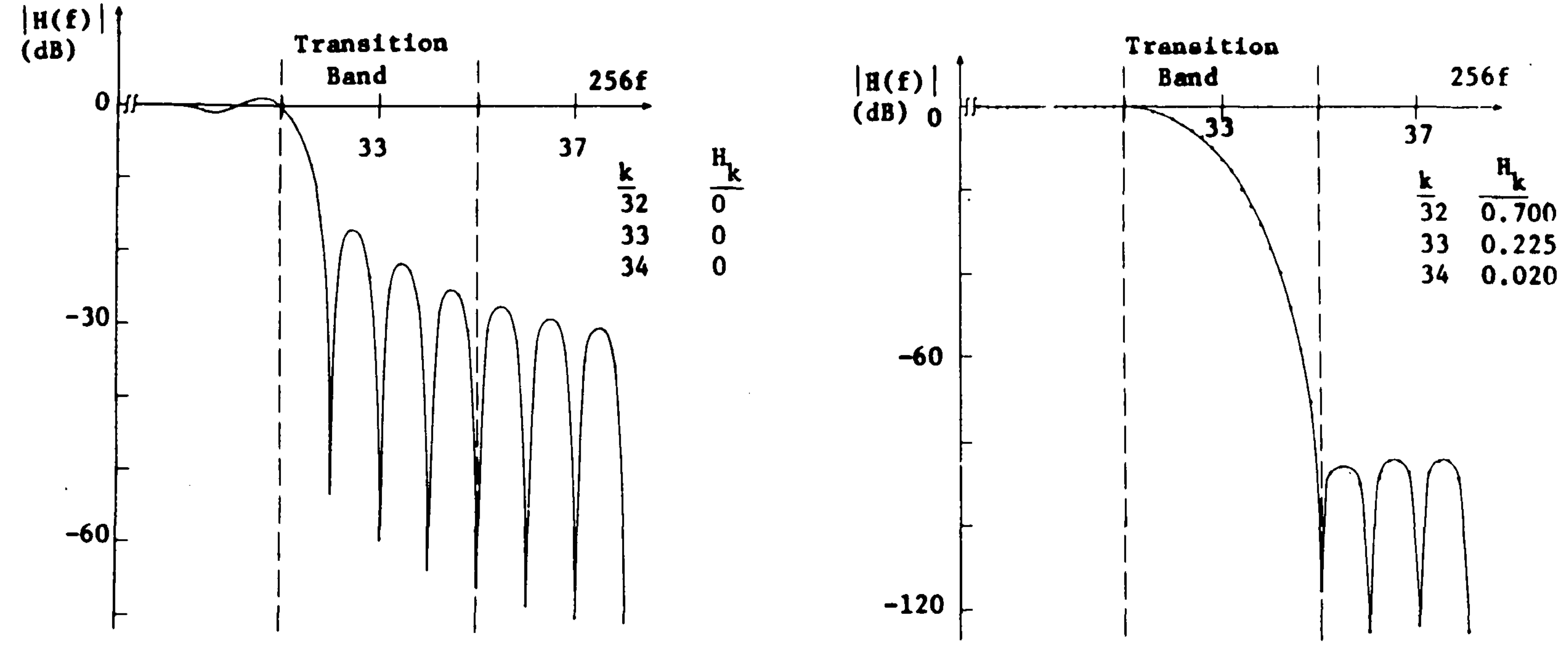
2.2.3 Optimization Technique for Frequency-Domain Design

Recently developed procedures to design frequency-sampling filters [62-64] employ optimization algorithms to minimize the maximum frequency response deviation of the designed filter from the ideal response over a specified frequency range. The basic function used in the optimization procedures can be derived by making the substitution $z = e^{j\omega T}$ in (2.13) and manipulating the resulting expression to yield

$$H(\omega) = \frac{1}{N} e^{-j[(\omega NT/2)(1 - 1/N)]} \sum_{k=0}^{N-1} \frac{H_k e^{-j\pi k/N} \sin(\omega NT/2)}{\sin(\omega T/2 - \pi k/N)} \ . \qquad (2.18)$$

This is an equation for the continuous frequency response of the filter as a linear function of its real frequency samples H_k. Hence, it is possible to apply linear optimization techniques to select a set of H_k values to minimize or maximize some characteristic of the continuous frequency response. Gold & Jordan [62] have used this technique to minimize the maximum sidelobe amplitude in the out-of-band response for a low-pass filter. A frequency characteristic resulting from their design procedure is compared in Fig. 2-8 with a characteristic resulting from the original procedure. Both filters have $N = 256$, $T = 1$ s, $H_k = 1$ for $0 \leq k \leq 31$ and $H_k = 0$ for $35 \leq k \leq 128$. The frequency samples H_{32}, H_{33}, and H_{34} in (2.18), which were set to zero for the original design, define a transition band whose values are optimally selected in the second design as 0.7, 0.225, and 0.02, respectively. By using the optimization technique, the maximum out-of-band response is reduced from -28 dB to -85 dB.

Both the original and optimum frequency-sampling design procedures can be used to design transversal filters. After obtaining the set of H_k's by either method, the inverse DFT, (2.12), is used to calculate the corresponding transversal filter multiplier



(a) Original Technique (b) Optimum Technique

Fig. 2-8. Improvement in Stop-Band Amplitude Response of Low-Pass Filter Due to Optimization Technique.

coefficients $\{h_m\}$. Unfortunately, this approach results in a transversal filter with N multipliers, 256 for the above example, and consequently a more inefficient design is obtained than with the recursive realization. This illustrates one of the main advantages of the frequency-sampling filter over the transversal filter, i.e., fewer multipliers are needed to obtain sharp cutoff-frequency characteristics.

No technique to go directly from time-domain specifications to filter coefficients has been reported for the time-domain design of the frequency-sampling filter. This type of time-domain design is the subject of the next chapter.

CHAPTER 3

DIGITAL FILTER DESIGN FROM IMPULSE-RESPONSE SPECIFICATIONS

Many of the results which have been derived for optimum PAM data transmission, as outlined in Section 1.2.1, were expressed as equations for the required impulse response of the transmitter and receiver filters. However, no straightforward technique has been developed to design and synthesize filters directly with digital hardware from these time-domain specifications. An algorithm is introduced in this chapter for that purpose. The procedure consists of three steps. First, equations are derived for the impulse response of a digital filter as a linear function of its unknown multiplier coefficients. Next, the impulse response is constrained with these equations to have appropriate values at certain critical time points. Finally, a linear programming routine is applied to solve the constraint equations and thus define the unknown filter coefficients.

3.1 TIME-DOMAIN DESIGN PROCEDURE

Expressions are derived in this section for the frequency-sampling filter impulse response, h_m, as a function of its unknown coefficients, H_k, $k = 1,2,\ldots,I[N/2]$. This derivation is not needed for the transversal filter since its multiplier coefficients are equal to the specified impulse-response values. The inverse DFT of a sampled frequency characteristic, which was obtained in Section 2.2, is an equation for the frequency-sampling filter impulse response, namely

$$h_m = \frac{1}{N} \sum_{k=0}^{N-1} H_k \, e^{jk\Omega mT} \qquad m = 0,1,\ldots,N-1 \,. \tag{3.1}$$

A z transfer function (2.16) was derived by assuming that the frequency samples were real and possessed the symmetry $H_k = H_{N-k}$. These restrictions were assumed to simplify the design problem. If the H_k were not real an optimization algorithm with complex arithmetic would be required. Furthermore, if the symmetry $H_k = H_{N-k}$ did not hold, filters with complex impulse responses would be possible. With these two assumptions (3.1) can be simplified. However, the cases for odd and even values of N will be considered separately.

3.1.1 Odd Number of Impulse-Response Values

Typical frequency samples for the N-odd case are illustrated in Fig. 3-1(a). If the H_0 sample is ignored, the samples are symmetrical about the line k = N/2. Therefore (3.1) can be written as

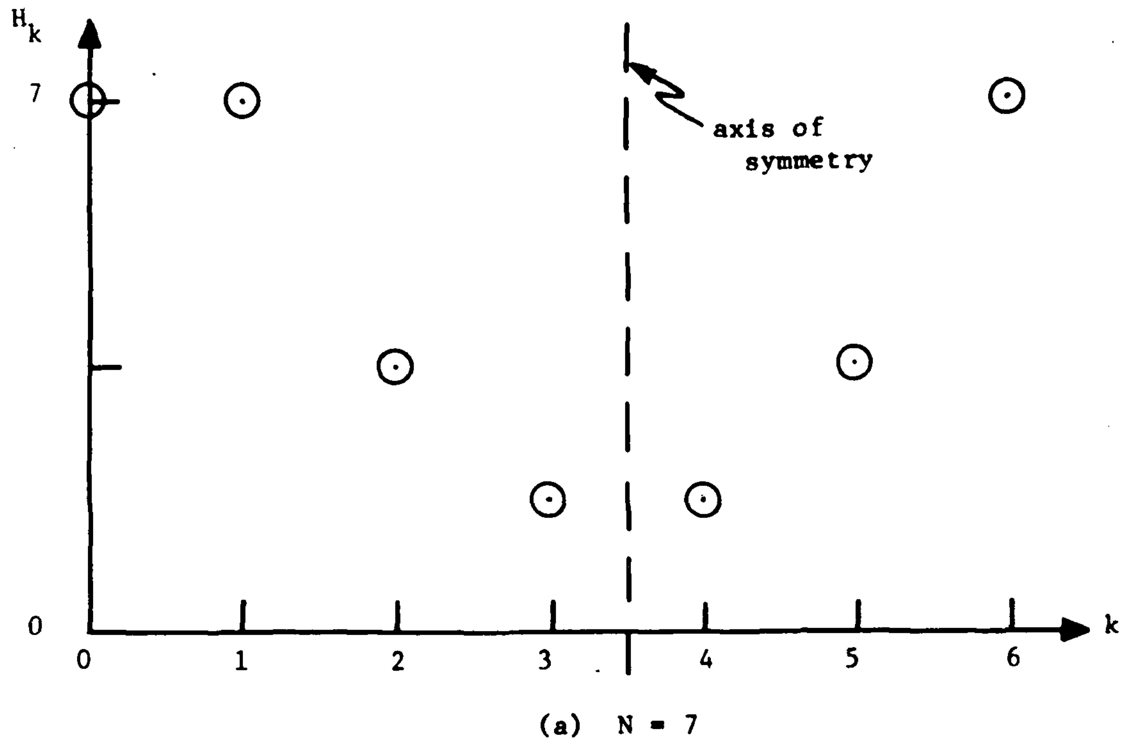
$$h_m = \frac{H_0}{N} + \sum_{k=1}^{(N-1)/2} \frac{H_k}{N} \left[e^{j2\pi km/N} + e^{j2\pi(N-k)m/N}\right] , \tag{3.2}$$

and then simplified to give

$$h_m = \frac{H_0}{N} + \sum_{k=1}^{(N-1)/2} \frac{2H_k}{N} \cos(2\pi km/N) \quad , \tag{3.3}$$

which is a real, linear function of the multiplier coefficients.

It can be seen from (3.3) that the impulse response for this case has the symmetry property $h_m = h_{N-m}$ for m = 1,2,...,(N-1)/2. If the multiplier coefficients are multiplied by $(-1)^k$, as was done



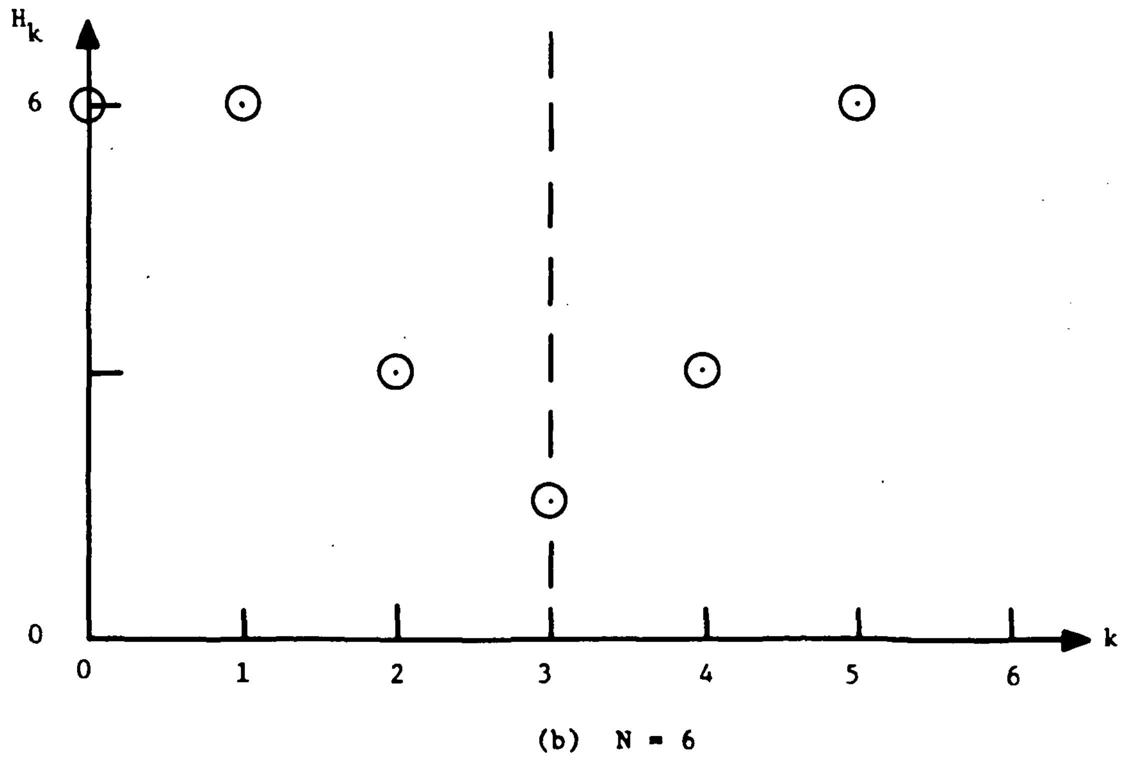


Fig. 3-1. Comparison of Frequency-Sample Symmetries for Odd and Even N.

in (2.16) to obtain a smooth interpolated frequency response, the resulting impulse response

$$h_m = \frac{H_0}{N} + \sum_{k=1}^{(N-1)/2} (-1)^k \frac{2H_k}{N} \cos(2\pi km/N) \ . \qquad (3.4)$$

still has the symmetry $h_m = h_{N-m}$. However, the impulse-response peak for (3.3) occurs at $m = 0$, whereas for (3.4) the peak would occur at $m = (N\pm1)/2$. This is evident by comparing Figs. 3-2(a) and 3-2(b), where the individual resonator impulse-response envelopes for two filters are shown. For the first filter, which corresponds to Fig. 3-1(a) and does not contain the $(-1)^k$ factor, the individual envelopes are all in phase at $t = 0$. Conversely, the envelopes for the second filter which has the $(-1)^k$ H_k coefficients are in phase at $t = NT/2$.

3.1.2 Even Number of Impulse-Response Values

Frequency samples for the N-even case, as illustrated in Fig. 3-1(b), have an axis of symmetry at N/2 which is also the location of a frequency sample. Consequently, both the k=0 and k=N/2 samples must be considered separately in writing the impulse response equation. With these observations, (3.1) becomes

$$h_m = \frac{H_0}{N} + \sum_{k=1}^{(N/2)-1} (-1)^k \frac{2H_k}{N} \cos(2\pi km/N) + (-1)^{N/2} \cos(m\pi) \frac{H_{N/2}}{N} \ . \qquad (3.5)$$

The inclusion of the $(-1)^k$ term again results in a shift of the impulse-response peak; this time from $m = 0$ to $m = N/2$. The impulse response symmetry $h_m = h_{N-m}$ also holds, as can be seen by inspecting Fig. 2-7(c) where the peak occurs at $N/2 = 10$.

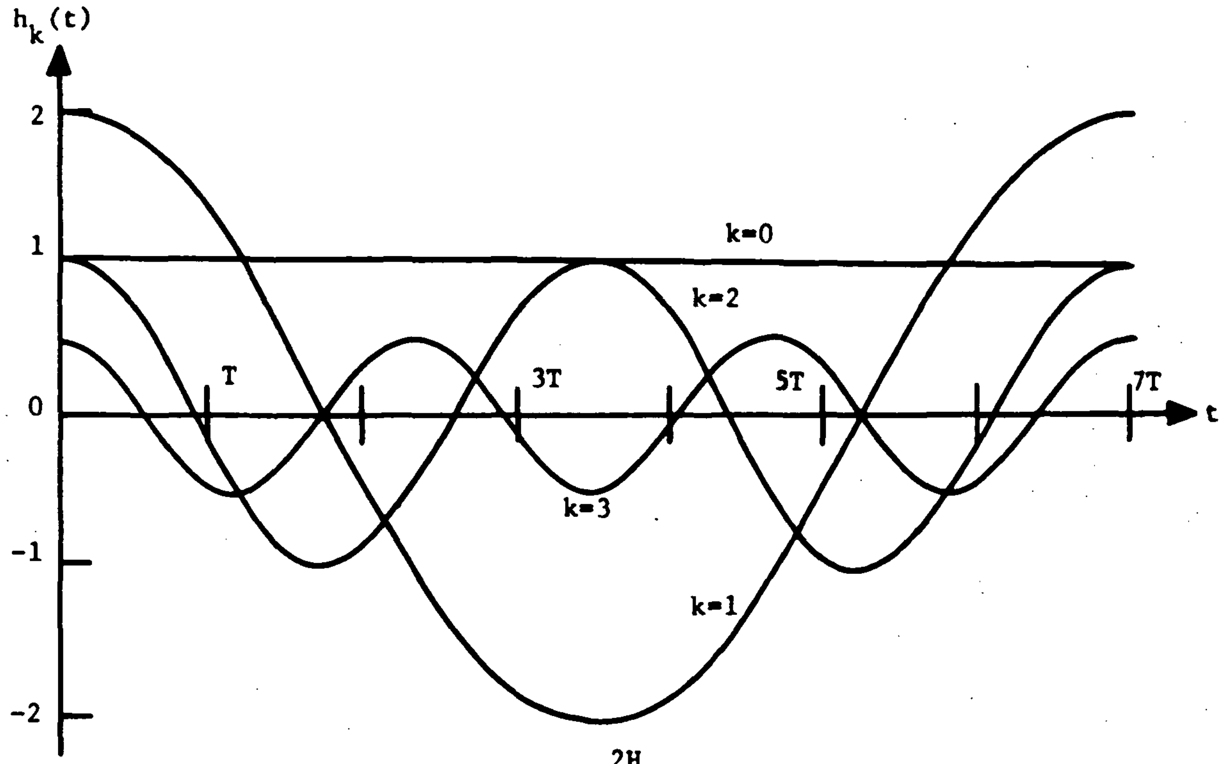
The signal component illustration of Fig. 3-2 also provides insight into the problem of designing frequency-sampling filters to have specified impulse responses. In a time-domain analysis, multiplier coefficients $\{H_k\}$ correspond to the components present in a Fourier series expansion of the impulse response. They are the weights of the sinusoidal Fourier components plotted in Fig. 3-2. Hence, the linear-programming routine can be thought of as a way to determine which Fourier components are required and what their corresponding weights should be so that the sum of the resulting sinusoids will give the desired impulse response. Of course, these component values are evaluated and added only every T seconds resulting in the discrete filter output.

3.1.3 Formulation of Constraint Equations

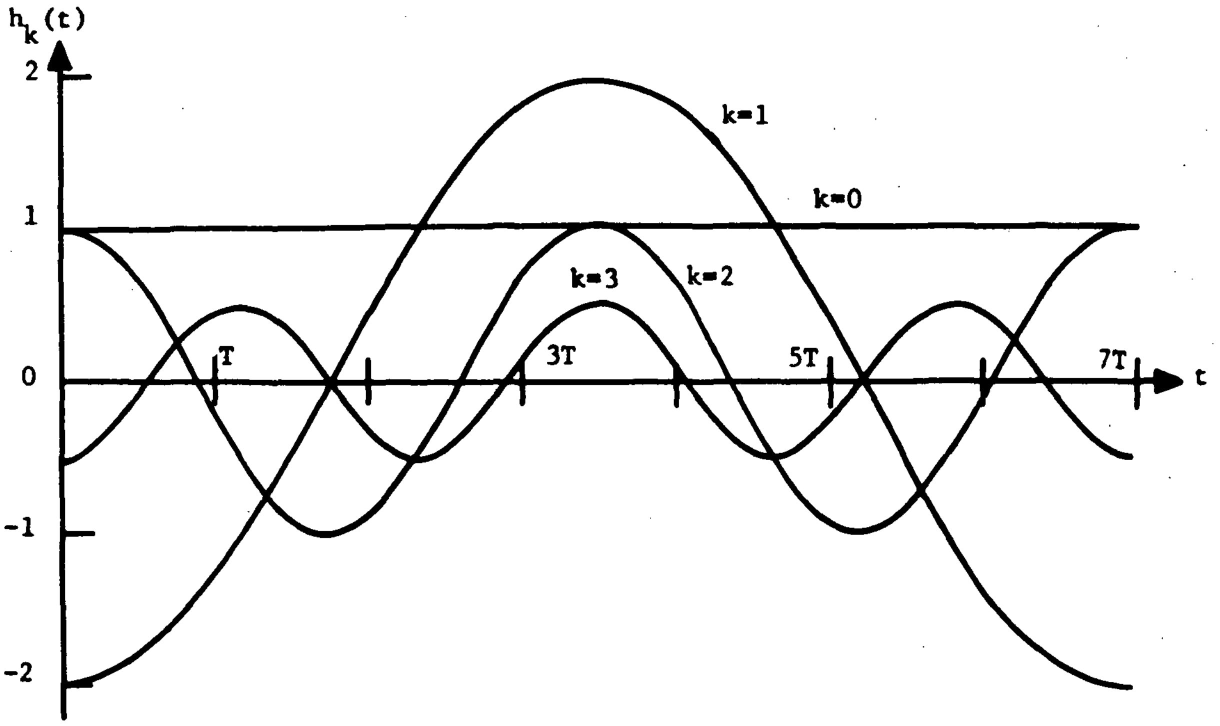
Equations (3.4) and (3.5) are discrete-valued expressions which can be used for the linear-programming time-domain design of the frequency-sampling filter. An expression for the unit-impulse response envelope, which is given by the sum of the individual resonator impulse-response envelopes, can be obtained from (3.4) or (3.5) by making the substitution $t = mT$ in (3.4) and (3.5), i.e.

$$h(t) = \frac{H_0}{N} + \sum_{k=1}^{(N-1)/2} (-1)^k \frac{2H_k}{N} \cos(2\pi kt/NT) \tag{3.6}$$

and



(a) $h_k(t) = \frac{2H_k}{N} \cos(2\pi kt/NT)$



(b) $h_k(t) = (-1)^k \frac{2H_k}{N} \cos(2\pi kt/NT)$

Fig. 3-2. Comparison of Impulse-Response Components for Two Sets of Multiplier Coefficients.

$$h(t) = \frac{H_0}{N} + \sum_{k=1}^{(N/2)-1} (-1)^k \frac{2H_k}{N} \cos(2\pi kt/NT) + (-1)^{N/2} \frac{H_{N/2}}{N} \cos(\pi t/T) \tag{3.7}$$

These expressions are continuous over the required range $0 \leq t \leq NT$ and are valid for N odd and even respectively. The envelope is a good approximation to the continuous signal that would be generated by a D/A converter and smoothing filter at the digital filter output. Since continuous linear functions of the multiplier coefficients are now available, the impulse-response slope and other higher derivatives of h(t) can be constrained. For example, the required slope equation for N odd is

$$h'(t) = \frac{4\pi}{N^2 T} \sum_{k=1}^{(N-1)/2} k(-1)^{k+1} H_k \sin(2\pi kt/NT) . \tag{3.8}$$

Equations (3.4) through (3.8) have been written in a form such that the theory of linear programming can be applied to solve for the unknown H_k, $k = 0,1,\ldots,I[N/2]$. A typical set of equations which might be used to specify a desired impulse response can be stated as follows. The slope of the impulse response at time t_1, i.e.,

$$h'(t_1) = a_{11} H_0 + a_{12} H_1 + \ldots + a_{1R} H_{R-1} \tag{3.9a}$$

is to be minimized subject to the constraint equations

$$h(t_1) = a_{21} H_0 + a_{22} H_1 + \ldots + a_{2R} H_{R-1} \geq 1 \tag{3.9b}$$

and

$$h(t_2) = a_{31} H_0 + a_{32} H_1 + \ldots + a_{3R} H_{R-1} = 0 . \tag{3.9c}$$

The coefficients a_{ij} are known, e.g., $a_{22} = (-2/N)\cos(2\pi t_1/NT)$. The H_k for k = R to I[N/2] are set to zero. Emphasis in this study will be on the design of low-pass and band-pass filters for which, frequency samples in certain regions are zero. Hence, the parameter R is defined where the frequency samples from k = R to k = I[N/2] are set to zero during the constraint equation formulation. The use of linear programming to solve for the set of R unknown multiplier values $\{H_k\}$ will be discussed in the two remaining sections of this chapter.

3.2 LINEAR-PROGRAMMING OPTIMIZATION ALGORITHM

Linear programming is an iterative procedure to solve for the set of R non-negative variables $\{x_j\}$ which maximizes or minimizes the linear function

$$Z = c_1x_1 + c_2x_2 + \dots + c_Rx_R \, , \tag{3.10}$$

subject to the M linear constraint equations

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{iR}x_R \left\{ \begin{matrix} \le \\ = \\ \ge \end{matrix} \right\} b_i \qquad i = 1,2,\dots,M \, . \tag{3.11}$$

Without loss of generality, it will be assumed that all $b_i \ge 0$ and that Z is only to be maximized. Equation (3.10) is called the "objective function". A solution to (3.11) for which all $x_i \ge 0$ is called a "feasible solution". A feasible solution with no more than M positive x_i is called a "basic feasible solution", and a basic feasible solution which maximizes (3.10) is called an "optimal basic feasible solution". A solution for which Z can be made arbitrarily large is called an "unbounded solution".

The systematic procedure to solve (3.10) and (3.11) for the optimal basic feasible solution, called the "simplex method", was first developed by Dantzig [65] and has been discussed in detail by Gass [66] and Hadley [67]. The simplex method is a procedure which consists of starting with a basic feasible solution to (3.11) and then moving from the first basic feasible solution to a second, etc. each time moving to a new solution which increases the value of Z. After a finite number of steps, an optimal basic feasible solution will be reached, i.e., no other solution yields an increase in Z. The basic mathematical technique of the simplex procedure and a computer implementation of the procedure, called LINPO, are presented in Appendices A and B, respectively. Two solutions to the following example problem are also included in the appendices to illustrate the simplex method. A hand-computation solution is developed in Section A.2, while a LINPO subroutine solution is presented in Section B.2.

Consider the problem of finding a basic feasible solution which will maximize the objective function

$$Z = x_1 + x_2 \tag{3.12}$$

subject to the constraints

$$\begin{aligned} x_1 + x_2 &\geq 2 \ , \\ 3x_1 + x_2 &\leq 15 \ , \\ x_1 + 2x_2 &\leq 9 \ . \end{aligned} \tag{3.13}$$

and

Two or three variable linear-programming problems can be represented and solved graphically. Since a feasible solution requires positive variables, all possible solutions to the problem must be in the first quadrant of the x_1-x_2 coordinate system as shown in Fig. 3-3. The three constraint equations further restrict the solution to the interior of the crosshatched region. Consequently, all points inside the region represent feasible solutions. In order to determine the optimal solution, a family of straight lines has been superimposed on the figure with each line representing points that satisfy (3.12) for one value of Z. Hence, the optimal solution is that point or set of points located in the feasible region which also lie on the objective function line with the largest value of Z. For this example, the optimal solution is denoted by the point (4.2, 2.4) where $Z_{max} = 6.6$.

3.3 DESIGN EXAMPLES

Two examples are presented to illustrate the applicability of the LINPO algorithm to the time-domain design of frequency-sampling filters. One example involves the design of a pulse-shaping digital filter to generate a raised-cosine pulse and the second involves the design of a partial-response pulse-shaping filter. Although the two design examples involve pulse-shaping filters, both can be construed to be designs of digital matched filters for the detection of the corresponding pulse in a background of additive white Gaussian noise.

The general procedure to be used for LINPO designs from impulse response specifications is outlined in the following table.

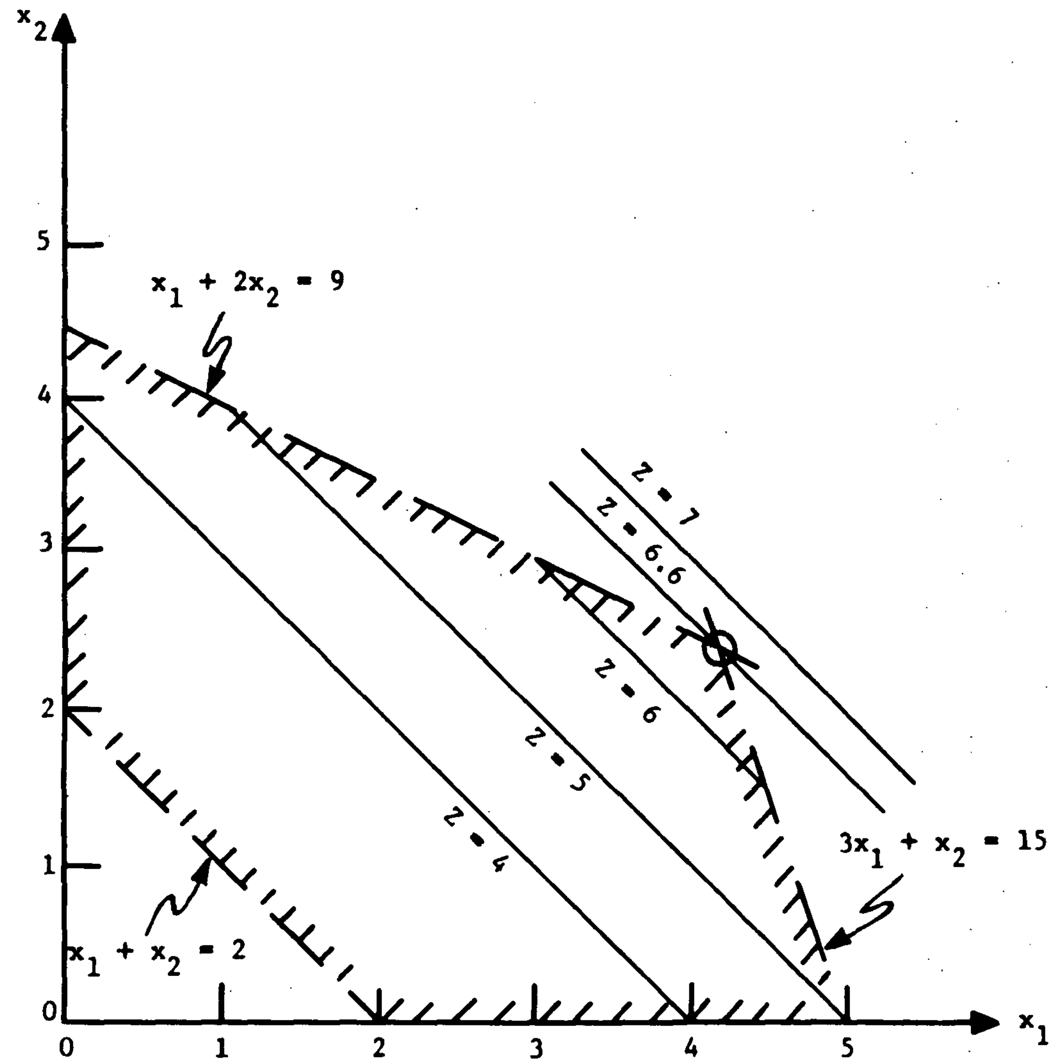


Fig. 3-3. Graphical Representation of Linear-Programming Problem.

TABLE 3.1

LINPO ALGORITHM DESIGN STEPS

1) Determine the critical time points of the desired impulse response, write constraint equations for each point, and determine an objective function. Since finite impulse-response filters are being designed, only a finite number of points of an infinite response specification can be constrained. Consequently, the most significant portion of the response must be ascertained and only points in that time interval constrained. For PAM waveforms, this interval is centered on the pulse peak and extends in both directions for a few Nyquist intervals.

2) Determine N, the number of impulse-response values, and T, the filter sampling interval. Time-domain restrictions on these values are that NT must equal the time interval corresponding to the significant portion of the impulse response and NT/2 must equal the time of the pulse peak. Physical restrictions are that the digital-logic speed limits the minimum value of T and the feasible number of shift-register stages for the comb filter limits the maximum value of N.

3) Determine R, the maximum number of resonator stages in the filter. If the bandwidth of the desired impulse response is known, R is defined since R/NT equals that bandwidth for frequency-sampling filters. Otherwise, the bandwidth is estimated from the impulse response shape and that estimate is used to calculate an initial R. Restrictions are that $R \leq I[N/2]$ and that the amount of digital logic required for the resonator stages is feasible.

4) Calculate coefficients of the unknown multiplier values in the constraint equations and objective function for input to the LINPO subroutine. Solve for the set $\{H_k\}$ with the LINPO subroutine.

5) Calculate the impulse response of the filter defined by the $\{H_k\}$ from the LINPO design. If the response is unsatisfactory, change the constraint equations and/or the parameters N, T, and R so that the response is improved when the previous step is repeated.

3.3.1 Raised-Cosine Pulse Filter

A pulse shape frequently used for baseband data transmission is the raised-cosine pulse [6] which is described by

$$g(t) = \frac{\sin(\pi t/t_b)}{\pi t/t_b} \; \frac{\cos(\alpha\pi t/t_b)}{1 - (2\alpha t/t_b)^2} , \tag{3.14}$$

where t_b is the bit time and g(t) has the non-zero frequency spectrum

$$G(f) = \begin{cases} \frac{1}{f_1} & 0 \leq |f| \leq (1-\alpha)f_1 \\ \frac{1}{2f_1}\left[1 - \sin\left(\frac{\pi f}{2\alpha f_1} - \frac{\pi}{2\alpha}\right)\right] & (1-\alpha)\, f_1 \leq |f| \leq (1+\alpha)f_1 . \end{cases} \tag{3.15}$$

The nominal cutoff frequency f_1 is equal to $1/(2t_b)$. A delayed version of (3.14), $g(t - 3t_b)$ is plotted in Fig. 3-4 for $\alpha = 1$. Only the portion of (3.14) in the interval of $\pm 3t_b$ on either side of the pulse peak has been considered significant since beyond this time interval the maximum amplitude is less than 0.002. The primary advantage of this pulse shape, as was discussed in Section 1.1, is the presence of the zero crossings at multiples of t_b. Zero amplitude at these points eliminates the intersymbol-interference transmission impairment. The design problem therefore is to specify a filter which has its impulse response given by (3.14)

The frequency-sampling filter impulse-response envelope, (3.6) or (3.7), is symmetrical about the point NT/2; so the response need only be constrained on one side of the peak. Consequently, a set of constraint equations which might be sufficient for the design are

$$\begin{array}{lll} h(3t_b) = 1 & h(5t_b) = 0 & h'(4t_b) \leq 0 \\ h(4t_b) = 0 & h(6t_b) = 0 & h'(5t_b) \leq 0 , \end{array} \tag{3.16}$$

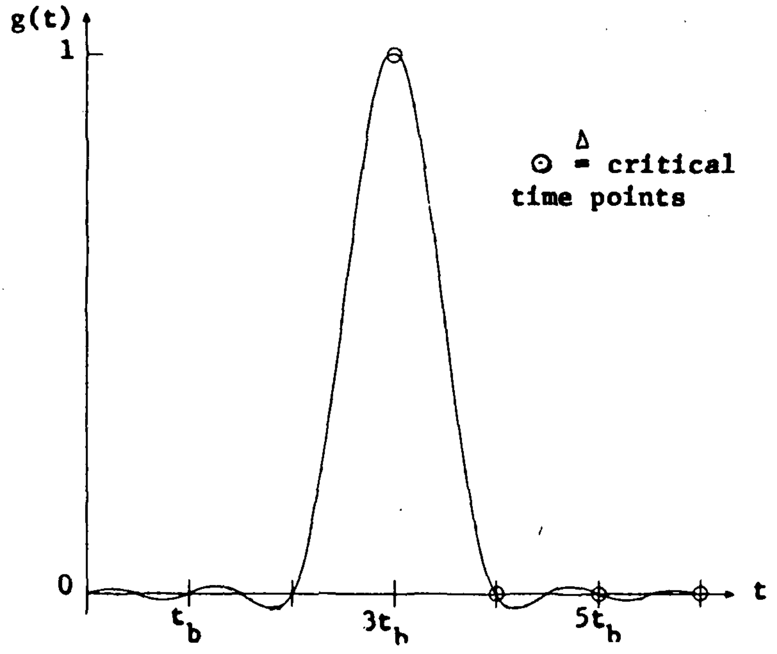


Fig. 3-4. Ideal Raised-Cosine Pulse.

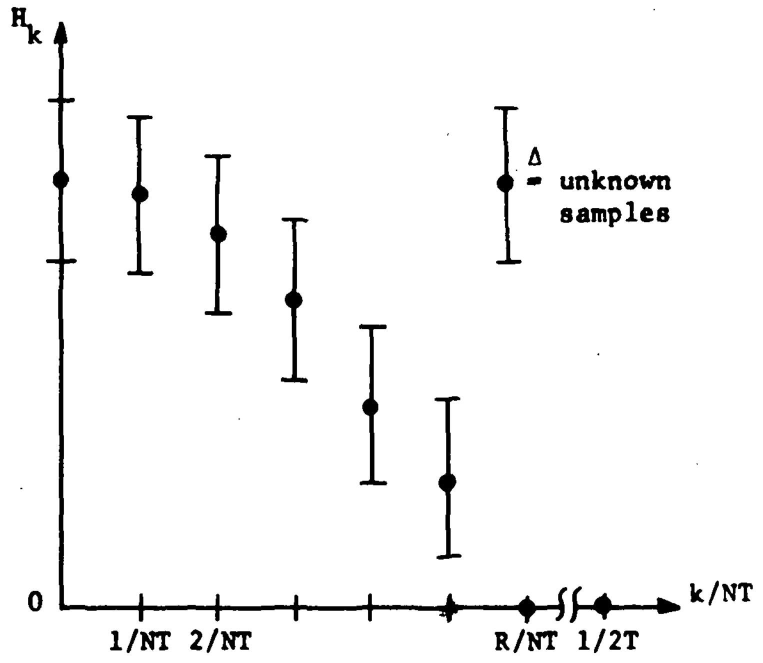


Fig. 3-5. Magnitude Samples for Raised-Cosine Filter Design (R=6).

where it is desired to maximize

$$Z = h'(4t_b) + h'(5t_b) \ . \qquad (3.17)$$

This objective function along with the two slope constraint equations will attempt to force the slope to be zero at the indicated zero-crossing points. This modification of the raised-cosine pulse would be desired to reduce the effects of timing jitter which might exist in transmission systems.

Parameters, other than the multiplier coefficients, which must also be specified are the number of impulse-response values, the sampling interval, and the number of resonator stages. The relationship of these parameters to the known and unknown multiplier coefficients in the frequency domain is illustrated in Fig. 3-5 for N even. Since the spectrum of the raised-cosine pulse has a low-pass characteristic, the frequency samples from $k = R$ to $k = N/2$ must be set to zero. Furthermore, the spacing between the samples, $1/NT$, has been fixed by the requirement that $NT/2 = 3t_b$, which is the pulse peak. It will be assumed for this example that the logic speed is sufficient for $T = t_b/10$. Hence $N = (3t_b)(2/T) = 60$, which is a realistic value for the number of shift-register stages in the comb filter. With $\alpha = 1$ in (3.15), it can be seen that the raised-cosine spectrum extends to $2f_1 = 1/t_b$. Consequently, it will be assumed that $R/NT = 2f_1$ which then yields $R = NT/t_b = 6$.

The filter design has now been reduced to a linear-programming problem with six constraint equations specified by (3.16), an objective function given by (3.17), and six unknowns, $\{H_k\}$. However, when these constraint equations are expressed in the form of (3.9), the a_{ij} coefficients, $i = 1,2,\ldots,7$ and $j = 1,2,\ldots,6$ must be calculated before linear programming can be used. An auxiliary subroutine, called COCAL1, has been developed for this purpose and is described in Appendix C. In addition to calculating the a_{ij} coefficients, this subroutine calculates and establishes all other arrays and parameters required for a linear-programming solution and then calls the LINPO subroutine which attempts to find an optimal basic feasible solution.

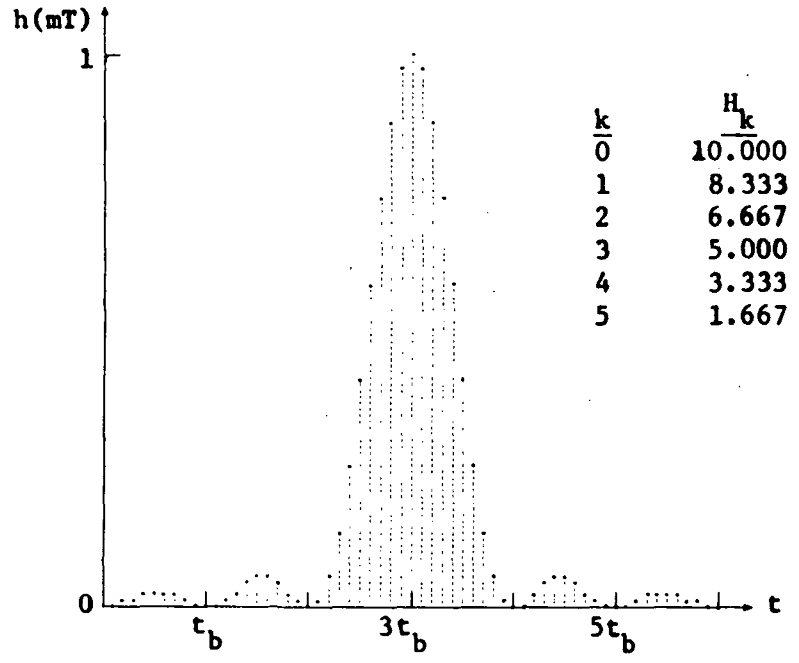
The use of COCAL1 and LINPO to solve (3.16) and (3.17) for the unknown H_k is presented as an example in Section C.1. The results of that linear-programming solution and the corresponding impulse response of the digital filter are shown in Fig. 3-6(a). Although the pulse has the desired magnitudes at $(3 \pm n)t_b$, $n = 0,1,2,3$, and the slope at the zero crossings has been forced to zero, it does not have the $\alpha = 1$ raised-cosine shape. Since zero crossings are missing at $4.5t_b$ and $5.5t_b$, two more constraints, i.e.,

$$h(4.5t_b) = 0$$

and

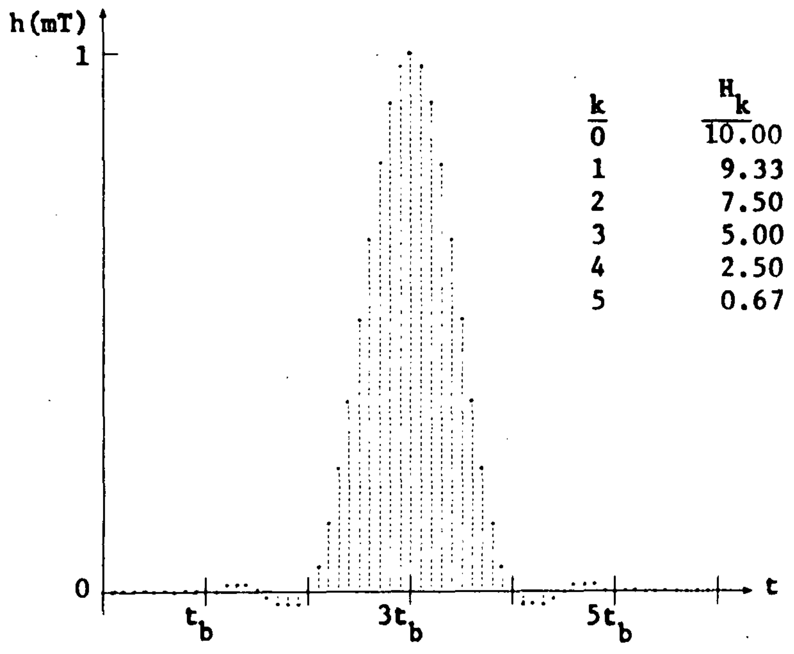
$$h(5.5t_b) = 0$$

were added to the set of (3.16) and the problem solved again. The second LINPO design, as shown in Fig. 3-6(b), yields a digital filter with an accurate raised-cosine impulse response. The difference between the impulse response and the ideal pulse is less than 0.0012 for each of the 60 response values, which is less than 0.12% of the peak pulse value. The sum of the slopes at the two zero-crossing points, $4t_b$ and $5t_b$, could not be driven to zero for the second design, as indicated by the optimum value of the objective function



k	H_k
0	10.000
1	8.333
2	6.667
3	5.000
4	3.333
5	1.667

(a) Six Constraints (Z = 0.0)



k	H_k
0	10.00
1	9.33
2	7.50
3	5.00
4	2.50
5	0.67

(b) Eight Constraints (Z = -0.036)

Fig. 3-6. Comparison of Raised-Cosine Filter Designs for Two Sets of Constraint Equations.

Z = -0.036. Consequently, for applications where jitter is a definite problem the first design would be desired. Conversely, if jitter is negligible the second design will generate an excellent raised-cosine pulse and could be used. It may seem obvious that the zero crossings at $4.5t_b$ and $5.5t_b$ should have been constrained in the first place. However, it is desirable to keep the number of constraint equations to a minimum which in effect reduces the hardware required.

A situation which did not arise during this example is the case of no feasible solution existing for a specified set of consistent constraint equations. The number of equations was increased from six to eight during this design, but the number of unknowns was held to six. If the number of equations is continually increased in this manner without a corresponding increase in the number of unknowns, a point will be reached where the problem is overconstrained in that it does not have a feasible solution. Consequently, the number of unknowns would have to be increased to obtain a feasible solution. However, this increase requires more hardware in the filter since each unknown corresponds to a resonator stage. Hence, it may become necessary to make tradeoffs between hardware complexity and the number of constraint equations employed in the design. A set of constraint equations with no feasible solution is illustrated in the next example.

3.3.2 Partial-Response Pulse Filter

The raised cosine pulse shapes have zero magnitude at multiples of the time parameter t_b; hence, zero intersymbol interference.

Another class of pulse shapes which permit intersymbol interference, but in a controlled amount, are the partial-response pulses [31] defined by

$$g(t) = \sum_{n=-2}^{2} k_n \frac{\sin[\pi(t - nt_b)/t_b]}{\pi(t - nt_b)/t_b} . \tag{3.18}$$

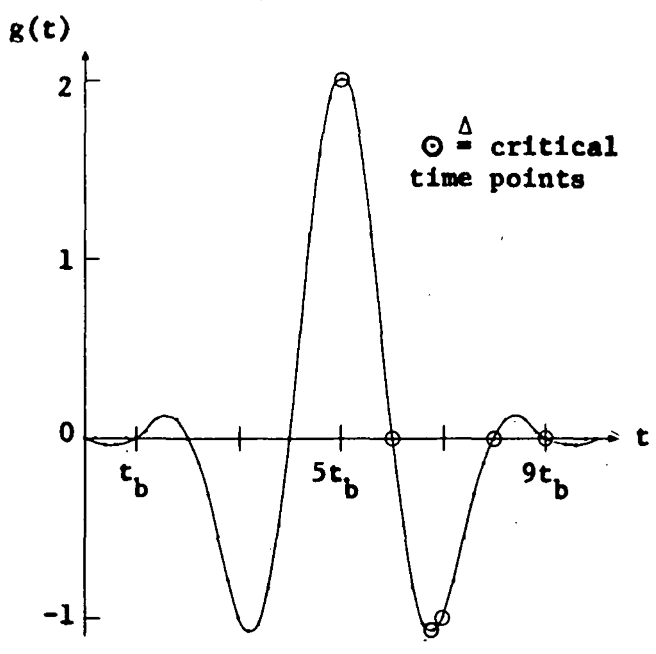
The significant portion of a typical pulse is plotted in Fig. 3-7(a) for the set of integer weights $k_{-2} = k_2 = -1$, $k_{-1} = k_1 = 0$, and $k_0 = 2$. These weights result in a pulse which has intersymbol interference at times $\pm 2t_b$ from the peak but not at other multiples of t_b. The corresponding pulse spectrum is

$$G(f) = \frac{4}{f_1} \sin^2(\pi f/f_1) \qquad 0 \leq |f| \leq f_1 . \tag{3.19}$$

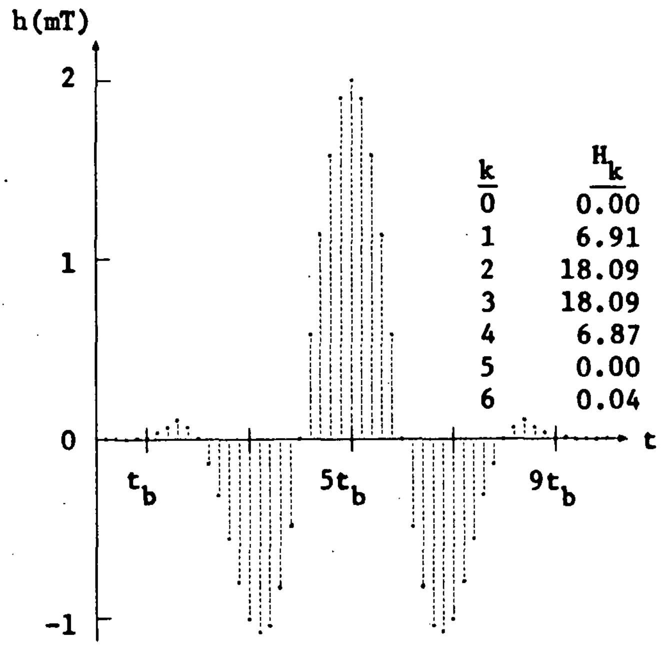
It is evident from a comparison of (3.15) and (3.19) that by allowing this specified amount of interference, the total channel bandwidth required for the pulse is halved and the need for the channel to possess dc-frequency response is eliminated.

A frequency-sampling filter which generates the partial-response pulse is designed in the same manner as the raised-cosine pulse filter. With the peak of the impulse response specified by $NT/2 = 5t_b$ and assuming that $T = t_b/5$, the resulting number of impulse response values for the pulse is $N = (5t_b)(2/T) = 50$. The number of resonators stages, R, required to generate the pulse is obtained by setting $R/NT = f_1$, which is specified in (3.19) to be the required bandwidth. It then follows that $R = NT/2t_b = 5$.

Critical values of the partial-response pulse are the peak value of 2.0 at $5t_b$, the intersymbol interference magnitude of -1.0 at $7t_b$,



(a) Ideal Partial-Response Pulse



k	H_k
0	0.00
1	6.91
2	18.09
3	18.09
4	6.87
5	0.00
6	0.04

(b) Impulse Response of Designed Filter (Z = 0.0)

Fig. 3-7. Comparison of Partial-Response Filter Design to Ideal Pulse.

and the zero magnitudes at $6t_b$, $8t_b$ and $9t_b$. Although not a critical value, another distinctive characteristic of the pulse is the zero slope at $6.756t_b$. A corresponding set of constraint equations are

$$\begin{array}{lll} h(5t_b) = 2 & h(7t_b) = -1 & h(9t_b) = 0 \\ h(6t_b) = 0 & h(8t_b) = 0 & h'(6.756t_b) \leq 0 \end{array} \tag{3.20}$$

By maximizing the objective function

$$Z = h'(6.756t_b) \ , \tag{3.21}$$

subject to the last constraint equation, it should be possible to force the slope to zero at the desired time point. However, the linear-programming problem defined by (3.20) and (3.21) has six constraint equations but only five unknowns. As a consequence, application of the COCAL1 and LINPO subroutines to solve the problem resulted in the statement "infeasible", i.e., no feasible solution exists for five variables. Consequently, it was necessary to either increase the number of unknowns or decrease the number of equations. All of the constraints were desirable, so the number of unknowns had to be increased to seven before the satisfactory solution shown in Fig. 3-7(b) was obtained.

The difference between the designed filter impulse response and the partial-response pulse is less than 0.84% of the peak for each of the 50 response values. The price paid to obtain a feasible solution, however, was an increase in pulse bandwidth from 5/NT to 7/NT and the possible addition of two resonators. Fortunately, the additional multiplier coefficients corresponding to this increased bandwidth were insignificant i.e., $H_5 = 0.000$ and $H_6 = 0.043$. In fact, if H_6 is ignored and the resulting impulse response is compared with the ideal pulse, it has been observed that the maximum error remains at 0.84%

3.3.3 Additional Comments

Although the linear-programming technique for the design of digital filters has been stated as a time-domain procedure, the two examples made use of pulse bandwidth information for specification of R, the number of unknowns. The bandwidth, and consequently R, can always be obtained from a rigorous Fourier analysis of the specified waveform or to a lesser degree of accuracy from an inspection of the waveform shape. Alternatively, digital filters can be designed entirely in the time domain. For example, a matched filter for the Ricker seismic wavelet [68] has been designed without spectral information. The desired impulse response and the three attempts that were required to determine the final filter coefficients are illustrated in Figs. 3-8 through 3-10. Constraint equations and values used for the parameters N and T were defined from the Ricker wavelet plot to be

$$\begin{aligned} h(.022) &= -0.443 & \quad h(.032) &= 0.198 \\ h(.028) &= 0 & \quad h'(.032) &\le 0 \end{aligned} \tag{3.22}$$

where it was desired to maximize

$$Z = h'(.032) \tag{3.23}$$

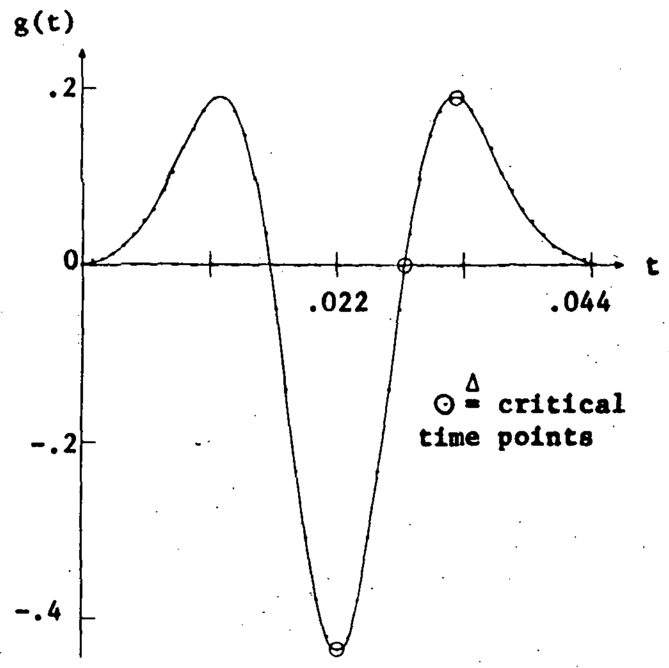
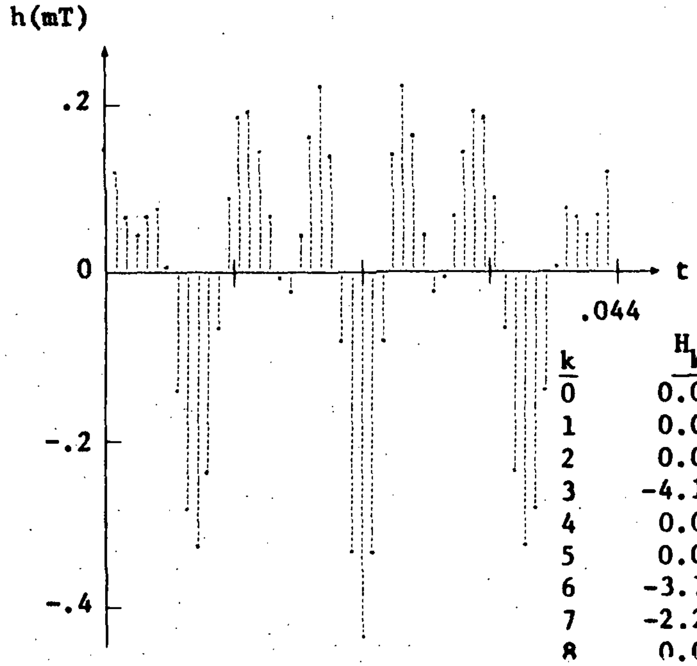
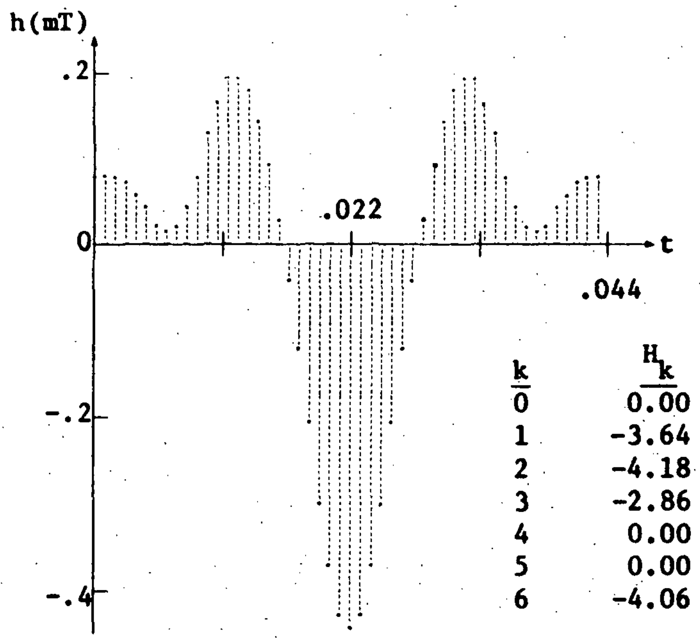


Fig. 3-8. Ricker Wavelet.



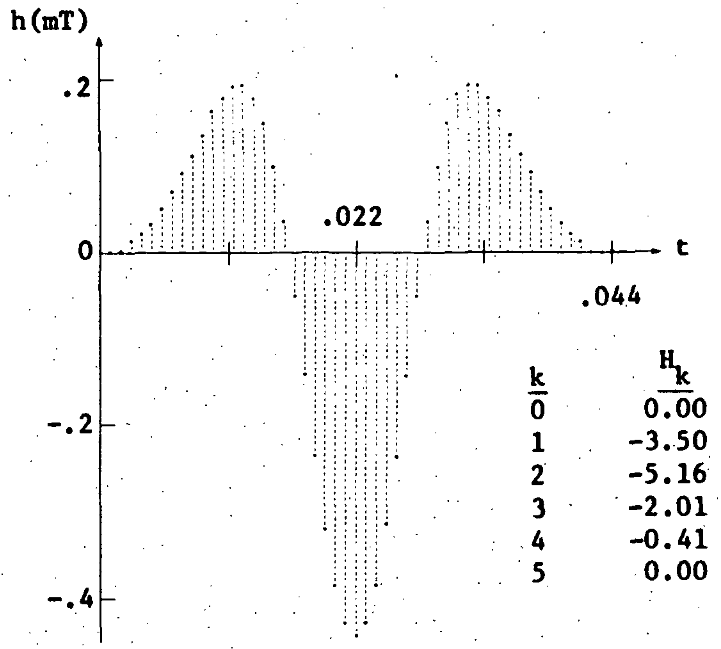
k	H_k
0	0.00
1	0.00
2	0.00
3	-4.11
4	0.00
5	0.00
6	-3.70
7	-2.20
8	0.00
9	-1.08

Fig. 3-9. Initial Ricker-Wavelet Matched Filter Design (R=10).



k	H_k
0	0.00
1	-3.64
2	-4.18
3	-2.86
4	0.00
5	0.00
6	-4.06

(a) Second Design (R=7).



k	H_k
0	0.00
1	-3.50
2	-5.16
3	-2.01
4	-0.41
5	0.00

(b) Third Design (R=6).

Fig. 3-10. Comparison of Other Ricker-Wavelet Matched Filter Designs.

with NT=0.44s and T = 0.00088s. The number of unknowns was not specified from the wavelet bandwidth but was determined by iteration. If R was chosen too small, the linear-programming problem would not have a feasible solution, but if it was too large the resulting impulse response would have an unnecessarily high frequency content. After an initial estimate of ten for R, the impulse response of Fig. 3-9 was obtained. Although the response satisfies the constraint equations, it is very evident from the high frequency envelope that R is too large. Consequently, R was reduced to seven in Fig. 3-10(a), and finally to six, before the satisfactory impulse response of Fig. 3-10(b) was obtained. The corresponding changes in the set of multiplier coefficients $\{H_k\}$ can be observed from the results given with each figure. Objective function values were Z = 0 for each case. The application of linear programming to frequency-sampling digital filter design for baseband-channel equalization, which will be considered in the next chapter, employs only time-domain specifications. Therefore, an iterative procedure such as just discussed is used to determine a satisfactory solution.

All examples and equations derived in this chapter have been for the design of frequency-sampling filters. No application of linear programming to pulse-shaping transversal filters has been considered, since the unknown multiplier coefficients for this filter are specified simply by samples of the desired pulse shape taken at T-second intervals. Also, it is a very inefficient way to generate pulses since a large number of multipliers are required, e.g., the raised cosine pulse of Fig. 3-4 would require 60 multipliers for a transversal filter as opposed to 16 for a frequency-sampling filter. However, the design of transversal filters for channel equalization is neither a trivial problem nor an inefficient solution, as will be illustrated in the next chapter.

CHAPTER 4

DIGITAL FILTER DESIGN FROM KNOWN CHANNEL SPECIFICATIONS

The application of linear programming to the time-domain design of frequency-sampling digital filters has been discussed in Chapter 3. An extension of that technique will be developed in this chapter for the purpose of equalizing nonideal baseband data transmission channels to reduce intersymbol interference. Time-domain equations are derived for the received pulse at the channel output so that the pulse can be constrained to have a desired shape before sampling and threshold detection. Linear programming is used to design either a frequency-sampling or transversal digital filter which in cascade with the channel will result in the desired pulse at the sampler when an impulse is applied to the equalizer.

4.1 DIGITAL FILTER EQUALIZERS

The technique of digital-filter equalization is illustrated in Fig. 4-1. The digital equalizer which replaces the normal transmitter filter will be designed to have a "predistorted impulse response". This predistortion is used to compensate for distortion that occurs during passage through the channel, thus resulting in an ideal pulse $y(t)$ at the output. Each symbol a_j is considered to be the weight of an impulse that is applied to the digital filter to generate a predistorted pulse. The required D/A converter and any

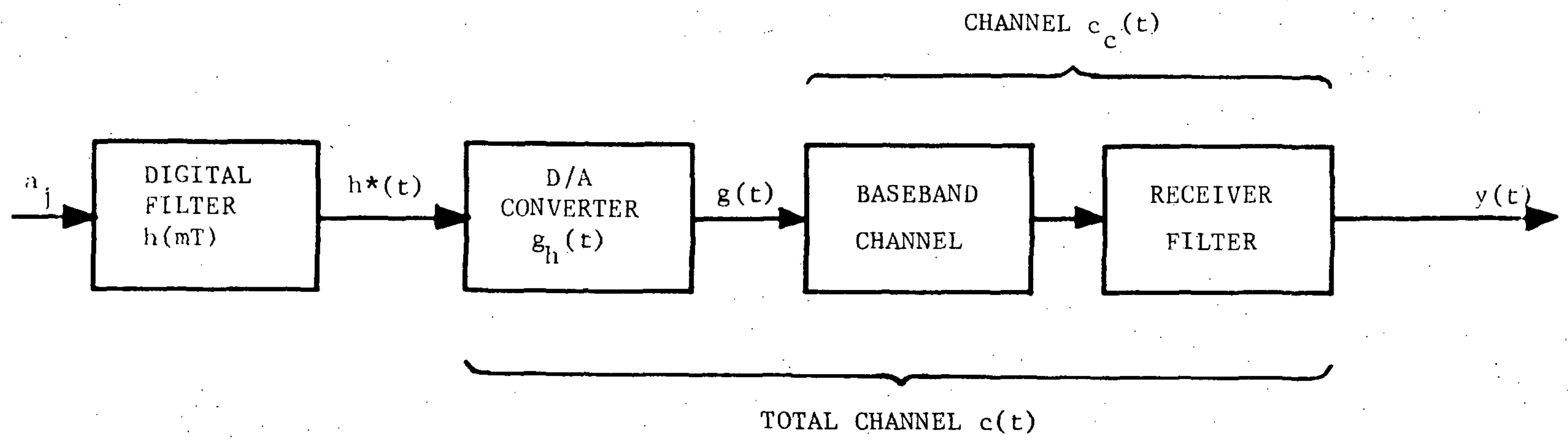


Fig. 4-1. Channel Equalization with Digital Filter

receiver filter which would be used to reduce channel noise are both considered as part of the channel and must be included in c(t), the total channel impulse response. Equations for y(t) and y'(t) as functions of both filter multiplier coefficients and values of c(t) will be derived in this section for two finite impulse-response filters. The relationships between c(t) and $c_c(t)$ will also be defined.

4.1.1 Frequency-Sampling Filter

Equations (3.4) and (3.5) were derived in Chapter 3 for the impulse response of the frequency-sampling filter. By assuming that $H_{N/2} = 0$ if N is even and redefining the upper summation limit to be R-1, the single equation

$$h_m = h(mT) = \frac{H_0}{N} + \sum_{k=1}^{R-1} (-1)^k \frac{2H_k}{N} \cos(2\pi km/N) \tag{4.1}$$

is obtained for the predistorted pulse that is to be applied to the channel input. Since the sampling process is modeled with an impulse modulator in z-transform theory, the digital filter unit-impulse response can be rewritten as

$$h^*(t) \triangleq \sum_{m=0}^{N-1} h(mT)\, \delta(t - mT) \, . \tag{4.2}$$

Consequently, the channel output for $a_j = 1$ is given by

$$\begin{aligned} y(t) &= h^*(t) * c(t) \\ &= \sum_{m=0}^{N-1} h(mT) \int_0^t \delta(\tau - mT)\, c(t - \tau) d\tau \, . \end{aligned} \tag{4.3}$$

The limits of integration on the convolution integral can be verified with the graphical-convolution sketch of Fig. 4-2(a). Applying the impulse-function sifting property to (4.3) yields

$$y(t) = \sum_{m=0}^{L} h(mT)\, c(t - mT) \tag{4.4}$$

where

$$L = \begin{cases} I[t/T] & t < NT \\ N-1 & t \geq NT . \end{cases} \tag{4.5}$$

The channel output is obtained by substituting (4.1) into (4.4) i.e.,

$$y(t) = \frac{H_0}{N} \sum_{m=0}^{L} c(t - mT) + \sum_{m=0}^{L} \sum_{k=1}^{R-1} (-1)^k \frac{2H_k}{N} \cos(2\pi km/N)\, c(t - mT) . \tag{4.6}$$

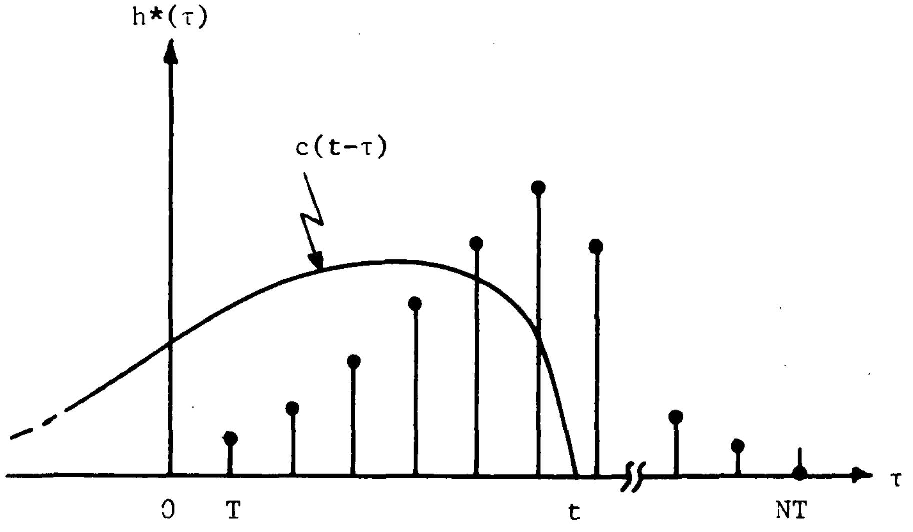
Hence, the output pulse slope is

$$y'(t) = \frac{H_0}{N} \sum_{m=0}^{L} c'(t - mT) + \sum_{m=0}^{L} \sum_{k=1}^{R-1} (-1)^k \frac{2H_k}{N} \cos(2\pi km/N)\, c'(t - mT) . \tag{4.7}$$

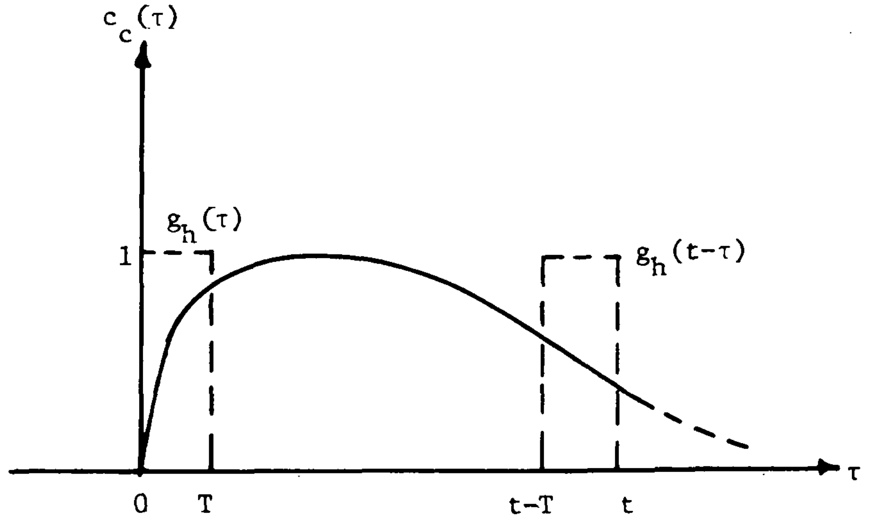
These last two equations are used to calculate constraint equation coefficients during linear-programming design of frequency-sampling equalizers.

4.1.2 Transversal Filter

It was shown previously by substituting (2.3) into (2.1) that the transversal-filter impulse response values are equal to the filter



(a) Functions for (4.3)



(b) Functions for (4.12)

Fig. 4-2. Establishing Integration Limits for Convolution Integrals.

multiplier coefficients. Consequently, the filter unit-impulse response can be written as

$$h^*(t) = \sum_{m=0}^{N-1} H_m \, \delta(t - mT) \ , \tag{4.8}$$

where the multiplier coefficients for the transversal filter are now denoted by H_m. Hence, the channel output is

$$y(t) = h^*(t) * c(t) = \sum_{m=0}^{L} (-1)^m H_m \, c(t - mT) \ , \tag{4.9}$$

where the parameter L was defined previously in (4.5). The factor $(-1)^m$ has been included because the set $\{H_m\}$ will be the solution of a linear-programming problem, and thus must be nonnegative. Without this factor, only nonnegative impulse response transversal filters would be designed, and for lowpass channels the output can not be constrained to have the desired zero crossings. Restricting every other multiplier coefficient to be negative simplifies the linear-programming problem of finding a feasible solution. The derivative of the output pulse is given by

$$y'(t) = \sum_{m=0}^{L} (-1)^m H_m \, c'(t - mT) \ . \tag{4.10}$$

Equations (4.9) and (4.10) are the desired expressions to be used during linear-programming designs of transversal equalizers.

4.1.3 Channel Impulse Response

Since the derivative of the total channel impulse response appears in both (4.7) and (4.10), continuity requirements on c(t) must be investigated. The D/A converter will be modeled mathematically with a zero-order hold impulse response, i.e.,

$$g_h(t) = u(t) - u(t - T) \tag{4.11}$$

where u(t) represents the unit-step function. Hence, if the receiver filter is considered to be included in the channel impulse response $c_c(t)$, the total channel response is given by

$$c(t) = c_c(t) * g_h(t) \ . \tag{4.12}$$

With the graphical convolution of Fig. 4-2(b), it can then be seen that

$$c(t) = \begin{cases} \displaystyle\int_0^t c_c(\tau)d\tau & 0 < t \le T \\ \displaystyle\int_{t-T}^t c_c(\tau)d\tau & t > T \ . \end{cases} \tag{4.13}$$

Therefore, it follows from Leibnitz's rule that the derivative exists and can be written as

$$c'(t) = \begin{cases} c_c(t) & 0 < t \le T \\ c_c(t) - c_c(t - T) & t > T \ , \end{cases} \tag{4.14}$$

if $c_c(t)$ is continuous over the interval $0 < t < \infty$. The t = 0 point has been excluded in (4.13) and (4.14) because some channels of interest will have a step discontinuity at the origin. However, all channel responses to be considered will be continuous for $t > 0$ and therefore (4.14) is valid.

4.2 COMPARISON OF EQUALIZER DESIGNS

Results of various design examples are presented in the remainder of this chapter to illustrate some of the distinctive characteristics of digital-filter channel equalization and guidelines for the use of linear programming to design these equalizers. The ease of design and relative performance of the frequency-sampling and transversal equalizers are compared by using both techniques to equalize the channel

$$c_c(t) = 4\pi^2 t\, e^{-2\pi t}\, u(t) \quad \leftrightarrow \quad C_c(f) = \frac{1}{(1 + jf)^2} \qquad (4.15)$$

Typically, the channel unit-impulse response will not be symmetric about the response peak. Consequently, unlike the frequency-sampling filter design examples of Section 3.3, both sides of the pulse must be constrained for equalizer designs.

4.2.1 Frequency-Sampling Equalizer Design

Since the channel described in (4.15) has a normalized break-point frequency $f_b = 1$ Hz, which is assumed to equal the Nyquist bandwidth, $1/2t_b$, it follows that the minimum bit interval, i.e., the Nyquist interval, is $t_b = 0.5$ s. Consequently, the waveform specified for the equalized channel output was the raised-cosine pulse of Fig. 3.4 with $t_b = 0.5$ s, which resulted in the following set of constraints:

$$\begin{array}{llll} y(0.5) = 0 & y(1.5) = 1 & y(2.5) = 0 & y'(1.0) \geq 0 \\ y(0.75) = 0 & y(2.0) = 0 & y(3.0) = 0 & y'(2.0) \leq 0 \\ y(1.0) = 0 & y(2.25) = 0 & y'(0.5) \geq 0 & y'(2.5) \leq 0 \ . \end{array} \tag{4.16}$$

The objective function to be maximized is

$$Z = -y'(0.5) - y'(1.0) + y'(2.0) + y'(2.5) \ , \tag{4.17}$$

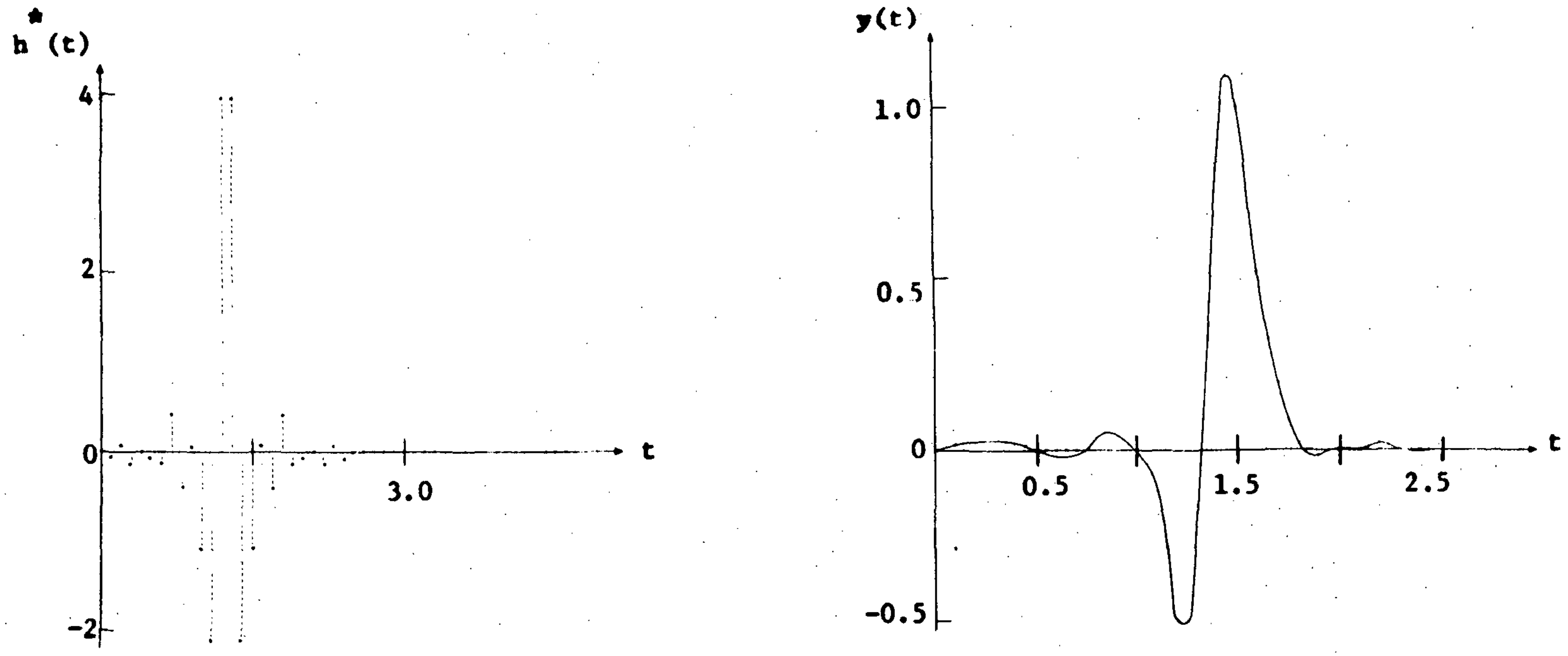
which attempts to flatten the response on both sides of the pulse peak.

The frequency-sampling equalizer design is also not as straightforward as the pulse-shaping filter design in the determination of the parameters N, T, and R. The point NT/2 corresponding to the equalizer impulse response peak no longer coincides with the peak of the response being constrained, namely the channel output, since there is a delay in passage through the channel. Consequently, the time delay must be either estimated and subtracted from the time of the constrained peak to give the proper NT/2 value or it can be obtained by iteration. The delay was estimated from a sketch of the phase characteristic for this channel to be approximately 0.25 s resulting in the value NT = 2.5 s. Thus, for an assumed sampling interval of $T = 0.2t_b$ the corresponding number of impulse response samples is N = 25. The parameter R is left as a variable to be determined by iteration.

The impulse-response output of a known transmission channel, (4.15), which is in cascade with a frequency-sampling equalizer has now been constrained by (4.16) and (4.17) to have a desired impulse response. The equalizer consists of a comb filter with 25 delay elements and a set of R resonators with unknown multiplier coefficients $\{H_k\}$. The LINPO subroutine can now be used to solve this linear-programming design problem to give the $\{H_k\}$. However, the coefficients of the unknown $\{H_k\}$ as defined by (4.6) and (4.7) must be first calculated. Consequently, an auxiliary subroutine called COCAL2, which is described in Appendix C, is used to prepare equalizer design problems for a LINPO solution, similar to the way COCAL1 was used for pulse-shaping filter designs.

The use of COCAL2 and LINPO to solve (4.16) and (4.17) for the unknown H_k is presented as an example in Section C.2. Results of that solution are illustrated in Fig. 4-3 where the 25 discrete values of the impulse response corresponding to the designed equalizer are plotted. Also shown is the output of the data channel when this impulse response is applied to a D/A converter at the channel input. By comparing the channel output with the constraint set of (4.16) it can be seen that all constraints are satisfied. Nevertheless, the pulse shape does not approach the desired characteristic since there is a large negative overshoot. Also, the channel output peak doesn't occur exactly at t = 1.5 s since the maximum channel delay is actually 0.159 s rather than the 0.25 s used to calculate the product NT. Consequently, the equalizer parameters N, T, and R were varied in an attempt to improve the design but no significant improvement could be obtained over the illustrated design of N = 25, T = 0.1 s, and R = 12.

The negative overshoot, which is a characteristic of frequency-sampling equalization, can be rationalized by sketching the graphical convolution of the equalizer output given in Fig. 4-3(a) with the channel impulse response as in Fig. 4.2(a). Considering just the six most significant values of h*(t), it can be seen that the negative overshoot is formed as c(t) translates past the two negative h*(t) values to the left of the h*(t) peak. If these two values were not



(a) Equalizer Output

(b) Channel Output

Fig. 4-3. Frequency-Sampling Filter Equalization.

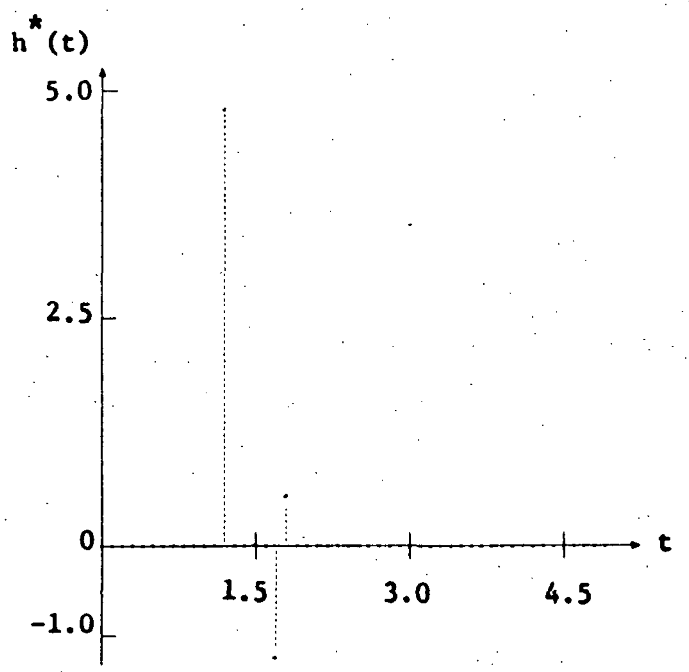
present, the y(t) pulse peak would be monotonically formed without an overshoot as c(t) translates past the two positive h*(t) values. Next, as desired the y(t) pulse would be immediately forced toward zero by the two negative values on the right of the h*(t) peak. Unfortunately, this desired nonsymmetrical impulse response cannot be obtained from a frequency-sampling filter with real multiplier coefficients, and thus the negative overshoot will always be present.

4.2.2 Transversal Equalizer Design.

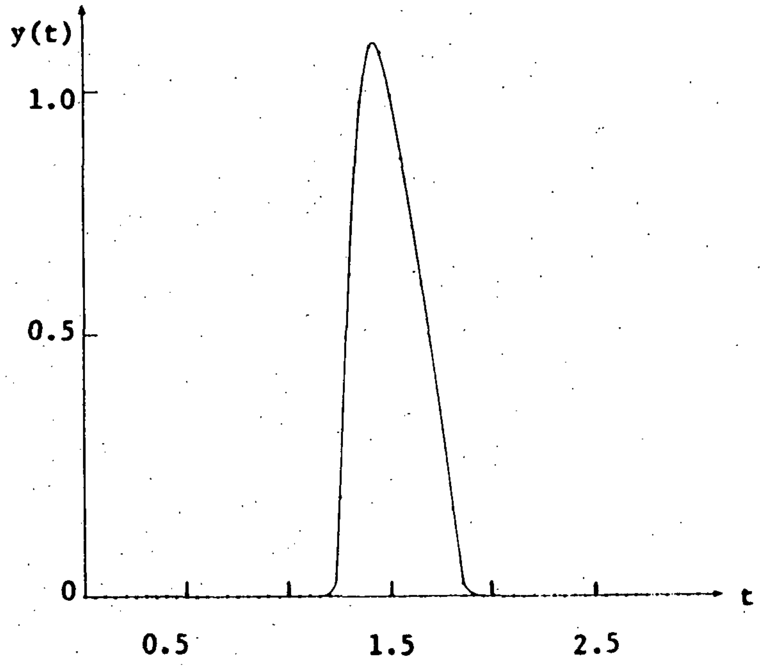
Most of the design problems encountered during the frequency-sampling equalizer design are eliminated when transversal equalizers are considered. Since the transversal filter has the same number of impulse response values as multiplier coefficients, the parameters N and R are equal; thus, the design problem is simplified by reducing the number of unknowns that must be defined. Additionally, unlike the frequency-sampling equalizer, the LINPO requirement of real multiplier coefficients does not imply that the impulse response for this filter need be symmetrical. Since this filter also does not have a characteristic impulse-response peak at NT/2, there is no need to accurately estimate the delay through the channel to define the NT product. The discrete input to the channel is the set of multiplier coefficient values which are applied to the channel in sequence, one every T seconds. Consequently, the primary requirement on the NT product is that the channel's response to the impulse response values in this time interval be of sufficient length to span the constrained output interval.

As an example, a transversal filter was designed to equalize the channel described by (4.15) so that the constraint set of (4.16) and (4.17) would be satisfied. All that remains to be specified for a LINPO solution are the values for N and T. The same parameters are used as in the previous example, namely a sampling interval T = $0.2t_b$, and N = 25 impulse response values. Hence, the resulting NT product of 2.5 s is approximately equal to the constrained time interval and should be more than sufficient for the channel output to satisfy the constraints. It will be shown in Section 4.3 that the sampling interval has a direct effect on the transmitted energy requirement and can be optimized to reduce this energy.

The COCAL2 subroutine was used to calculate coefficients defined by (4.9) and (4.10), and then LINPO was employed to obtain a solution to the design problem. The solution, which is described in more detail in Section C.2, is illustrated in Fig. 4-4. The transversal equalizer requires only three multiplier coefficients as indicated by the three nonsymmetrical impulse response values in Fig. 4-4(a). Since the last response value is at t = 1.8 s, the parameter N could have been reduced to 19 with no effect on the results. However, N will normally be set to a large value to eliminate the problem of choosing exactly its minimum possible value. By using this approach, the LINPO solution will indicate the true number of impulse response values required and specify the excess values to be zero. Thus, the multipliers and shift-register delays corresponding to these zero values need not be implemented, however, the channel response will occur earlier. The channel output produced by applying the three-valued impulse response to the channel is plotted in Fig. 4-4(b).



(a) Equalizer Output



(b) Channel Output

Fig. 4-4. Transversal-Filter Equalization.

All constraint equations are satisfied and there are no overshoots to cause jitter problems.

4.2.3 Timing Jitter Sensitivity

The sensitivities to timing jitter of the two types of digital equalizers were compared by considering the equalization of channels with more severe characteristics. Both frequency-sampling and transversal filters were designed to equalize the channels

$$c_c(t) = \frac{(2\pi)^n}{(n-1)!} t^{n-1} e^{-2\pi t} u(t) \qquad n = 3, 4, 5 \tag{4.18}$$

in the manner of the previous two design examples with the same constraint set, T, N, and R values. Channel outputs, y(t), were then plotted so that the intersymbol interference (ISI) distortion could be calculated for each design as a function of timing jitter. This distortion is defined by

$$D = \sum_{\substack{\ell=0 \\ \ell\neq 3}}^{\infty} |y(\ell t_b + t_p)| \tag{4.19}$$

where the time t_p is the percentage of the Nyquist interval that the sampling times at the receiver are displaced by jitter. The distortion was next expressed as a percentage of the pulse peak, $y(3t_b)$ for each design and plotted in Fig. 4-5 as a function of t_p. It is obvious that there is an appreciable advantage in using transversal-filter equalization for channels where jitter may be present. Furthermore, even if there was perfect timing, a frequency sampling equalizer could not be designed with LINPO for the n = 5 channel in

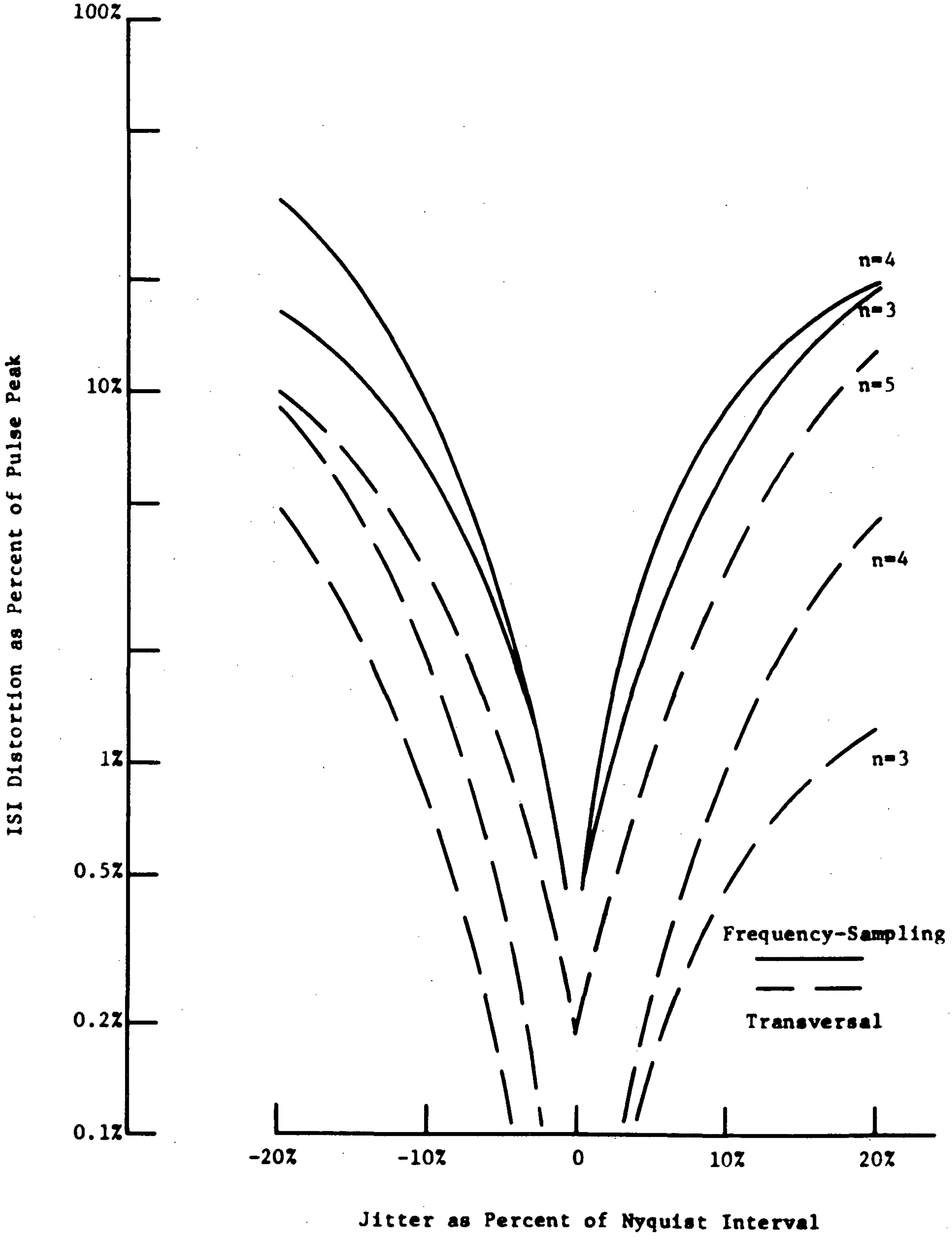


Fig. 4-5. Comparison of Timing-Jitter Degradation.

(4.18). Since the transversal equalizer has the inherent advantages of being less sensitive to jitter and easier to design, the remainder of this chapter will be devoted to its design.

4.3 OTHER TRANSVERSAL-EQUALIZER CONSIDERATIONS

It has been shown that the only unknowns to be defined before using LINPO for a transversal-equalizer design are a set of constraint equations; N, the total number of impulse response values; and T, the sampling interval. The selection of these parameters will now be considered.

4.3.1 Effect of Constraint Set

Only one constraint-equation set, (4.16), was used in the previous section to design the transversal equalizer. Consequently, the effects of other constraint sets on the equalizer design were determined for the channel of (4.18) with n = 6, for which the distortion effects were more pronounced. Summarized in Table 4.1 are nine of the sets of equations that were used. Starting with the set of seven constraints (NC = 7)

$$\begin{array}{lll} y(1.0) = 0 & y(1.5) = 1 & y(2.0) = 0 \\ y(1.25) \geq 0.5 & y(1.75) \geq 0.5 & y'(1.0) \geq 0 \\ & y'(2.0) \leq 0 \ , & \end{array} \qquad (4.20)$$

additional constraints were added for each set, usually another time point at which y(t) was to be forced to zero. The parameters T = 0.1 s and N = 40 were used for all designs. For complete elimination of ISI distortion, y(t) must be forced to zero at every time

TABLE 4.1

CONSTRAINT SETS

		NUMBER OF CONSTRAINTS (NC)								
EQUATIONS	t	7	11	12	13	14	15	16	17	21
$y(t) = 0$	0.50		X	X	X	X	X	X	X	X
	0.75							X	X	X
	1.00	X	X	X	X	X	X	X	X	X
	2.00	X	X	X	X	X	X	X	X	X
	2.25								X	X
	2.50		X	X	X	X	X	X	X	X
	2.75									X
	3.00			X	X	X	X	X	X	X
	3.25									X
	3.50				X	X	X	X	X	X
	3.75									X
	4.00					X	X	X	X	X
	4.25									X
	4.50						X	X	X	X
$y(t) \geq .5$	1.25	X	X	X	X	X	X	X	X	X
	1.75	X	X	X	X	X	X	X	X	X
$y(t) = 1$	1.50	X	X	X	X	X	X	X	X	X
$y'(t) \geq 0$	0.50		X	X	X	X	X	X	X	X
	1.00	X	X	X	X	X	X	X	X	X
$y'(t) \leq 0$	2.00	X	X	X	X	X	X	X	X	X
	2.50		X	X	X	X	X	X	X	X

point which is an integer multiple of t_b from the pulse peak. If only a finite number of such points are so constrained, some residual distortion will exist. This is illustrated by the transversal filter curve (n = 5) in Fig. 4-5, where for zero jitter a 0.2% residual distortion still exists. Hence, the number of time points at which y(t) was constrained to be zero was increased until the residual distortion became insignificant. Typical output pulses resulting from the designs are plotted in Fig. 4-6, with other results summarized in Table 4.2. The tabulated results for NC = 11 through 15 indicate that an additional multiplier is required in the equalizer for each

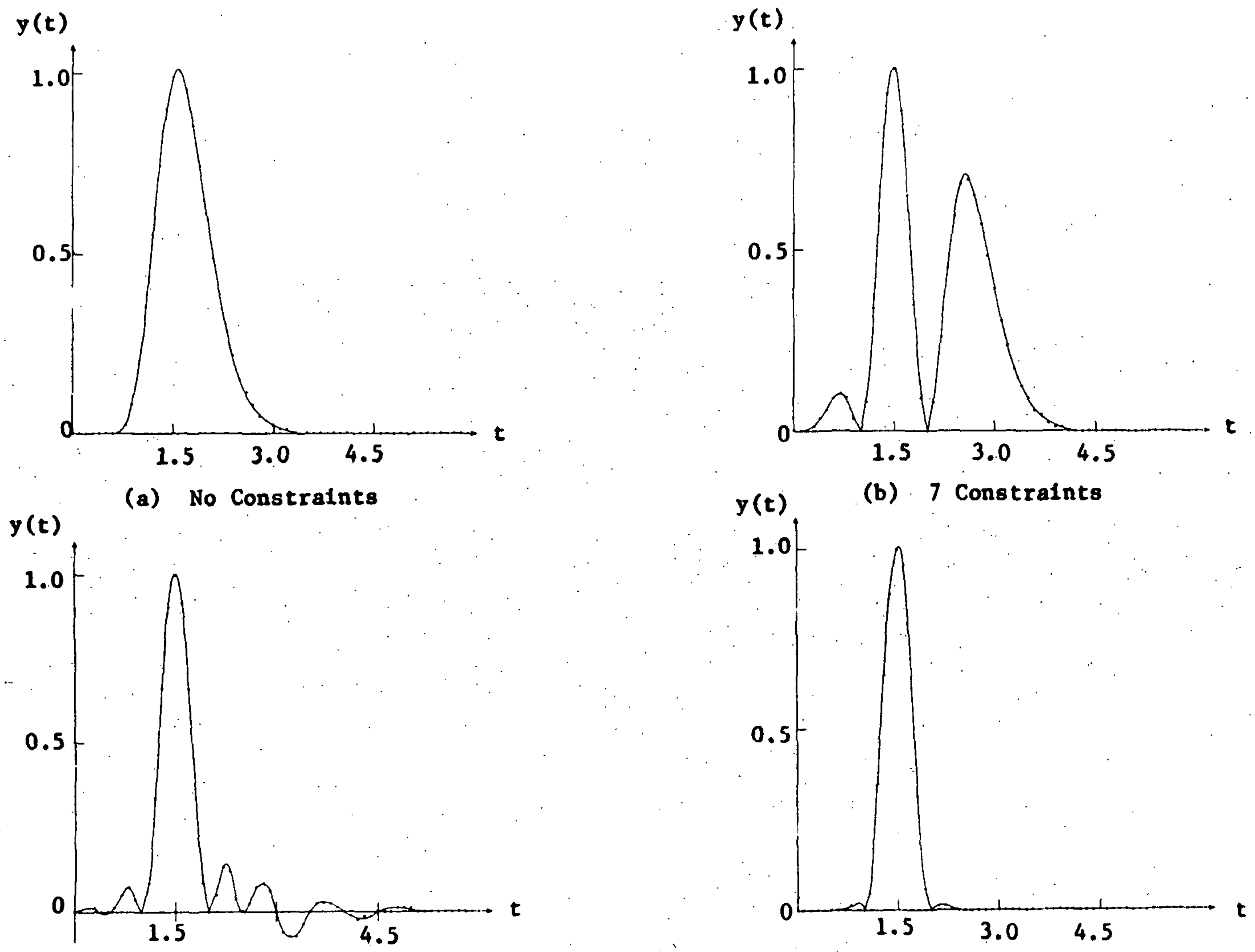
TABLE 4.2

COMPARISON OF EQUALIZER CHARACTERISTICS

Number of Constraints	Multipliers Required	Residual ISI Distortion	Energy Transmitted
7	7	1.240	75.6
11	11	1.739	120.8
12	12	0.222	149.7
13	13	0.083	150.0
14	14	0.035	150.1
15	15	0.015	150.1
16	15	0.015	344.8
17	17	0.002	554.3
21	20	0.000	556.0

additional time point constrained to zero at an integer multiple of t_b from the peak. However, the energy transmitted to the baseband channel, i.e.,

$$E_T = \int_0^\infty g^2(t)dt = T \sum_{m=0}^{N-1} h_m^2 , \tag{4.21}$$



(a) No Constraints

(b) 7 Constraints

(c) 15 Constraints

(d) 21 Constraints

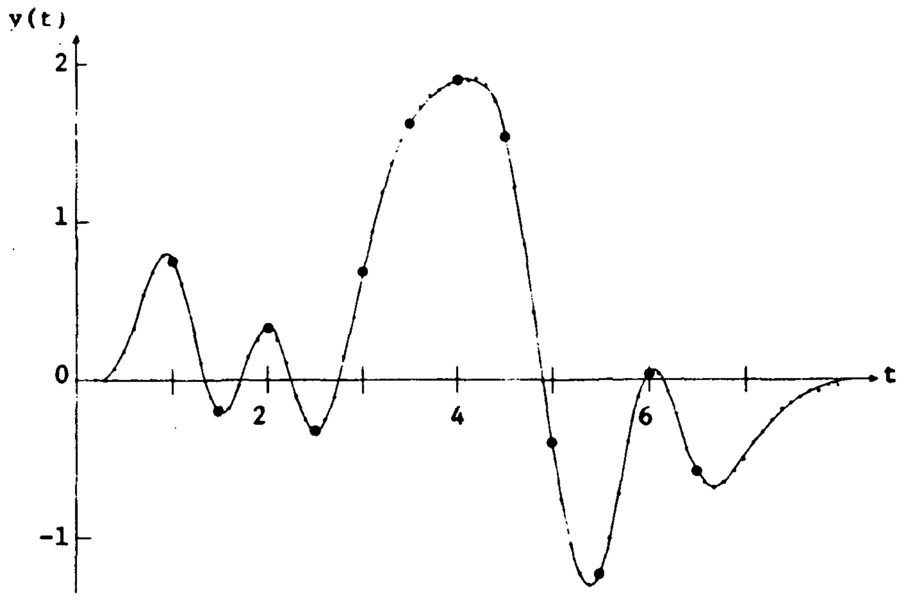
Fig. 4-6. Pulse Shape Sensitivity to Constraints.

where $g(t) \triangleq h^*(t) * g_h(t)$ approaches a constant value. The residual distortion decreases with each added constraint and is approaching zero; consequently, the multiplier value h_m required to satisfy each added constraint also is approaching zero. Therefore, the summation of (4.21) approaches a constant value. It can be seen from Table 4.1 that the last three constraint sets, NC = 16 through 21, force zeros at time points between those resulting from the first six sets. These constraints result in the ISI distortion being reduced to an insignificant value at the cost of additional multipliers and a greatly increased transmitted energy. Attempting to force $y(t)$ to be zero at time points this close together requires components of $g(t)$ to be transmitted at higher frequencies where the channel attenuation is much greater.

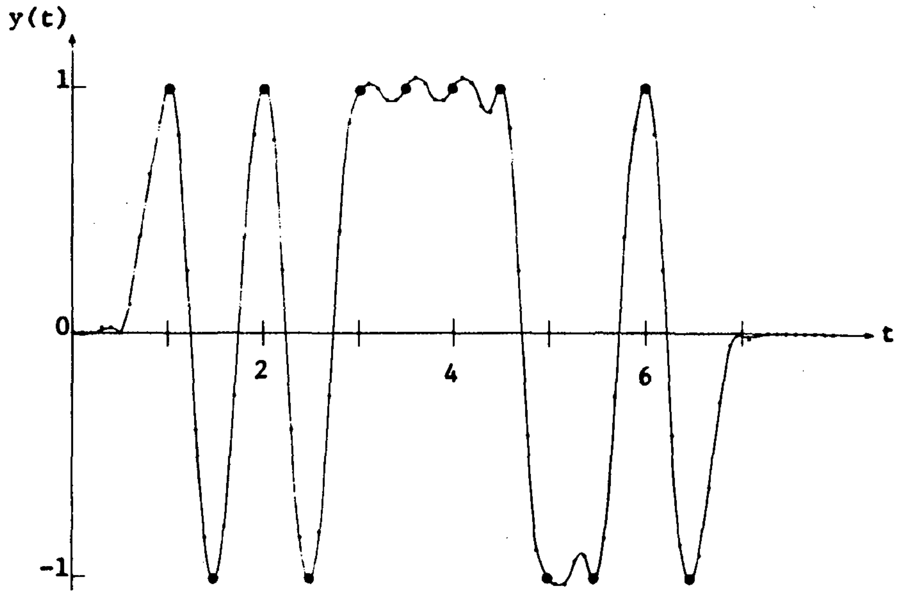
The binary data sequence 101011110010 was transmitted through the unequalized channel and then through the same channel using the NC = 15 constraint equalizer to illustrate the improvement obtained by using this design technique. Since the time between bits of the data sequence is 0.5 s and the sampling interval of the equalizer is 0.1 s, four equally-spaced zeros must be applied to the equalizer between each bit of the sequence so that the proper waveform will be generated. Comparison of the received waveforms in Fig. 4-7 shows that the equalized channel output has no intersymbol interference. Conversely, for the unequalized channel, detection errors are possible even in the absence of noise.

4.3.2 Effect of Sampling Interval

It has been indicated previously that N, the number of impulse response values, should be made quite large so as to insure that



(a) No Equalization



(b) Transversal-Filter Equalization

Fig. 4-7. Effect of Equalization on Data Sequence 101011110010.

the product NT spans the constrained output time interval. Only the pertinent number of shift register stages will be retained in the final design. This leaves only the effect of the parameter T to be determined.

The value of T is bounded on the lower end by the available logic speed and on the upper end by the Nyquist interval, i.e., $t_c \le T \le t_b$, where t_c is the total time required for any one of the N multiplications plus one multi-input addition needed to calculate each transversal filter output sample. Hence, the effect of varying the parameter T between these limits was observed for equalizers designed with the NC = 15 constraint set and N = 40. The most significant result from these designs was the change in transmitted energy E_T defined by (4.21). These changes are displayed in Fig. 4.8 for a number of channels. As T approaches t_b in value there is a great increase in required energy. A small percentage of the increase is a consequence of multiplying by T in (4.21), but the primary increase is caused by the reduction in bandwidth of the g(t) pulse applied to the channel. As T increases, the high frequency components of g(t) are severely attenuated as a result of the $\sin(\pi fT)/\pi fT$ frequency characteristic of the D/A converter. However, higher frequency components of g(t) must still be transmitted to satisfy the constraints on y(t). Hence, a very large increase in the transmitted energy. Conversely, as T approaches zero the bandwidth of the transmitted pulse becomes excessively large and since there is no energy minimization in the design constraints, these higher frequency components are not forced to zero, resulting in a slight increase in the transmitted energy.

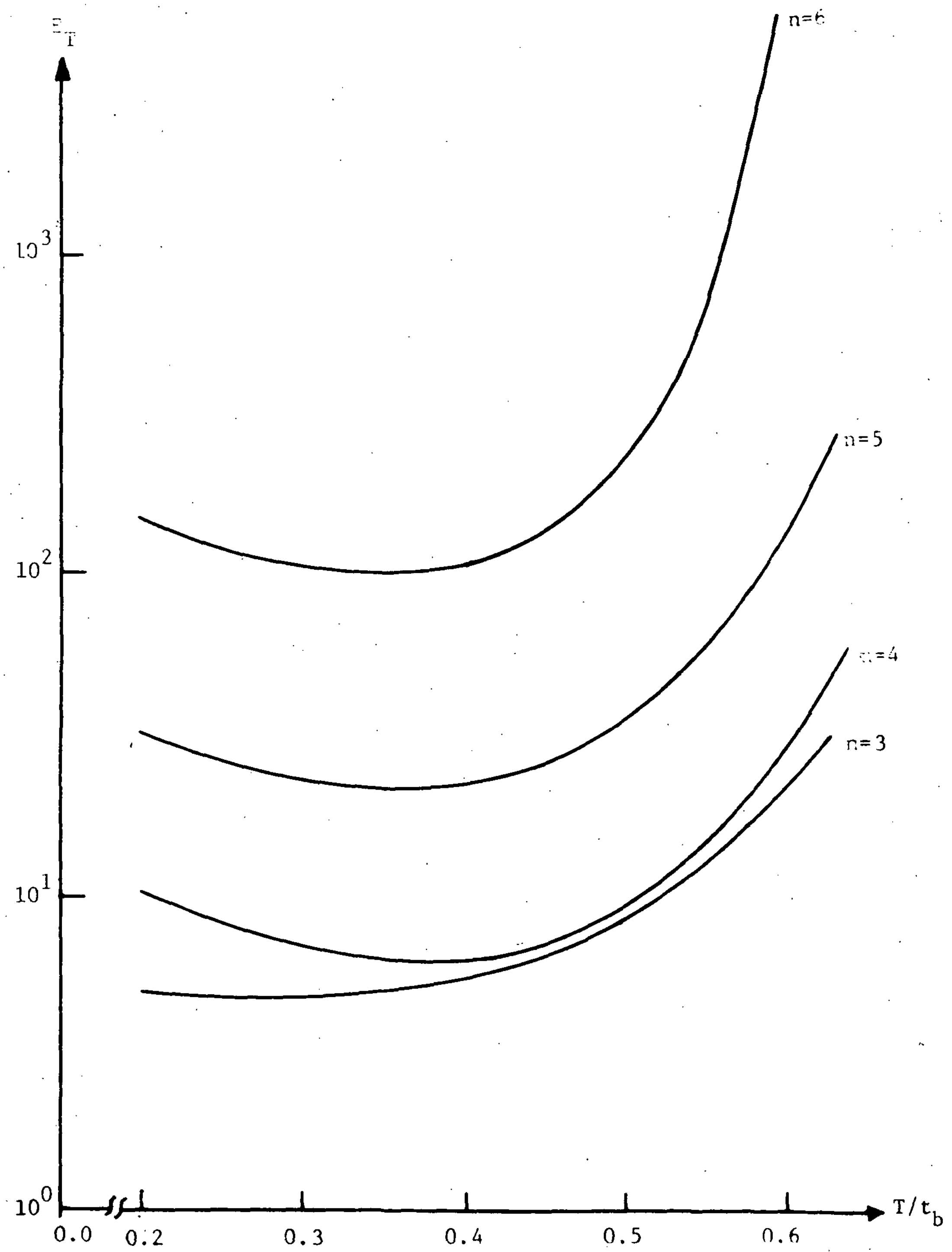
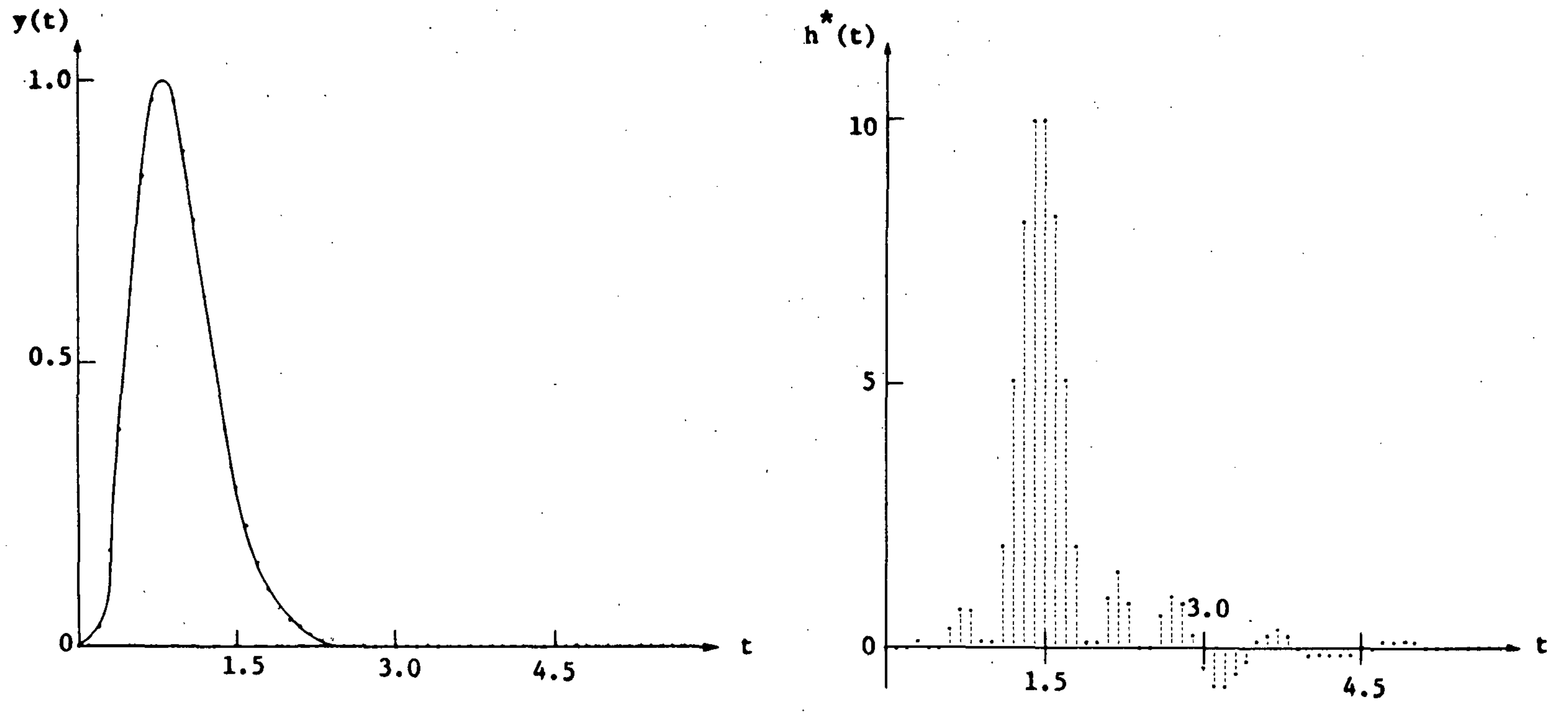


Fig. 4-8. Comparison of Transmitted Energy Requirements.

It can be observed from the curves of Fig. 4-8 that a ratio of $T/t_b \approx 0.4$ minimizes the transmitted energy. For example, if the equalizer with NC = 15 and T = 0.1 s were redesigned using T = 0.2 s the transmitted energy would be reduced from 150.1 V^2s as indicated in Table 4.2 to approximately 100 V^2s. It was also observed from the series of equalizer designs that the tails of the received pulses increased in magnitude as T varied from the minimum energy value of $0.4t_b$. The number of multiplier coefficients required did not change appreciably as T varied.

4.3.3 Location of Equalizer

It has been assumed throughout this chapter that the equalizer was positioned at the channel input, because there is a speed advantage in using this location for high data-rate applications. If it is assumed that $0.4t_b = T = t_c$ then the maximum data rate through the equalized channel is $1/t_b = 0.4/t_c$. For the normal transversal filter with N multipliers, t_c consists of the time required to perform one multiplication and one multi-input addition. However, with the equalizer at the channel input, multiplication is not required. Only one of two possible bits, a ONE or ZERO, will be applied to the equalizer input, and eventually to the equalizer multipliers, during each sampling interval. Therefore, each multiplier can be replaced with a storage register which contains the multiplier coefficient; hence, t_c is the time required to read and add the appropriate coefficients. By reducing t_c, the maximum data rate has been increased. Multipliers would normally be required for equalizers at the channel output, since there is a wide range of possible inputs from the analog channel. However, there are advantages to equalization at the channel output. For example, in the case of the adaptive equalizer there would be no need to transmit the impulse responce c(t) back to the channel input in order to update the equalizer coefficients. If equalization is desired at the channel output, the linear programming design procedure of this chapter is still valid. This is illustrated by comparing the channel output of Fig. 4-6(c) with the equalizer output plotted in Fig. 4-9(b). The former response was obtained with a transversal equalizer designed and used at the channel input, while the latter is the output with the equalizer at the receiver.



(a) Channel Output (b) Equalizer Output

Fig. 4-9. Equalization at Channel Output.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

It has been shown that linear programming optimization techniques can be employed effectively in the time domain to design finite-duration impulse-response digital filters. It has been further demonstrated that this technique has direct application to the design of frequency-sampling filters to be used for pulse generation or matched-filter detection, and both frequency-sampling and transversal filters to be used for equalization of transmission channels with known impulse responses.

5.1 CONCLUSIONS

Frequency-sampling filters, which have not been designed previously in the time domain by any technique, can be defined readily by using linear programming to specify filter multiplier coefficient values. It was illustrated with examples that a simple design procedure which can be employed is to: (1) constrain the impulse response equation of the filter to have desired values at critical time points, (2) define the number of impulse-response values, maximum number of resonators and sampling-interval parameters for the filter, and (3) solve the resulting set of constraint equations with a linear programming algorithm. It was also shown that the number of response values and sampling-interval parameters are defined by the significant duration and point of symmetry of the filter impulse response, whereas the number of resonators are defined by the impulse response bandwidth, if available, or by iteration. Comparisons of filter designs with desired pulses indicate that this technique yields very accurate pulse shapes with minimum hardware requirements.

This design procedure can be extended readily to the design of digital equalizers for known transmission channels by constraining the output of an equalizer-channel cascade rather than a filter impulse response. It has been concluded by studying the equalization of typical transmission channels that the transversal equalizer is easier to design, requires considerable less hardware, and is less sensitive to timing jitter than the frequency-sampling equalizer. It was further observed from a study of transversal equalizer constraint sets that an additional multiplier is required for each zero specified in the output response when spaced at Nyquist time intervals, but that the required transmission energy approaches a constant value. However, if zeros are forced in the response at closer time intervals, the transmitted energy requirements greatly increase. Results of varying the sampling interval in a series of equalizer designs indicate that this energy can be minimized by proper choice of the ratio of sampling to Nyquist intervals.

The ability developed in this study to work directly from channel time-domain characteristics allows the effect of both amplitude and phase distortions to be considered simultaneously. Unlike frequency-domain designs, which usually require separate filters to compensate for each source of distortion, one equalizer can be designed easily to compensate for both impairments. Furthermore, the design technique is valid for equalizers positioned at either the transmitter or receiver.

5.2 RECOMMENDATIONS FOR FURTHER STUDY

Several possibilities exist for improvement in the equalizer design technique of this study. The feasibility of modifying or replacing the linear programming optimization algorithm should be investigated. For example, if an objective function in the form of

$$Z = c_1 x_1^2 + c_2 x_2^2 + \ldots + c_R x_R^2 \tag{5.1}$$

was used, the transmitted energy (4.21) could be minimized directly. This would require a nonlinear programming routine because of the nonlinear objective function. Another possibility is to use integer linear programming, i.e., the variables x_i of the problem are integers. The integer variables would more accurately represent the possible multiplier coefficients, because the coefficients must be implemented with a finite number of bits. Thus, only a discrete, equally-spaced set of values would be possible solutions to the linear programming problem.

The linear programming technique could be applied to the design of other classes of recursive digital filters. Since these filters typically have an infinite impulse response, it may be possible to minimize the residual intersymbol interference with fewer multiplier coefficients using less transmitted energy than the transversal equalizer requires.

Finally, the use of the time-domain linear-programming technique to design filters should be investigated for other applications. For example, the design of resonators for digital formant synthesizers would probably be a good application. Since the speech process can be described as the excitation of resonators by a series of impulses, specifications for synthesizer designs are often in the form of desired impulse responses.

APPENDIX A

LINEAR PROGRAMMING FUNDAMENTALS

The general linear-programming problem is to solve for the R non-negative variables x_j which maximize the objective function

$$Z = c_1x_1 + c_2x_2 + \ldots + c_Rx_R \,, \tag{A.1}$$

subject to the M linear constraint equations

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{iR}x_R \left\{ \begin{matrix} \leq \\ = \\ \geq \end{matrix} \right\} b_i \quad i = 1,2,\ldots,M \,. \tag{A.2}$$

One solution to this problem, namely the simplex procedure outlined in Section 3.2, is the subject of this appendix. Theoretical aspects of the simplex method are discussed first, followed by an example to illustrate the method.

The simplex procedure involves a series of iterations, each of which generates a new basic feasible solution to (A.2) that yields a greater value of the objective function than the previous solution. The procedure continues until a finite maximum value of the objective function is obtained or an unbounded solution is indicated. The aforementioned terms, e.g., basic feasible solution, were defined in Section 3.2.

Any inequalities in (A.2) are converted to equalities by the inclusion of non-negative slack and surplus variables. For example, assuming in (A.2) that "$\leq$" holds for U equations, "$\geq$" for V-U equations, and "=" for M-V equations then

$$\sum_{j=1}^{R} a_{hj}x_j + x_{R+h} = b_h \qquad h = 1,2,\ldots,U\ , \qquad \text{(A.3a)}$$

$$\sum_{j=1}^{R} a_{kj}x_j - x_{R+k} = b_k \qquad k = U+1,\ldots,V\ , \qquad \text{(A.3b)}$$

and

$$\sum_{j=1}^{R} a_{pj}x_j = b_p \qquad p = V+1,\ldots,M\ . \qquad \text{(A.3c)}$$

Hence, there are U slack variables x_{R+h} and V-U surplus variables x_{R+k}. The addition of these variables will have no effect on the final solution since they will be added to the objective function with a zero cost, i.e., a coefficient $c_j = 0$, $j = R+1, R+2,\ldots,N$.

The M equations (A.3) in N unknowns, N = R+V, can be written in vector notation* as

$$\underline{A}\ \underline{x} = x_1\underline{a}_1 + \ldots + x_N\underline{a}_N = \underline{b}\ , \qquad \text{(A.4)}$$

with the objective function written as

$$Z = c_1x_1 + \ldots + c_Nx_N = \underline{c}\ \underline{x} \qquad \text{(A.5)}$$

It will be assumed that the rank of $\underline{A}$ is M and that a solution to (A.4)

*A general matrix, e.g., $\underline{A}$ will be denoted by a capital letter, whereas a single-row or single-column matrix, called a vector, will be denoted by a lower-case letter, e.g., $\underline{x}$. A column vector will be denoted by parentheses, e.g., $\underline{x} = (x_1,x_2,\ldots,x_N)$ and a row vector by brackets, e.g., $\underline{c} = [c_1,c_2,\ldots,c_N]$.

exists, i.e., $N \geq M$. If $N = M$, there is a unique solution to the problem and there is no need for linear programming. On the other hand if $N > M$ there are a number of solutions and linear programming is used to determine the optimal solution. If either of the two assumptions does not hold, the simplex method will indicate that either there is no solution (due to inconsistent constraints) or there is redundancy in the constraint equations.*

A.1 ALGEBRAIC FORMULATION OF SIMPLEX PROCEDURE

The simplex solution of the general linear-programming problem is an algebraic technique, which is best described in linear algebra terminology. Any set of M linearly independent columns of $\underline{A}$ forms a so-called "basis" for the vector space (E^M) and the matrix $\underline{B}$ formed from these M columns is called the "basis matrix". Each column of $\underline{B}$ is identical to a column of $\underline{A}$, but the order of the columns in $\underline{B}$ may be different from $\underline{A}$. Corresponding to each basis matrix there exists a "basic feasible solution", $\underline{x}_B$, to (A.4), i.e.,

$$\underline{x}_B = \underline{B}^{-1}\,\underline{b} = (x_{B1},\ldots,x_{BM}) \quad . \tag{A.6}$$

The elements of the column vector $\underline{x}_B$ are called "basic variables" and represent values for M of the N unknown variables x_i of (A.4). The variable x_i which corresponds to the basic variable is determined from the column order of $\underline{A}$ in the basis matrix $\underline{B}$, e.g., if the third column

*For a further discussion of inconsistency and redundancy see Section 4.6 of Hadley [67].

of $\underline{A}$ is the first column of $\underline{B}$ then $x_{B1} = x_3$, etc. The remaining N-M variables will be assigned zero values.

For each basic solution, (A.5) can be written as

$$Z = c_{B1}x_{B1} + \ldots + c_{BM}x_{BM} = \underline{c}_B \, \underline{x}_B \; . \tag{A.7}$$

The component of $\underline{c}$ that corresponds to c_{Bi} is again determined by the column order of $\underline{A}$ in $\underline{B}$, e.g., $c_{B1} = c_3$ if the third column of $\underline{A}$ is the first column of $\underline{B}$. The other N-M coefficients of $\underline{c}$ have no effect on (A.7) since they are multiplied by zero-valued variables. The problem of selecting the columns of $\underline{A}$ which constitute the initial basis matrix will be discussed in Section A.3.

Each iteration of the simplex procedure consists of replacing one column of the basis matrix $\underline{B}$ with one of the N-M columns of $\underline{A}$ not in the basis at that time. This substitution must result in a feasible basic solution (not just a basic solution which could contain negative variables) and increase the value of Z in (A.7). Consequently, a procedure will be discussed in the remainder of this section for determining the column to remove from $\underline{B}$ so that a feasible solution is obtained and the column from $\underline{A}$ to substitute into $\underline{B}$ so that Z is increased. However, certain variables must be defined first.

Any column of $\underline{A}$ can be written as a linear combination of the columns of $\underline{B}$ since $\underline{B}$ is a basis matrix. For example, the j^{th} column of $\underline{A}$ is

$$\underline{a}_j = y_{1j} \, \underline{b}_1 + \ldots + y_{Mj} \, \underline{b}_M = \underline{B} \, \underline{y}_j \; . \tag{A.8}$$

The subscript order of the vector $\underline{y}_j$ components is important. The first subscript refers to a column of $\underline{B}$ that the component is multiplied by and the second refers to the column of $\underline{A}$ that is being expressed as a linear combination. Another function of the vector $\underline{y}_j$ is to define a parameter, z_j, where

$$z_j = y_{ij}\ c_{B1} + \ldots + y_{Mj}\ c_{BM} = \underline{c}_B\ \underline{y}_j \tag{A.9}$$

which is used to indicate possible increases in Z.

It has been shown [67] that the column from $\underline{A}$ to be inserted into the basis should be $\underline{a}_k$ where k is defined by*

$$z_k - c_k = \min_j \{z_j - c_j;\ z_j - c_j < 0\} \tag{A.10}$$

and that the column of $\underline{B}$ to be removed is $\underline{b}_r$ where

$$\frac{x_{Br}}{y_{rk}} = \min_i \left\{ \frac{x_{Bi}}{y_{ik}}\ ;\ y_{ik} > 0 \right\}. \tag{A.11}$$

These substitutions will insure a basic feasible solution after each iteration and will on the average give the largest possible increase in Z per iteration. The cost parameters, c_j in (A.10), are constants fixed by the objective function, but the parameters z_j must be recalculated after each iteration. Likewise, the parameters x_{Bi} and y_{ik} of (A.11) must be recalculated after each iteration. The repetitive calculations that are required for each iteration can be summarized as follows:

*The set notation $\{a_j;\ f(a_j)\}$ is used to define the set of elements a_j which satisfy some restriction $f(a_j)$. Consequently, the element or elements in that set which have the smallest value are represented by $\min_j \{a_j,\ f(a_j)\}$.

(1) Calculate the basic solution $\underline{x}_B$ with (A.6) and the new basis matrix $\underline{B}$.

(2) Calculate the vector $\underline{y}_j$ with (A.8) for every $\underline{a}_j$ not in the basis.

(3) Calculate z_j with (A.9) for every $\underline{y}_j$ from step (2).

(4) Calculate $z_j - c_j$ for every result of steps (2) and (3).

(5) Select the column $\underline{a}_k$ to enter the basis with (A.10).

(6) Calculate the ratio x_{Bi}/y_{ik} for every $y_{ik} > 0$ of $\underline{y}_k$.

(7) Select the column $\underline{b}_r$ to leave the basis with (A.11).

(8) Calculate the new basis matrix by substituting $\underline{a}_k$ for $\underline{b}_r$.

This procedure can end in one of two ways. If $z_j - c_j \geq 0$ for all $\underline{a}_j$ not in the basis, then the last basic feasible solution is an optimal solution. If there exists some column $\underline{a}_j$ not in the basis such that $z_j - c_j < 0$ and $y_{ij} \leq 0$ for all $i = 1,2,\ldots,M$ then an unbounded solution exists.

A.2 EXAMPLE USING SIMPLEX PROCEDURE

The example that was solved by graphical techniques in Section 3.2 will now be worked by the simplex procedure to illustrate the steps of each iteration. The problem was to maximize

$$Z = x_1 + x_2$$

subject to the constraints

$$x_1 + x_2 \geq 2 \ ,$$

$$3x_1 + x_2 \leq 15 \ ,$$

and

$$x_1 + 2x_2 \leq 9 \ .$$

After adding one surplus and two slack variables, the appropriate matrices are given by

$$\underline{A} = \begin{bmatrix} 1 & 1 & -1 & 0 & 0 \\ 3 & 1 & 0 & 1 & 0 \\ 1 & 2 & 0 & 0 & 1 \end{bmatrix} ,$$

$$\underline{b} = (2,\ 15,\ 9) ,$$

and

$$\underline{c} = [1,\ 1,\ 0,\ 0,\ 0] .$$

As will be illustrated in Section A.3 a basis matrix which will give an initial feasible basic solution is

$$\underline{B} = (\underline{a}_1,\ \underline{a}_4,\ \underline{a}_5) = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} ,$$

where the corresponding solution is

$$\underline{x}_B = B^{-1}\ \underline{b} = (2,\ 9,\ 7) .$$

The vectors not in the basis are $\underline{a}_2$ and $\underline{a}_3$. Hence, from (A.8)

$$\underline{y}_2 = B^{-1}\ \underline{a}_2 = (1,\ -2,\ 1)$$

and

$$\underline{y}_3 = \underline{B}^{-1}\ \underline{a}_3 = (-1,\ 3,\ 1) .$$

The solution is not unbounded since at least one $y_{ij} > 0$ for both vectors. From (A.9),

$$z_2 = \underline{c}_B\ \underline{y}_2 = [1,\ 0,\ 0]\ (1,\ -2,\ 1) = 1$$

and

$$z_3 = \underline{c}_B\ \underline{y}_3 = -1 .$$

Since

$$z_2 - c_2 = 1 - 1 = 0 ,$$

and

$$z_3 - c_3 = -1 - 0 = -1 ,$$

it follows from (A.10) that the vector to be entered is y_3, i.e., $k = 3$. Since two $\underline{y}_3$ components are positive, it follows from (A.11) that two ratios must be calculated, i.e.,

$$\frac{x_{B2}}{y_{23}} = \frac{9}{3} = 3 \quad \text{and} \quad \frac{x_{B3}}{y_{33}} = \frac{7}{1} = 7 ,$$

which implies $r = 2$; hence $\underline{b}_2$ (or equivalently $\underline{a}_4$) must be removed. Thus the second iteration starts with

$$\underline{B} = (\underline{a}_1, \underline{a}_3, \underline{a}_5) = \begin{bmatrix} 1 & -1 & 0 \\ 3 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} .$$

Execution of the second repetition yields

$$\underline{x}_B = (5, 3, 4)$$

for the basic solution with

$$\underline{y}_2 = (1/3, - 2/3, 5/3) ,$$

$$\underline{y}_4 = (1/3, 1/3, - 1/3) ,$$

$$z_2 - c_2 = -2/3 ,$$

and

$$z_4 - c_4 = 1/3 ,$$

for the remaining variables. Consequently, the vector to be entered is given by $k = 2$, while the ratios

$$\frac{x_{B1}}{y_{12}} = 15 \qquad \text{and} \qquad \frac{x_{B3}}{y_{32}} = \frac{12}{5}$$

imply that $\underline{b}_3 = \underline{a}_5$ is replaced with $\underline{a}_2$. The new basis matrix is

$$\underline{B} = (\underline{a}_1, \underline{a}_3, \underline{a}_2)$$

and the corresponding basis solution is

$$\underline{x}_B = (4.2, 4.6, 2.4) .$$

It then follows that

$$\underline{y}_4 = (0.4, 0.2, -0.2) ,$$

$$\underline{y}_5 = (-0.2, 0.4, 0.6) ,$$

$$z_4 - c_4 = 0.2 ,$$

and

$$z_5 - c_5 = 0.4 .$$

Since $z_j - c_j \geq 0$ for the two vectors not in the basis, the basic feasible solution is optimal. Hence, the complete solution is

$$\underline{x} = (4.2, 2.4, 4.6, 0, 0) ,$$

and the corresponding maximum value of the objective function is

$$Z = \underline{c}\ \underline{x} = 6.6 .$$

A.3 PROCEDURE FOR FINDING INITIAL BASIC SOLUTION

The initial basic feasible solution is obtained by always starting the procedure with an identity basis matrix, $\underline{I}$. If an M-by-M identity matrix can be obtained by manipulation of the columns in $\underline{A}$, as would be the case if all M constraint equations require slack variables, these M columns are used for the first basis matrix. Then

$$\underline{x}_B = \underline{B}^{-1}\,\underline{b} = \underline{I}\,\underline{b} = \underline{b}\ .$$

Since $\underline{b}$ is required to be non-negative, the initial basic solution is feasible.

If a complete I matrix does not appear in $\underline{A}$, a new set of constraint equations are used so that an I matrix will be available, namely,

$$\underline{A}\,\underline{x} + \underline{I}\,\underline{x}_a = \underline{b}\ .$$

The vector $\underline{x}_a$ represents artificial variables that are added to the original constraint equations with unity coefficients so that an $\underline{I}$ matrix can be formed. The components of the artificial vector that correspond to constraint equations with slack variables will be zero.

To illustrate the use of artificial variables consider starting the example in Section A.2 without being given the initial basis matrix. The constraint equations with one artificial variable added are

$$x_1 + x_2 - x_3 \qquad\qquad + x_{a1} = 2\ ,$$

$$3x_1 + x_2 \qquad + x_4 \qquad\qquad = 15\ ,$$

and

$$x_1 + 2x_2 \qquad\qquad + x_5 \qquad = 9\ ,$$

and the expanded coefficient matrix is

$$\underline{A} = \begin{bmatrix} 1 & 1 & -1 & 0 & 0 & 1 \\ 3 & 1 & 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The initial basis matrix is

$$\underline{B} = [\ \underline{e}_1, \underline{a}_4, \underline{a}_5\ ] ,$$

where $\underline{e}_1$ represents the artificial vector. Then the initial basic feasible solution is $\underline{x}_B = \underline{b}$. However, this is a solution to the new constraint set, not the original set. Any solution to both sets must have $\underline{x}_a = 0$. Hence, the iterations must drive the nonzero components of $\underline{x}_a$ to zero. This can be done with a two-phase objective equation.

During the first phase, the artificial variables are assigned a cost of -1 and the other variables a cost of zero. Thus, the first phase objective function

$$Z^* = - \sum_{i=1}^{M} x_{ai}$$

will be maximized by the simplex procedure. Since the simplex procedure prevents variables from becoming negative, the maximum value of Z* is zero and all artificial variables will be zero when this maximum is reached. At this point, which is the end of Phase 1, a basic feasible solution to the original problem is available. If the maximum value of Z* is negative, the artificial variables cannot be driven to zero and no feasible solution to the problem exists.

The basic feasible solution which exists at the end of Phase 1, where max {Z*} = 0, is not optimal. Therefore, Phase 2 consists of assigning the original costs to the legitimate variables and costs of zero to the artificial variables. The simplex procedure then obtains the optimal solutions in the normal manner.

Continuing Phase 1 for the example, the columns of $\underline{A}$ not in the first basis are $\underline{a}_1$, $\underline{a}_2$, and $\underline{a}_3$. Since $\underline{c}_B = (-1, 0, 0)$, it follows that

$$\underline{y}_1 = (1, 3, 1) ,$$

$$\underline{y}_2 = (1, 1, 2) ,$$

and

$$\underline{y}_3 = (-1, 0, 0) .$$

Then, since all non-artificial variables have zero cost values,

$$z_1 - c_1 = -1 ,$$

$$z_2 - c_2 = -1 ,$$

and

$$z_3 - c_3 = 1 .$$

Hence, k = 1 and the ratios are

$$\frac{x_{B1}}{y_{11}} = 2, \quad \frac{x_{B2}}{y_{21}} = 5, \quad \text{and} \quad \frac{x_{B3}}{y_{31}} = 9 .$$

Since these ratios imply that r = 1, column $\underline{b}_1 = \underline{e}_1$ is removed from the basis and replaced with $\underline{a}_1$. This gives

$$\underline{B} = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad \text{and } \underline{x}_B = (x_1, x_4, x_5) .$$

The artificial variable has been removed from the basis. Since any variable not in the basis is assigned a zero value it follows that

$$Z^* = - \sum_{i=1}^{3} x_{ai} = 0.$$

Phase 1 has ended with a basic feasible solution. The resulting $\underline{B}$ matrix is the one that was initially assumed in Section A.2 to work the original example, i.e., Phase 2 of the simplex procedure.

APPENDIX B

SUBROUTINE LINPO

LINPO is a double-precision computer subroutine written in Fortran IV which uses the simplex method of Appendix A to solve the general linear-programming problem. LINPO is a mnemonic for LINear Programming Optimization. This subroutine was developed by modifying an existing routine [69] so that the internal calculations of the iterative procedure could be followed and the routine could be used to design digital filters. Instructions for using the subroutine, an example solution, a flow-chart, and a program listing are contained in this appendix.

B.1 USER INSTRUCTIONS

The call statement for the subroutine is CALL LINPO(A,B,IE,NEQ, NVA,NSP,NT) with the required input parameters described in Table B.1.

TABLE B.1

LINPO INPUT PARAMETERS

A:	A two-dimensional double-precision array of real numbers representing the objective-function and constraint-equation coefficients. Coefficients of the objective function multiplied by -1 must be placed in the first row with each of the remaining rows containing the coefficients for one constraint equation.
B:	A double-precision array of non-negative numbers representing the right-hand sides of the constraint equations and entered in the same order as the A array. The first value, which corresponds to the objective function, must be zero.

IE: An array of integer constants representing the type of inequality for each equation. The constant is one of the integers zero through three depending on the type, namely, objective function (IE = 0), less than or equal (IE =1), equal to (IE = 2), or greater than or equal (IE = 3). The integers are entered in the same order as the B array.

NEQ: Integer constant M + 1 equal to the number of constraint equations including the objective function.

NVA: Integer constant R equal to number of unknowns in the original constraint equations.

NSP: Integer constant specifying tableau output, i.e., tableau output (NSP = 1), or no tableau output (NSP = 0).

NT: Integer constant specifying type of digital filter to be designed, namely, frequency-sampling filter (NT = 0,1,2,3) or transversal filter (NT = 4,5,6,7). Furthermore, the LINPO subroutine can be used for general linear-programming applications other than the design of digital filters by setting NT $\geq$ 8.

B.2 EXAMPLE PROBLEM

The problem chosen to illustrate the use of LINPO is the one which has been worked by a graphical technique in Section 3.2 and by hand computation in Section A.2, i.e., to find the maximum value of Z where

$$Z = x_1 + x_2 ,$$

subject to the constraints

$$x_1 + x_2 \geq 2 ,$$

$$3x_1 + x_2 \leq 15 ,$$

and

$$x_1 + 2x_2 \leq 9 ,$$

will now be solved using subroutine LINPO. The main program required is illustrated in Fig. B-1 with the corresponding subroutine output given in Fig. B-2.

```
DOUBLE PRECISION A(27,50), B(4)
DIMENSION IE(4)
DATA A/-1.D0,1.D0,3.D0,1.D0,23*0.D0,-1.D0,1.D0,1.D0,2.D0,1319*0./
DATA B/0.D0,2.D0,15.D0,9.D0/
DATA IE/0,3,1,1/
DATA NEQ,NVA,NSP,NT/4,2,1,8/
CALL LINPO(A,B,IE,NEQ,NVA,NSP,NT)
STOP
END
```

Fig. B-1. Main Program for LINPO Solution.

Information is output in a tableau form, where the tableau entries are defined in Table B.2. Two rows of the tableau contain values for $z_j - c_j$. For Phase 1 of the simplex method, $z_j - c_j$ values in the last row are used to determine the vector $\underline{a}_k$ to enter the basis, during which time the first row values are not significant. Conversely, for Phase 2 the first row values are used and the last row has no significance. The current objective function value is the last entry in the last row for Phase 1 and the last entry in the first row for Phase 2. The components of each $\underline{y}_j$ and the basic solution components X_{Bi}, i = 1,2,...,M, are contained in the other M rows of the

TABLE B.2

TABLEAU FORMAT

A1		A5	XB
z_1-c_1	. . .	z_5-c_5	Z
y_{11}		y_{15}	x_{B1}
y_{21}		y_{25}	x_{B2}
y_{31}		y_{35}	x_{B3}
z_1-c_1	. . .	z_5-c_5	Z*

```
                                          DATA LOADED

                                       TABLEAU 0 PHASE 1

                VECTOR ENTERED A 0                                    VECTOR REMOVED 0 0

BASIS VECTORS         A 1                A 2                A 3                A 4                A 5                XB
     **         -0.100000000D 01   -0.100000000D 01    0.0                0.0                0.0                0.0
     0 0         0.100000000D 01    0.100000000D 01   -0.100000000D 01    0.0                0.0                0.200000000D 01
     A 4         0.300000000D 01    0.100000000D 01    0.0                0.100000000D 01    0.0                0.150000000D 02
     A 5         0.100000000D 01    0.200000000D 01    0.0                0.0                0.100000000D 01    0.900000000D 01
     **         -0.100000000D 01   -0.100000000D 01    0.100000000D 01    0.0                0.0               -0.200000000D 01

                                       TABLEAU 1 PHASE 1

                VECTOR ENTERED A 1                                    VECTOR REMOVED 0 0

BASIS VECTORS         A 1                A 2                A 3                A 4                A 5                XB
     **          0.0                0.0               -0.100000000D 01    0.0                0.0                0.200000000D 01
     A 1         0.100000000D 01    0.100000000D 01   -0.100000000D 01    0.0                0.0                0.200000000D 01
     A 4         0.0               -0.200000000D 01    0.300000000D 01    0.100000000D 01    0.0                0.900000000D 01
     A 5         0.0                0.100000000D 01    0.100000000D 01    0.0                0.100000000D 01    0.700000000D 01
     **          0.0                0.0                0.0                0.0                0.0                0.0

                                          FEASIBLE

                                       TABLEAU 1 PHASE 2

                VECTOR ENTERED A 3                                    VECTOR REMOVED A 4

BASIS VECTORS         A 1                A 2                A 3                A 4                A 5                XB
     **          0.0               -0.666666667D 00    0.0                0.333333333D 00    0.0                0.500000000D 01
     A 1         0.100000000D 01    0.333333333D 00    0.0                0.333333333D 00    0.0                0.500000000D 01
     A 3         0.0               -0.666666667D 00    0.100000000D 01    0.333333333D 00    0.0                0.300000000D 01
     A 5         0.0                0.166666667D 01    0.0               -0.333333333D 00    0.100000000D 01    0.400000000D 01
     **          0.0                0.0                0.0                0.0                0.0                0.0

                                       TABLEAU 2 PHASE 2

                VECTOR ENTERED A 2                                    VECTOR REMOVED A 5

BASIS VECTORS         A 1                A 2                A 3                A 4                A 5                XB
     **          0.0                0.0                0.0                0.200000000D 00    0.400000000D 00    0.660000000D 01
     A 1         0.100000000D 01    0.0                0.0                0.400000000D 00   -0.200000000D 00    0.420000000D 01
     A 3         0.0                0.0                0.100000000D 01    0.200000000D 00    0.400000000D 00    0.460000000D 01
     A 2         0.0                0.100000000D 01    0.0               -0.200000000D 00    0.600000000D 00    0.240000000D 01
     **          0.0                0.0                0.0                0.0                0.0                0.0

                              OPTIMUM Z        0.660000000D 01

                                       BASIC SOLUTION

                              VARIABLE         VALUE
                                  1        0.420000000D 01
                                  3        0.460000000D 01
                                  2        0.240000000D 01
```

Fig. B-2. Tableau Output of Subroutine LINPO.

tableau. Also printed after each iteration are the vectors currently in the basis and the vectors which were entered and removed to form that basis. However, the initial vector entered and removed statements have no significance. All artificial vectors are represented by QO.

Intermediate computer solutions contained in the tableau of Fig. B-2 agree with the results obtained by hand computation in Appendix A. for example the last iteration values in Section A.2 were

$$\underline{x}_B = (4.2,\ 4.6,\ 2.4)$$

$$\underline{y}_4 = (0.4,\ 0.2,\ -0.2)$$

$$\underline{y}_5 = (-0.2,\ 0.4,\ 0.6)\ ,$$

$$z_4 - c_4 = 0.2\ ,$$

$$z_5 - c_5 = 0.4\ ,$$

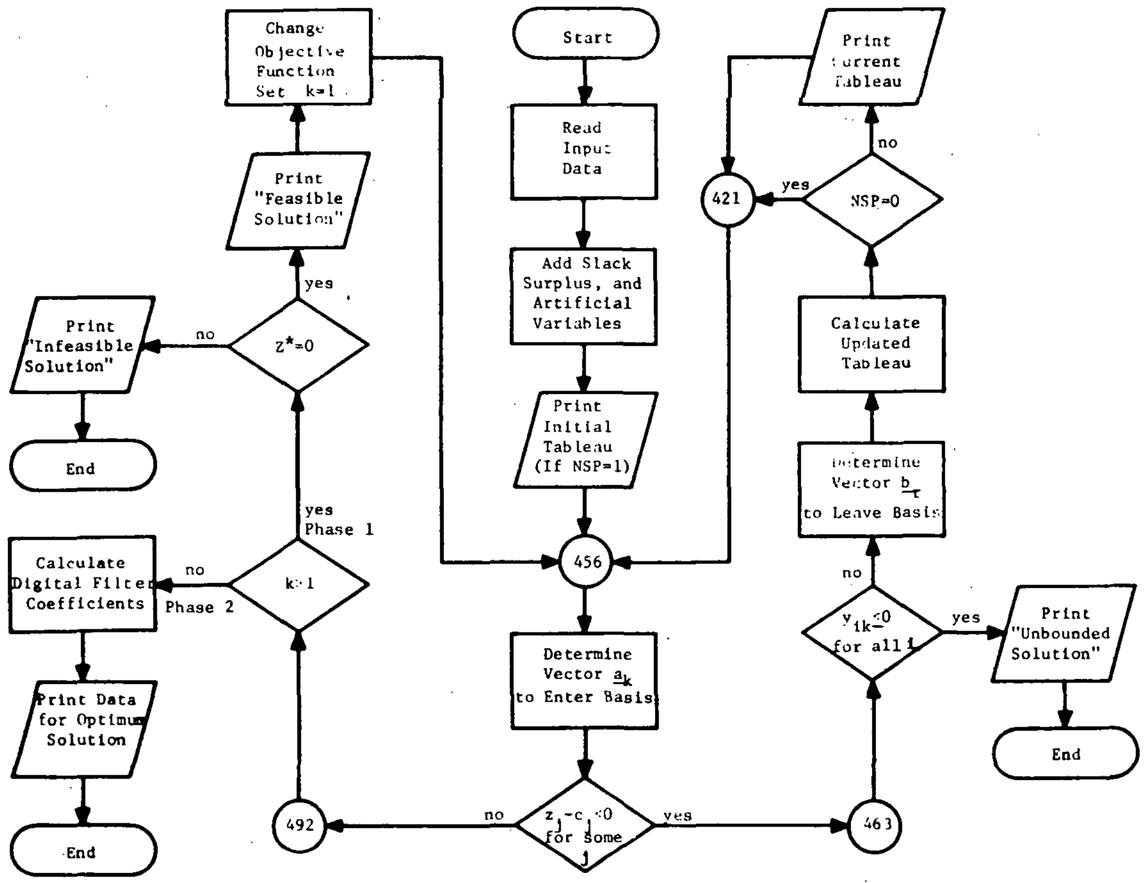
and

$$Z = 6.6$$

Each of these values can be found in the appropriate columns of Tableau 2 of Phase 2.

B.3 FLOWCHART AND PROGRAM LISTING

A flowchart and program listing for the LINPO subroutine are given in Figs. B-3 and B-4, respectively. The calculations (A.6) through (A.11), as discussed in Appendix A, make up the right-hand loop in the flow chart and each passage through the loop represents one iteration. The results of these calculations can be printed out at the end of each iteration in the tableau form. The left-hand loop determines if the solution at the end of Phase 1 is feasible and if so, sets up the problem for Phase 2. At the end of Phase 2 a branch of this loop calculates (from the optimal solution) the digital filter coefficient values H_0 and $(-1)^k 2H_k$, k = 1,2,...,NVA-1, for a frequency-sampling filter design or $(-1)^m H_m$, m = 0,1,...,NVA-1, for a transversal filter design. These coefficient values along with the optimal solution are then printed. Filter coefficients for a frequency-sampling and a transversal filter design example are printed in Figs. C-6 and C-7 respectively.



Start
Read Input Data
Add Slack Surplus, and Artificial Variables
Print Initial Tableau (If NSP=1)
456
Determine Vector a_k to Enter Basis
$z_j - c_j < 0$ for some j
yes
no
463
$y_{ik} \leq 0$ for all i
yes
Print "Unbounded Solution"
End
no
Determine Vector b_r to Leave Basis
Calculate Updated Tableau
NSP=0
no
Print Current Tableau
yes
421
492
k=1
yes Phase 1
no Phase 2
$z^*=0$
yes
Print "Feasible Solution"
Change Objective Function Set k=1
no
Print "Infeasible Solution"
End
Calculate Digital Filter Coefficients
Print Data for Optimum Solution
End

Fig. B-3. Flowchart for Subroutine LINPO.

```
      SUBROUTINE LINPO(A,B,IE,NEQ,NVA,NSP,NT)
C
C     THIS SUBROUTINE SOLVES THE GENERAL LINEAR-PROGRAMMING
C     PROBLEM BY THE SIMPLEX METHOD. THE SUBROUTINE ACCEPTS
C     UP TO 25 CONSTRAINT EQUATIONS AND 40 VARIABLES.
C
      DIMENSION L( 27),IE(NEQ)
      DOUBLE PRECISION A(27, 50),W( 27),X,XMIN,B(NEQ),H(40)
      COMMON /HNAME/ H
      INTEGER AOA(200),AOA1(200),AO,AN,BN,XN,BL,ON
      KJ11=0
      DATA  BL/' '/
      DATA XN/'X'/
      DATA AN/'A'/
      DATA BN/'B'/
      DATA ON/'O'/
      GO TO 422
C     WRITE TABLEAU HEADINGS AND VALUES (THRU STATEMENT 421)
400   IF (NSP.EQ.0) GO TO 421
      JJJ1=JJJ
      IF(JJJ.GT.6)  JJJ1=6
      KPH=1
      IF(K.EQ.1)   KPH=2
      IF(K.GT.1)   KPH=1
      WRITE(6,401)KKK,KPH
401   FORMAT('0',51X,'TABLEAU',I2,1X,'PHASE',I2)
      IF(KJ11.EQ.0) GO TO 402
      WRITE(6,403)L(JK),KJ11
      GO TO 405
402   WRITE(6,404) L(JK),KJ11
403   FORMAT('0',20X,'VECTOR ENTERED A',I2,41X,'VECTOR',
     1' REMOVED A',I2)
404   FORMAT('0',20X,'VECTOR ENTERED A',I2,41X,'VECTOR',
     1' REMOVED O',I2)
405   WRITE(6,406)
406   FORMAT('0',   'BASIS VECTORS')
      DO 407 I=1,JJJ
      AOA1(I)=BL
      AOA(I)=AN
      IF(I.EQ.JJJ)  AOA1(I)=XN
407   IF(I.EQ.JJJ) AOA(I)=BN
      IF(JJJ.LE.6) GO TO 408
      WRITE(6,409) (AOA1(I),AOA(I),I,I=1,JJJ1)
      GO TO 410
408   JJJJ=JJJ1-1
      WRITE(6,409) (AOA1(I),AOA(I),I,I=1,JJJJ ),
     1AOA1(JJJ1),AOA(JJJ1)
409   FORMAT('+',24X,6(2A1,I2,13X))
```

Fig. B-4. Program Listing for Subroutine LINPO.

```
410    CONTINUE
       JJ1=JJJ
       IF(JJJ.GT.6) JJ1=6
       DO 411  I=1,III
       LLL=BL
       AO=BL
       IF(I.GT.1) LLL=L(I)
       IF(I.GT.1) AO=AN
       IF(L(I).EQ.0.AND.I.GT.1) AO=ON
       IF(I.EQ.III) LLL=BL
       IF(I.EQ.III) AO=BL
411    WRITE(6,412) AO,LLL,(A(I,J),J=1,JJ1)
412    FORMAT(' ', 4X,A1,I2,11X,6(D15.8, 2X))
       K1K=2
       JJ11=6
       JJ1=JJ1+1
       JJJ1=JJJ-6
413    IF(JJJ1)421,421,414
414    JJ11=JJJ1+JJ11
       IF(JJJ1.GT.6)  JJ11=6*K1K
       IF(JJJ1.LE.6) GO TO 415
       WRITE(6,417) (AOA1(I),AOA(I),I,I=JJ1,JJ11)
       GO TO 418
415    JJ111=JJ11-1
       IF(JJ1.GT.JJ111) GO TO 416
       WRITE(6,417) (AOA1(I),AOA(I),I,I=JJ1,JJ111),
      1AOA1(JJ11),AOA(JJ11)
       GO TO 418
416    WRITE(6,417) AOA1(JJ11),AOA(JJ11)
417    FORMAT('0',23X,6(2A1,I2,13X))
418    DO 419 I=1,III
419    WRITE(6,420)(A(I,J),J=JJ1,JJ11)
420    FORMAT(' ',18X,6(D15.8,2X))
       JJJ1=JJJ1-6
       JJ1=JJ11+1
       K1K=K1K+1
       GO TO 413
421    CONTINUE
       IF(K) 457,456,457
422    CONTINUE
423    FORMAT(47X,I4,D20.8)
424    FORMAT('0',56X,'FEASIBLE')
425    FORMAT('0',56X,'INFEASIBLE')
426    FORMAT(46X,'VARIABLE',7X,'VALUE')
427    FORMAT(1X,//,45X,9HOPTIMUM Z,D20.8//)
428    FORMAT(53X,'BASIC SOLUTION'/)
429    FORMAT(1X,55X,'UNBOUNDED'///)
430    FORMAT(1H ,'ROWS EXCEED ALLOWABLE MAXIMUM OF 25',
```

Fig. B-4 (Continued). Program Listing for Subroutine LINPO.

```
     1' EQUATIONS'//'PROCESSING TERMINATED')
431   FORMAT(1H ,'COLUMNS EXCEED ALLOWABLE MAXIMUM OF',
     1' 40 VARIABLES'//'PROCESSING TERMINATED')
C     READ INPUT DATA (THRU STATEMENT 447)
      II=NEQ
      JJ=NVA+1
432   IF(II-25)434,434,433
433   WRITE(6,430)
      GO TO 507
434   IF(JJ- 41)436,436,435
435   WRITE(6,431)
      GO TO 507
436   III=II+1
      JJJ=JJ+II
      DO 437 I=1,III
      W(I)=0.0
      L(I)=0
      DO 437 J=JJ,JJJ
437   A(I,J)=0.0
      DO 438 J=1,NVA
438   A(III,J)=0.0
      JJJ=JJ
      DO 445 I=1,II
      JK=IE(I)
439   IF(JK)507,445,440
440   IF(JK-3)441,441,507
441   JK=JK-2
      IF(JK)442,444,443
442   A(I,JJJ)=1.
      L(I)=JJJ
      JJJ=JJJ+1
      GO TO 445
443   A(I,JJJ)=-1.
      JJJ=JJJ+1
444   L(I)=0
445   CONTINUE
      DO 446 I=1,II
446   A(I,JJJ)=B(I)
      JJ=JJJ
C     WRITE DATA LOADED
      WRITE(6,447)
447   FORMAT('1',54X,'DATA LOADED')
C     START OF SOLUTION PHASE
448   KKK=0
      JK=1
      K=0
C     ADD ARTIFICIAL VARIABLES TO PROBLEM (THRU STATEMENT 455)
449   I=1
```

Fig. B-4 (Continued). Program Listing for Subroutine LINPO.

```
450    I=I+1
       IF(I-III)451,455,455
451    IF(L(I))450,452,450
452    DO 454 J=1,JJ
       IF(A(I,J))453,454,453
453    A(III,J)=A(III,J)-A(I,J)
454    CONTINUE
       GO TO 450
455    GO TO 400
456    K=III
C      DETERMINE VECTOR TO ENTER BASIS (THRU STATEMENT 463)
457    J=0
       W(K)=0.0
       L(K)=0
458    J=J+1
       IF(J-JJ)459,462,462
459    IF(DABS(A(K,J)).LE.0.1D-08) A(K,J)=0.
       IF(A(K,J))460,458,458
460    IF(W(K)-A(K,J))458,458,461
461    W(K)=A(K,J)
       L(K)=J
       GO TO 458
462    IF(L(K))463,492,463
463    KJ=L(K)
C      DETERMINE IF SOLUTION IS UNBOUNDED
       DO 464 I=2,II
       IF(A(I,KJ))464,464,465
464    CONTINUE
C      WRITE UNBOUNDED
       WRITE(6,429)
       GO TO 507
C      DETERMINE VECTOR TO LEAVE BASIS (THRU STATEMENT 473)
465    I=1
       JK=0
466    I=I+1
       IF(I-II)467,467,473
467    IF(A(I,KJ))466,466,468
468    IF(DABS(A(I,KJ))-0.1D-08)469,469,470
469    A(I,KJ)=0.
       GO TO 466
470    X=A(I,JJ)/A(I,KJ)
       IF(JK)471,472,471
471    IF(X-XMIN)472,466,466
472    XMIN=X
       JK=I
       GO TO 466
473    X=A(JK,KJ)
       KJ11=L(JK)
```

Fig. B-4 (Continued). Program Listing for Subroutine LINPO.

```
      L(JK)=KJ
C     CALCULATE NEW TABLEAU VALUES (THRU STATEMENT 491)
      DO 474 I=1,III
474   W(I)=A(I,KJ)
      IJ=JK-1
      DO 482 I=1,IJ
      DO 482 J=1,JJ
      IF(A(JK,J))475,482,475
475   IF(DABS(A(JK,J))-0.1D-08)476,476,477
476   A(JK,J)=0.
      GO TO 482
477   IF(W(I))478,482,478
478   IF(DABS(W(I))-0.1D-08)479,479,480
479   W(I)=0.
      GO TO 482
480   A(I,J)=A(I,J)-W(I)*(A(JK,J)/X)
      IF(DABS(A(I,J))-0.1D-08)481,481,482
481   A(I,J)=0.
482   CONTINUE
      IJ=JK+1
      DO 490 I=IJ,III
      DO 490 J=1,JJ
      IF(A(JK,J))483,490,483
483   IF(DABS(A(JK,J))-0.1D-08)484,484,485
484   A(JK,J)=0.
      GO TO 490
485   IF(W(I))486,490,486
486   IF(DABS(W(I))-0.1D-08)487,487,488
487   W(I)=0.
      GO TO 490
488   A(I,J)=A(I,J)-W(I)*(A(JK,J)/X)
      IF(DABS(A(I,J))-0.1D-08)489,489,490
489   A(I,J)=0.
490   CONTINUE
      DO 491 J=1,JJ
491   A(JK,J)=A(JK,J)/X
      KKK=KKK+1
      GO TO 400
492   IF(K-1)497,497,493
C     DETERMINE IF SOLUTION IS FEASIBLE
493   IJ=JJ-1
      IF (DABS(A(K,JJ))-0.1D-08)494,494,496
494   CONTINUE
C     WRITE FEASIBLE
      WRITE (6,424)
      DO 495 J=1,JJ
495   A(III,J)=0.0
      K=1
```

Fig. B-4 (Continued). Program Listing for Subroutine LINPO.

```
      KKK=0
      GO TO 457
C     WRITE INFEASIBLE
496   WRITE ( 6,425)
      GO TO 507
C     WRITE OPTIMUM VALUE OF OBJECTIVE FUNCTION
497   WRITE (6,427) A(1,JJ)
      WRITE (6,428)
      WRITE (6,426)
C     WRITE LIST OF REAL VARIABLES
      DO 498 I=2,II
498   WRITE (6,423) L(I),A(I,JJ)
      IF (NT.GE.8) GO TO 507
      WRITE(6,499)
499   FORMAT(///,47X,'DIGITAL FILTER COEFFICIENTS')
      WRITE(6,500)
500   FORMAT('0',47X,'STAGE',8X,'VALUE')
C     CALCULATE DIGITAL FILTER COEFFICIENTS (THRU STATEMENT 504)
      DO 502 I=1,NVA
      DO 501 J=2,II
      IF(L(J).EQ.I) H(I)=A(J,JJ)
      IF(L(J).EQ.I) GO TO 502
      IF(J.EQ.II) H(I)=0.0D0
501   CONTINUE
502   CONTINUE
      DO 503 I=2,NVA,2
503   H(I)=(-1.D0)*(H(I))
      IF (NT.GE.4) GO TO 505
      DO 504 I=2,NVA
504   H(I)=(2.0D0)*(H(I))
C     WRITE LIST OF FILTER COEFFICIENTS
505   DO 506 I=1,NVA
506   WRITE(6,423)I,H(I)
507   RETURN
      END
```

Fig. B-4 (Continued). Program Listing for Subroutine LINPO.

APPENDIX C

AUXILIARY SUBPROGRAMS

The linear-programming subroutine LINPO used for the design of digital filters requires that coefficient arrays be calculated before the subroutine can be used. Since this procedure can become rather tedious due to the number and type of calculations required, especially for the channel equalization application, auxiliary Fortran IV subprograms COCAL1 and COCAL2 were developed, where COCAL is a mneomonic for COefficient CALculator. COCAL1 is a subroutine for the design of frequency-sampling filters which are to be used either for generating pulses to transmit data over ideal channels or for matched filter detection. COCAL2 is a subroutine for the design of either transversal or frequency-sampling filters which are to be used to equalize nonideal channels. The nonideal channel must be defined by its unit-impulse response in a function subprogram. Consequently, a number of impulse-response equations corresponding to common data transmission channels have been programmed and are incorporated in a function subprogram called CC. Also included is a subroutine CHECK to be used in conjunction with COCAL2 and CC. It calculates the unit-impulse response of the equalized channel after the design of the digital-filter equalizer.

C.1 SUBROUTINE COCAL1

The frequency-sampling filter impulse-response envelope was derived in Section 3.1 and expressed in (3.6) or (3.7) as

$$h(t) = \frac{H_0}{N} + \sum_{k=1}^{R-1} (-1)^k \frac{2H_k}{N} \cos(2\pi kt/NT) , \qquad (C.1)$$

where the upper limit on the sum has been replaced by R-1 and the frequency sample $H_{N/2}$ has been assumed to be zero if N is even. The derivative of (C.1) is

$$h'(t) = \frac{4\pi}{N^2T} \sum_{k=1}^{R-1} k(-1)^{k+1} H_k \sin(2\pi kt/NT) . \qquad (C.2)$$

Coefficients of the H_k terms must be calculated for the objective function and every constraint equation of any LINPO frequency-sampling filter design. Also, for any LINPO design the constraint equations must be expressed in a form such that all b_i as defined in (3.11) are nonnegative. Furthermore, the objective function must be expressed as a maximization problem. COCAL1 is a double-precision computer subroutine that takes a more general linear-programming design problem and sets up the proper LINPO input arrays to satisfy these restrictions. It also calls the LINPO subroutine to solve the problem.

There are a variety of problems that COCAL1 will accept. The objective function can be in any of the following forms:

$$\text{maximize } Z = \sum_j \alpha_j\, h(t_j) , \qquad (C.3a)$$

$$\text{maximize } Z = \sum_j \alpha_j h'(t_j) , \qquad \text{(C.3b)}$$

$$\text{minimize } Z = \sum_j \alpha_j h(t_j) , \qquad \text{(C.3c)}$$

or

$$\text{minimize } Z = \sum_j \alpha_j h'(t_j) , \qquad \text{(C.3d)}$$

where each α_j is either ±1. As many equations of one type as desired can be combined to form Z. Each individual constraint equation can be of the h(t) or h'(t) type with either positive, negative, or zero b_i.

C.1.1 User Instructions

The call statement for the subroutine is CALL COCAL1(CT,NTYPE,B, IE,NOBJ,N,T,NEQ,NVA,NSP,NT) where the input parameters IE, NEQ, NVA, and NSP were defined previously in Table B.1. The other input parameters are defined in Table C.1. The parameter NVA was defined as the number of unknowns in the constraint equations and equaled R in the constraint equation formulation of Appendix A. For the low-pass and band-pass filter design problems of Chapters 3 and 4, R and consequently NVA, is equal to the number of unknown multiplier coefficients to be determined optimally. The value for NT is generated internally in COCAL1 and is an output parameter.

TABLE C.1

COCAL1 INPUT PARAMETERS

CT:	A double-precision array of real numbers representing critical time points. All time points at which the impulse response is to be constrained must be included. Furthermore, if more than one constraint equation is written at the same critical time, that time point should be repeated for each of these constraint equations. However, time points that appear in the objective function equation should not be repeated if constraint equations have also been written at those points, as will normally be the case. If constraint equations have not been written at some of the objective function time points, those times should be included as the last members of the CT array.
NTYPE:	An array of integer constants representing the objective function and constraint equation types. The first value in the array represents the type of objective function, i.e., "0" for an objective function in the form of (C.3a), "1" for (C.3b), "2" for (C.3c) and "3" for (C.3d). The remaining values in the array are in one-to-one correspondence with the constraint equation time points of CT and represent the type of each constraint equation, i.e., "0" for an h(t) equation and "1" for an h'(t) equation.
B:	A double-precision array of positive or negative real numbers representing the right-hand sides of the constraint equations and entered in the same order as the NTYPE array. The first value, which corresponds to the objective function, must be zero.
NOBJ:	An array of positive and negative integer constants defining the objective function. The first value is a positive integer equal to the number of equations to be combined to form the objective function. The signs of the remaining integers define the α_j of (C.3) while the associated absolute values define the locations in the CT array of the corresponding t_j.
N:	A double-precision constant equal to the number of impulse response values.
T:	A double-precision constant equal to the sampling interval.

C.1.2 Example Problem

The set of constraint equations that were defined in (3.16) for the raised-cosine design problem were

$$h(3t_b) = 1 \qquad h(5t_b) = 0 \qquad h'(4t_b) \leq 0$$

$$h(4t_b) = 0 \qquad h(6t_b) = 0 \qquad h'(5t_b) \leq 0$$

with the objective function

$$Z = h'(4t_b) + h'(5t_b)$$

to be maximized. A main program is given in Fig. C-1 as an example of the input required for a COCAL1 solution of this problem. The corresponding output of the LINPO subroutine is illustrated in Fig. C-2. Since NSP = 0 for this example, none of the tableaus were printed during the linear-programming solution. The results printed first are the optimum value of Z and the optimal basic solution. The six values of the basic solution represent the unknown H_k, k = 0,...,5. These values were used to calculate the digital filter coefficients H_0 and $2(-1)^k H_k$, k = 1,...,5 which are printed out last.

```
      DIMENSION NTYPE(7),IE(7),NOBJ(3)
      REAL*8 CT(6),B(7),N,T
      DATA CT/3.0D0,4.0D0,5.0D0,6.0D0,4.0D0,5.0D0/
      DATA NTYPE/1,0,0,0,0,1,1/
      DATA B/0.0D0,1.0D0,5*0.0D0/
      DATA IE/0,2,2,2,2,1,1,/
      DATA NOBJ/2,2,3/
      DATA N,T,NEQ,NVA,NSP/60.0D0,0.1D0,7,6,0/
      CALL COCAL1(CT,NTYPE,B,IE,NOBJ,N,T,NEQ,NVA,NSP,NT)
      STOP
      END
```

Fig. C-1. Main Program for COCAL1 Solution.

```
                    DATA LOADED

                      FEASIBLE


         OPTIMUM Z        0.0


                  BASIC SOLUTION

          VARIABLE           VALUE
               5        0.33333336D 01
               1        0.10000000D 02
               4        0.50000000D 01
               3        0.66666664D 01
               6        0.16666673D 01
               2        0.83333327D 01


           DIGITAL FILTER COEFFICIENTS

             STAGE           VALUE
               1        0.10000000D 02
               2       -0.16666665D 02
               3        0.13333333D 02
               4       -0.10000000D 02
               5        0.66666671D 01
               6       -0.33333345D 01
```

Fig. C-2. LINPO Output for Raised-Cosine Filter Design.

C.1.3 Flowchart and Program Listing

A flowchart and program listing for the COCAL1 subroutine are given in Figs. C-3 and C-4. The left-hand half of the flowchart consists of two loops which calculate coefficients required for the LINPO A array. The outer loop (thru statement 210) runs from 1 to NEQ with the first execution corresponding to the objective function and the remaining NEQ-1 executions corresponding to the constraint equations. For each execution of the outer loop, there are NVA executions of the inner loop (thru statement 208) to calculate the NVA coefficients of each equation. That set of NVA coefficients forms one row of the A array. Since the objective function will normally be a combination of equations, the appropriate coefficients of the equations that make up the objective function must be combined during the first passage through the outer loop. This combining of coefficients is controlled by the NOBJ array.

The right-hand half of the flowchart represents three operations. First, the coefficients in the first row of the A array are multiplied by -1 if the objective function is of the type (C.3c) or (C.3d). As stated in Table B.1, the objective-function coefficients must be multiplied by -1 before a linear-programming solution is obtained, assuming that the objective-function value Z is to be maximized, which always occurs internally in the linear-programming subroutine. However, if the coefficient signs are not changed the value -Z is maximized and is actually a minimization of Z. Consequently, if Z is to be minimized the signs are not changed and by default a minimization problem is changed to a maximization problem. Next, the constraint equations must be checked to see if the right-hand sides are negative. If they are, both sides of the equation are multiplied by -1 and the inequality changed. This results in all non-negative numbers in its B array. Finally, LINPO is called to solve the problem which has now been properly formulated.

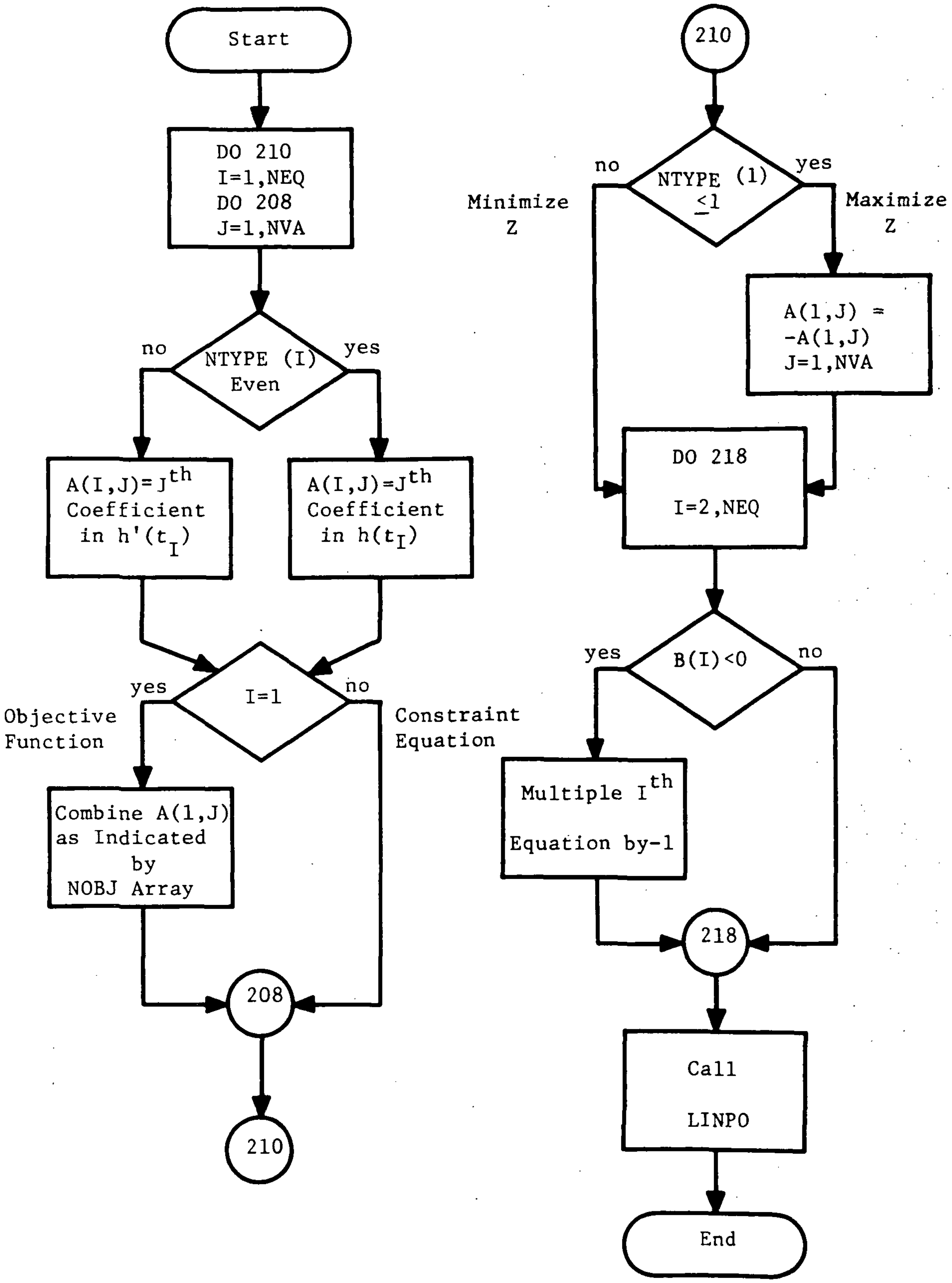


Fig. C-3. Flowchart for Subroutine COCAL1.

```
      SUBROUTINE COCAL1(T,NTYPE,B2,IE,NOBJ,M,T1,NEQ,NVA,NSP,NT)
C
C     THIS SUBROUTINE CALCULATES AND SETS UP ARRAYS REQUIRED
C     FOR THE LINPO DESIGN OF FREQUENCY-SAMPLING FILTERS.
C
      DIMENSION NTYPE(NEQ),IE(NEQ),NOBJ(1)
      REAL*8 SUM,B1,B2(NEQ),DCOS,PI,T(1),A(27,50),M,T1,
     1K,TT,DSIN
      PI=3.1415926D0
C     CALCULATE COEFFICIENTS FOR ARRAY A (THRU STATEMENT 210)
      DO 210 I=1,NEQ
      NTYP=NTYPE(I)+1
      GO TO (200,201,200,201),NTYP
200   NT=0
      GO TO 202
201   NT=1
202   K=0.0D0
      ISGN=1
      III=1
      IF (I.EQ.1) III=NOBJ(1)
C     CALCULATE COEFFICIENTS FOR ROW I OF ARRAY A
C     (THRU STATEMENT 208)
      DO 208 J=1,NVA
      SUM=0.0D0
      DO 206 II=1,III
      JJ=I-1
      IF (I.GT.1) GO TO 203
      JJ=II+1
      NBS=IABS(NOBJ(JJ))
      ISGN=ISIGN(1,NOBJ(JJ))
      JJ=NBS
203   TT=T(JJ)/T1
      B1=(2.0D0*PI*K)/M
      IF(NT -0)204,204,205
204   SUM=SUM+DCOS(B1*TT)*ISGN
      GO TO 206
205   SUM=SUM-B1*DSIN(B1*TT)*ISGN
206   CONTINUE
      A(I,J)=SUM*(2.0D0/M)
      IF(K.EQ.0.0D0) A(I,J)=A(I,J)/2.0D0
      K=K+1.0D0
208   CONTINUE
      DO 209 J=2,NVA,2
209   A(I,J)=-A(I,J)
210   CONTINUE
C     CONVERT MINIMIZATION PROBLEM TO A MAXIMIZATION PROBLEM
C     (THRU STATEMENT 212)
      NTYP=NTYPE(1)+1
```

Fig. C-4. Program Listing for Subroutine COCAL1.

```
      GO TO  (211,211,213,213),NTYP
211   DO 212 J=1,NVA
      A(1,J)=-A(1,J)
212   CONTINUE
C     CONVERT CONSTRAINT EQUATIONS IF RIGHT-HANDED SIDES ARE
C     NEGATIVE (THRU STATEMENT 218)
213   DO 218 I=2,NEQ
      IF(B2(I))214,218,218
214   B2(I)=-B2(I)
      DO 215 J=1,NVA
215   A(I,J)=-A(I,J)
      IF(IE(I)-2)216,218,217
216   IE(I)=3
      GO TO 218
217   IE(I)=1
218   CONTINUE
      CALL LINPO(A,B2,IE,NEQ,NVA,NSP,NT)
      RETURN
      END
```

Fig. C-4 (Continued). Program Listing for Subroutine COCAL1.

C.2 SUBROUTINE COCAL2

COCAL2 is a double-precision computer subroutine that formulates an equalizer design problem for a linear-programming solution in the same manner that COCAL1 structures a frequency-sampling filter design problem. The expression which was derived in Section 4.1 for the output of a data channel equalized with a frequency-sampling filter was

$$y(t) \doteq \frac{H_0}{N} \sum_{m=0}^{L} c(t - mT) + \sum_{m=0}^{L} \sum_{k=1}^{R-1} (-1)^k \frac{2H_k}{N} \cos(2\pi km/N)\, c(t - mT) \tag{C.4}$$

and similarly for a transversal equalizer

$$y(t) = \sum_{m=0}^{L} (-1)^m H_m\, c(t - mT) . \tag{C.5}$$

It was also shown that the c(t) values must be obtained from the actual channel impulse response by

$$c(t) = \begin{cases} \int_0^t c_c(\tau)\, d\tau & 0 < t \leq T \\ \int_{t-T}^t c_c(\tau)\, d\tau & T < t < \infty , \end{cases} \tag{C.6}$$

and that the derivatives of (C.4) and (C.5) are formed by replacing the appropriate $c'(t - mT)$ terms with

$$c'(t) = \begin{cases} c_c(t) & 0 < t \leq T \\ c_c(t) - c_c(t - T) & t > T . \end{cases} \tag{C.7}$$

During a LINPO design of either type of equalizer, the coefficients of the H_k or H_m terms must be calculated from (C.4) or (C.5) along with (C.6) and (C.7). In addition, other LINPO input arrays must be generated.

C.2.1 User Instructions

The call statement for the subroutine is CALL COCAL2(CT,NTYPE,B, IE,NOBJ,N,T,NEQ,NVA,NSP,NT,CC) where, with the following exceptions, the input parameters have been defined previously in Tables B.1 and C.1. Acceptable forms of the objective function are still defined by (C.3), with $h(t_j)$ replaced by $y(t_j)$. The possible values in the array NTYPE must be expanded to include both frequency-sampling and transversal filters. Values of NTYPE from 0 to 3, which previously identified the type of objective function and constraint equations for frequency-sampling pulse-shaping filters, now apply to frequency-sampling digital filter equalizers with the h(t) in Table C.1 replaced by y(t). NTYPE values from 4 through 7 identify transversal-equalizer designs and are correspondingly defined if each integer in the NTYPE description is increased by 4, e.g., 0 by 4, 1 by 5, etc. As an example, if the objective function for a transversal equalizer design is in the form of (C.3c), i.e., the minimization of a combination of $y(t_j)$ equations, the first value in the array would be NTYPE = 6.

Unlike frequency-sampling filter designs, the number of unknown multiplier coefficients for a transversal design is equal to the number of impulse-response values. Hence, NVA must be set equal to N for LINPO designs of transversal filters.

The parameter CC is not an input value but represents an external function that is called by COCAL2. Consequently, a function subprogram CC, which will be described in Section C.4, must be used with COCAL2.

C.2.2 Example Problem

An equalizer design problem introduced in Section 4.2.1 was to determine both frequency-sampling and transversal filters to equalize the channel $c_c(t) = 4\pi^2 t\ e^{-2\pi t}\ u(t)$ to meet the set of constraints specified by (4.16) and the objective function of (4.17). Filters with a sampling interval of 0.1 seconds and 25 impulse response values were specified. For the frequency-sampling filter, 12 multiplier coefficients were specified and from the impulse-response specification, 25 coefficients were allowable for the transversal filter. The input program required for a COCAL2 solution of this problem is illustrated in Fig. C-5.

Only one main program was required for this example since the set of specifications on the desired pulse shape were the same for both equalizers. COCAL2 is called twice, first for a frequency-sampling design and next for a transversal design. The only input-data difference between the two runs are the set of values for NTYPE and the value of NVA. Subroutine CHECK which is also called for each run and its associated input parameters will be described in Section C.3. The statement NN=2 specifies which channel model in subprogram CC is being equalized. LINPO outputs for the two runs are illustrated in Figs. C-6 and C-7. As indicated, the frequency-sampling filter requires 11 multipliers while the transversal filter only requires three.

```
      DIMENSION NTYPE(13),IE(13),NOBJ(5)
      REAL*8 CT(12),B(13),N,T,TIN,TM,DT
      COMMON NN
      EXTERNAL CC
      DATA CT/0.5DO,0.75DO,1.DO,1.5DO,2.DO,2.25DO,2.5DO,3.DO
     1,0.5DO,1.DO,2.DO,2.5DO/
      DATA NTYPE/1,8*0,4*1/
      DATA B/4*0.DO,1.DO,8*0.DO/
      DATA IE/0,8*2,2*3,2*1/
      DATA NOBJ/4,-1,-3,5,7/
      DATA N,T,NEQ,NVA,NSP/25.DO,0.1DO,13,12,0/
      NN=2
      DATA TIN,TM,DT/0.DO,3.DO,0.05DO/
      CALL COCAL2(CT,NTYPE,B,IE,NOBJ,N,T,NEQ,NVA,NSP,NT,CC)
      CALL CHECK(TIN,TM,DT,N,T,NVA,NT,CC)
      DO 1 I=1,13
    1 NTYPE(I)=NTYPE(I)+4
      NVA=25
      CALL COCAL2(CT,NTYPE,B,IE,NOBJ,N,T,NEQ,NVA,NSP,NT,CC)
      CALL CHECK(TIN,TM,DT,N,T,NVA,NT,CC)
      STOP
      END
```

Fig. C-5. Main Program for COCAL2 Solution.

C.2.3 Flowchart and Program Listing

Since COCAL2 performs the same functions using y(t) specifications as COCAL1 does with h(t) specifications, the flowchart of Fig. C-3 is still valid for COCAL2, with the exception that the flowchart statement numbers 208, 210, and 218 are now 316, 322, and 330, respectively. Also, coefficients in $y(t_I)$ and $y'(t_I)$ are calculated instead of $h'(t_I)$ and $h(t_I)$. The COCAL2 program listing is furnished in Fig. C-8.

```
                DATA LOADED

                 FEASIBLE


      OPTIMUM Z        0.0


              BASIC SOLUTION

      VARIABLE          VALUE
          1        0.17557522D 01
          2        0.21980829D 01
          3        0.38701008D 01
          5        0.89141884D 01
          4        0.62382957D 01
          7        0.93451122D 01
          8        0.88793843D 01
          9        0.82047071D 01
         10        0.72433066D 01
          6        0.98221871D 01
         11        0.36646189D 01
         16        0.0


       DIGITAL FILTER COEFFICIENTS

        STAGE           VALUE
          1        0.17557522D 01
          2       -0.43961657D 01
          3        0.77402016D 01
          4       -0.12476591D 02
          5        0.17828377D 02
          6       -0.19644374D 02
          7        0.18690224D 02
          8       -0.17758769D 02
          9        0.16409414D 02
         10       -0.14486613D 02
         11        0.73292379D 01
         12        0.0
```

Fig. C-6. LINPO Output for Frequency-Sampling Equalizer Design.

```
                  DATA LOADFD

                   FEASIBLF


          OPTIMUM Z       0.0


                BASIC SOLUTION

           VARIABLE          VALUE
                5       0.0
                4       0.0
                9       0.0
               23       0.0
                8       0.0
               13       0.48966586D 01
               19       0.56444058D 00
                0       0.0
               18       0.12696229D 01
                7       0.0
               22       0.0
               25       0.0



             DIGITAL FILTER COEFFICIENTS

             STAGE           VALUE
                1       0.0
                2       0.0
                3       0.0
                4       0.0
                5       0.0
                6       0.0
                7       0.0
                8       0.0
                9       0.0
               10       0.0
               11       0.0
               12       0.0
               13       0.48966586D 01
               14       0.0
               15       0.0
               16       0.0
               17       0.0
               18      -0.12696229D 01
               19       0.56444058D 00
               20       0.0
               21       0.0
               22       0.0
               23       0.0
               24       0.0
               25       0.0
```

Fig. C-7. LINPO Output for Transversal Equalizer Design.

```
      SUBROUTINE COCAL2(T,NTYPE,B2,IE,NOBJ,M,T1,NEQ,NVA,NSP,NT
     1,CC)
C
C     THIS SUBROUTINE CALCULATES AND SETS UP ARRAYS REQUIRED
C     FOR THE LINPO DESIGN OF FREQUENCY-SAMPLING OR
C     TRANSVERSAL EQUALIZERS.
C
      DIMENSION NTYPE(NEQ),IE(NEQ),NOBJ(1)
      REAL*8 SUM,B2(NEQ),CC,DCOS,PI,D,T(1),A(27,50),Y,M,
     1T1,K,N,X,Z
      REAL*8 A1(25,40)
      EXTERNAL CC
      PI=3.141592600
C     CALCULATE COEFFICIENTS FOR ARRAY A (THRU STATEMENT 322)
      DO 322 I=1,NEQ
      NTYP=NTYPE(I)+1
      GO TO (300,301,300,301,302,303,302,303),NTYP
300   NT=0
      GO TO 304
301   NT=1
      GO TO 304
302   NT=4
      GO TO 304
303   NT=5
304   K=0.0D0
      ISGN=1
      III=1
      IF (I.EQ.1) III=NOBJ(1)
      NVA1=NVA
      IF(NT.GE.4) NVA1=1
C     CALCULATE COEFFICIENTS OF ROW I OF ARRAY A
C     (THRU STATEMENT 316)
      DO 316 J=1,NVA1
      SUM=0.0D0
      DO 313 II=1,III
      JJ=I-1
      IF (I.GT.1) GO TO 305
      JJ=II+1
      NBS=IABS(NOBJ(JJ))
      ISGN=ISIGN(1,NOBJ(JJ))
      JJ=NBS
305   L=T(JJ)/T1+1
      TTL=FLOAT(L)
      IF (DBLE(TTL).GT.M) L=M
      N=0.0D0
      DO 311 IJ=1,L
      X=T(JJ)-N*T1
      Z=X-T1
```

Fig. C-8. Program Listing for Subroutine COCAL2.

```
      IF (Z) 306,306,307
306   IF(NT.EQ.0.OR.NT.EQ.4) CALL DQG16(0.0D0,X,CC,Y)
      IF(NT.EQ.1.OR.NT.EQ.5) Y=CC(X)
      GO TO 308
307   IF(NT.EQ.0.OR.NT.EQ.4) CALL DQG16(Z,X,CC,Y)
      IF(NT.EQ.1.OR.NT.EQ.5) Y=CC(X)-CC(Z)
308   IF(NT.GE.4) GO TO 309
      D=(2.0D0*PI*K*N)/M
      SUM=ISGN*Y*DCOS(D)+SUM
      GO TO 310
309   A1(II,IJ)=Y*ISGN
310   N=N+1.0D0
311   CONTINUE
      IF(NT.LT.4.OR.L.EQ.NVA) GO TO 313
      LL=L+1
      DO 312 JI=LL,NVA
312   A1(II,JI)=0.0D0
313   CONTINUE
      IF(NT.GE.4) GO TO 316
      A(I,J)=SUM*(2.0D0/M)
314   IF(K.EQ.0.0D0) A(I,J)=A(I,J)/2.0D0
315   K=K+1.0D0
316   CONTINUE
      IF (NT.LT.4) GO TO 320
317   DO 319 IJ=1,NVA
      SUM=0.0D0
      DO 318 II=1,III
318   SUM=SUM+A1(II,IJ)
319   A(I,IJ)=SUM
320   CONTINUE
      DO 321 J=2,NVA,2
321   A(I,J)=-A(I,J)
322   CONTINUE
C     CONVERT MINIMIZATION PROBLEM TO A MAXIMIZATION PROBLEM
C     (THRU STATEMENT 324)
      NTYP=NTYPE(1)+1
      GO TO(323,323,325,325,323,323,325,325),NTYP
323   DO 324 J=1,NVA
      A(1,J)=-A(1,J)
324   CONTINUE
C     CONVERT CONSTRAINT EQUATIONS IF RIGHT-HANDED SIDES ARE
C     NEGATIVE (THRU STATEMENT 330)
325   DO 330 I=2,NEQ
      IF(B2(I))326,330,330
326   B2(I)=-B2(I)
      DO 327 J=1,NVA
327   A(I,J)=-A(I,J)
      IF(IE(I)-2)328,330,329
```

Fig. C-8 (Continued). Program Listing for Subroutine COCAL2.

```
328   IE(I)=3
      GO TO 330
329   IE(I)=1
330   CONTINUE
      CALL LINPO(A,B2,IE,NEQ,NVA,NSP,NT)
      RETURN
      END
```

Fig. C-8 (Continued). Program Listing for Subroutine COCAL2.

C.3 SUBROUTINE CHECK

Equations (C.4) and (C.5) represent the received pulse at the output of a channel equalized with a frequency-sampling filter and transversal filter respectively. In order to verify the results of the LINPO equalizer designs, these equations along with (C.6) were programmed to form the double-precision computer subroutine called CHECK which calculates and generates plots of y(t).

The call statement for the subroutine is CALL CHECK(TIN,TM,DT, N,T,NVA,NT,CC) where the input parameters TIN, TM, and DT are defined in Table C.2 and the other parameters have been defined previously.

TABLE C.2

CHECK INPUT PARAMETERS

TIN:	A double-precision constant equal to the initial time point at which y(t) is desired to be evaluated.
TM:	A double-precision constant equal to the maximum time point at which y(t) is desired to be evaluated.
DT:	A double-precision constant equal to the time interval desired between consecutive evaluations of y(t).

Filter coefficients H_k or H_m need not be externally entered into the subroutine since these values are placed in a labeled COMMON block by LINPO at the completion of the design and are therefore available to CHECK. The integer constant NT is required by CHECK to determine which equalizer has been designed, but is generated by COCAL2 and does not have to be stated in the main program. An example of the use of the CHECK subroutine in a main program is illustrated in Fig. C-5 with the output plots generated by the subroutine given in Figs. 4-3(b) and 4-4(b).

A flowchart and program listing for the subroutine are presented in Figs. C-9 and C-10 respectively. The primary loop of the program is executed for each of the NDATA time points at which the function y(t) is to be evaluated. During each execution, channel impulse-response values obtained by integrating $c_c(\tau)$ from the function subprogram CC are combined with either the H_k or H_m values from LINPO, depending on the values of NT, to form $y(t_I)$. After all NDATA values of $y(t_I)$ have been calculated, GRAPH1 [52] is called to plot $y(t_I)$ versus t_I.

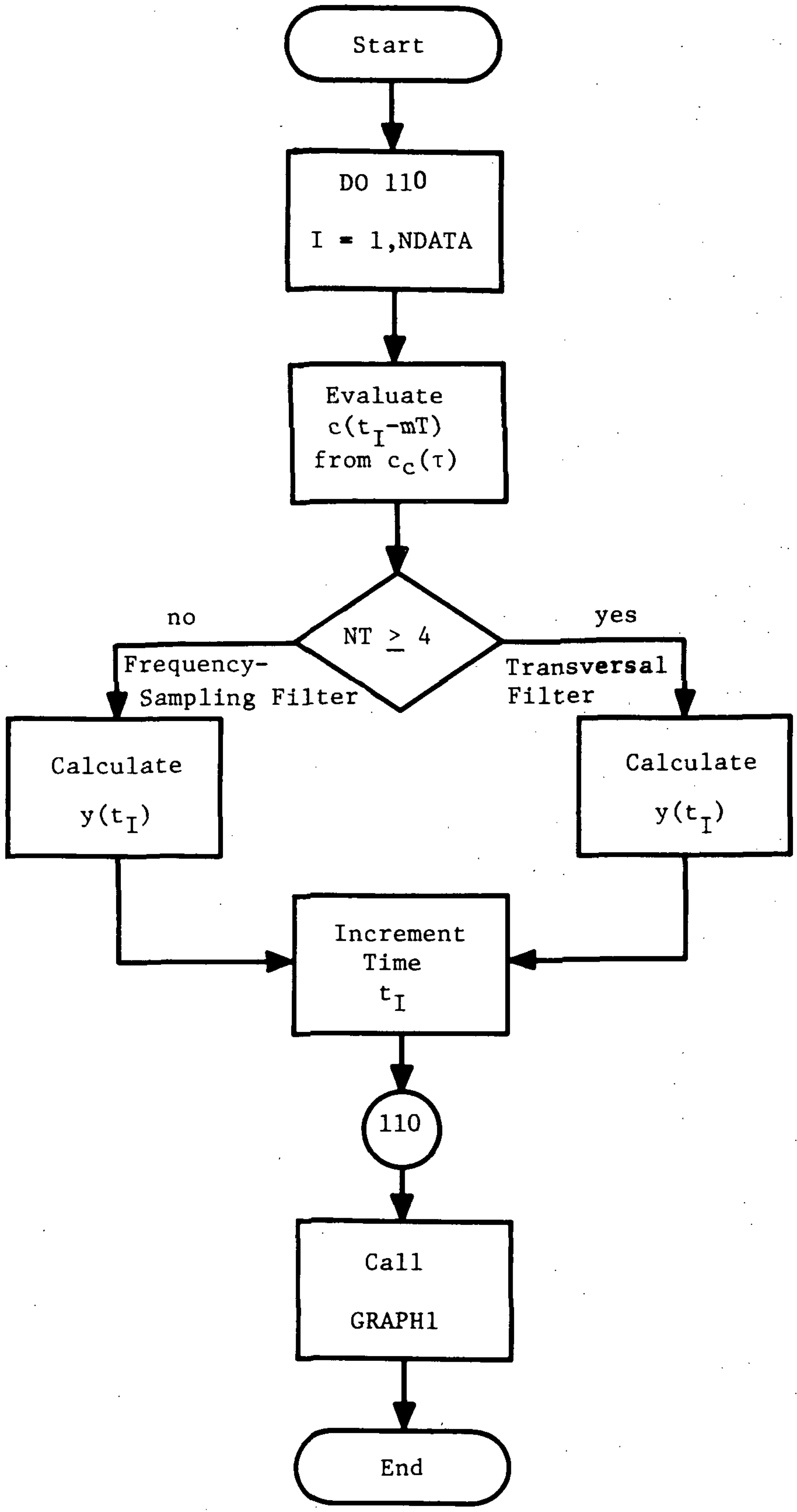


Fig. C-9. Flowchart for Subroutine CHECK.

```
      SUBROUTINE CHECK(TIN,TM,DT,M,T1,NVA,NT,CC)
C
C     THIS SUBROUTINE CALCULATES THE IMPULSE RESPONSE OF
C     AN EQUALIZED DATA TRANSMISSION CHANNEL. EITHER
C     TRANSVERSAL OR FREQUENCY-SAMPLING DIGITAL FILTERS
C     MAY BE USED AS EQUALIZERS.
C
      DOUBLE PRECISION H(40)
      DIMENSION A(500),YY(500)
      REAL*8 SUM,CC,DCOS,PI,D,V,Y(200),M,T1,K,N,X,Z,SUM1
      REAL*8   TIN,TM,DT,SUM2,TMP
      COMMON/HNAME/H
      EXTERNAL CC
      PI=3.1415926D0
      TMP=TIN
      NDATA=(TM-TIN)/DT+1
C     CALCULATE VALUE OF Y(T) FOR EACH TIME T (THRU
C     STATEMENT 110)
      DO 110 I=1,NDATA
      SUM2=0.0D0
      A(I)=SNGL(TIN)
      L=TIN/T1+1
      TTL=FLOAT(L)
      IF (DBLE(TTL).GT.M) L=M
      N=0.0D0
C     CALCULATE IMPULSE RESPONSE VALUES OF UNEQUALIZED
C     CHANNEL (THRU STATEMENT 104)
      DO 104 IJ=1,L
      X=TIN-N*T1
      Z=X-T1
      IF (Z) 100,100,101
100   CALL DQG16(0.0D0,X,CC,V)
      GO TO 102
101   CALL DQG16(Z,X,CC,V)
102   N=N+1.0D0
      IF(NT.LT.4) GO TO 103
      IF(H(IJ).EQ.0.0D0) GO TO 104
      SUM2=SUM2+V*H(IJ)
      GO TO 104
103   Y(IJ)=V
104   CONTINUE
C     DETERMINE Y(T) FOR TRANSVERSAL FILTER
      IF(NT.GE.4)YY(I)=SUM2
      IF(NT.GE.4) GO TO 109
      SUM1=0.0D0
105   K=0.0D0
C     CALCULATE Y(T) FOR FREQUENCY-SAMPLING FILTER
C     (THRU STATEMENT 109)
```

Fig. C-10. Program Listing for Subroutine CHECK.

```
      DO 108 J=1,NVA
      IF (H(J).EQ.0.0D0) GO TO 107
      SUM=0.0D0
      N=0.0D0
      DO 106 IJ=1,L
      D=(2.0D0*PI*K*N)/M
      SUM=Y(IJ)*DCOS(D)+SUM
      N=N+1.0D0
106   CONTINUE
      SUM1=SUM1+SUM*H(J)
107   K=K+1.0D0
108   CONTINUE
      YY(I)=SNGL(SUM1/M)
109   TIN=TIN+DT
110   CONTINUE
      TIN=TMP
      CALL GRAPH1(A,YY,NDATA,1)
      RETURN
      END
```

Fig. C-10 (Continued). Program Listing for Subroutine CHECK.

C.4 FUNCTION SUBPROGRAM CC

Function subprogram CC is called by the subroutines COCAL2 and CHECK to calculate channel impulse-response values. The eight models of typical channels which are available from the subprogram are listed in Table C.3. It should be noted that the channels have been normalized in the subprogram to have a unity breakpoint frequency.

TABLE C.3

IMPULSE-RESPONSE CHANNEL MODELS

NN	$c_c(t)$	NN	$c_c(t)$
1	$2\pi\, e^{-2\pi t}\, u(t)$	5	$\frac{1}{24}(2\pi)^5\, t^4\, e^{-2\pi t}\, u(t)$
2	$\pi^2 t e^{-2\pi t}\, u(t)$	6	$\frac{1}{120}(2\pi)^6\, t^5\, e^{-2\pi t}\, u(t)$
3	$\frac{1}{2}(2\pi)^3\, t^2\, e^{-2\pi t}\, u(t)$	7	$e^{-2\pi t} \sin\,(2\pi t)\;\; u(t)$
4	$\frac{1}{6}(2\pi)^4\, t^3\, e^{-2\pi t}\, u(t)$	8	$e^{-2\pi t} \cos\,(2\pi t)\; u(t)$

In order to specify which channel model is desired, the parameter NN must be declared with a blank COMMON statement and its value given in the main program that calls either COCAL2 or CHECK. The main program must also declare the function CC to be external. For example, see Fig. C-5 where the statement NN = 2 specifies the second channel response listed in Table C.3. The CC function program listing is given in Fig. C-11.

```
      DOUBLE PRECISION FUNCTION CC(T)
C
C     THIS FUNCTION SUBPROGRAM EVALUATES ONE OF EIGHT
C     POSSIBLE IMPULSE RESPONSE FUNCTIONS FOR THE
C     TIME POINT T.
C
      DOUBLE  PRECISION T,DEXP,ALPHA,ALPN,AT,TN,PROD,DSIN,DCOS
      COMMON NN
      N=NN
      ALPHA=6.2831852D0
      AT=ALPHA*T
      IF (AT.GE.165.D0) GO TO 601
      IF(N.LE.6) GO TO 603
      M=N-6
      GO TO(604,605),M
603   IF(N.EQ.1)GO TO 607
      IF(T.LT.1.D-9) GO TO 601
      TN=T**(N-1)
      ALPN=ALPHA**N
      GO TO 608
607   TN=1.0D0
      ALPN=1.0D0
608   PROD=1.0D0
      DO 600 I=2,N
600   PROD=(I-1)*PROD
C     CALCULATE CC(T) FOR CHANNEL WITH SIMPLE POLES
      CC=(DEXP(-AT)*ALPN*TN)/PROD
      GO TO 602
C     CALCULATE CC(T) FOR CHANNEL WITH COMPLEX POLES
604   CC=DEXP(-AT)*DSIN(AT)
      GO TO 602
C     CALCULATE CC(T) FOR CHANNEL WITH COMPLEX POLES
C     AND A SIMPLE ZERO
605   CC=DEXP(-AT)*DCOS(AT)
      GO TO 602
601   CC=0.0D0
      GO TO 602
606   CC=1.0D0
602   RETURN
      END
```

Fig. C-11. Program Listing for Function CC.

LIST OF REFERENCES

1. R. A. Kaenel, Ed., "Special issue on digital signal processing," IEEE Trans. Audio and Electroacoustics, vol. AU-18, December 1970.

2. H. Nyquist, "Certain topics in telegraph transmission theory," Trans. AIEE, vol. 47, pp. 617-644, April 1928.

3. J. M. Wozencraft and I. M. Jacobs, Principles of Communication Engineering, New York: John Wiley & Sons, 1967, Chapter 4.

4. H. Nyquist, "Certain factors affecting telegraph speed," Bell System Tech. J., vol. 3, pp. 324-326, April 1924.

5. W. R. Bennett and J. R. Davey, Data Transmission, New York: McGraw-Hill, 1965.

6. R. W. Lucky, J. Salz, and E. J. Weldon, Jr., Principles of Data Communication, New York: McGraw-Hill, 1968.

7. H. A. Wheeler, "The interpretation of amplitude and phase distortion in terms of paired echoes," Proc. IRE, vol. 27, pp. 359-385, June 1939.

8. E. D. Sunde, "Theoretical fundamentals of pulse transmission - I," Bell System Tech. J., vol. 33, pp. 721-788, May 1954.

9. E. D. Sunde, "Theoretical fundamentals of pulse transmission - II," Bell System Tech. J., vol. 33, pp. 987-1010, July 1954.

10. B. P. Bogert, "Demonstration of delay distortion correction by time-reversal techniques," IRE Trans. Communication Systems, vol. CS-5, pp. 2-7, December 1957.

11. I. Gerst and J. Diamond, "The elimination of intersymbol interference by input signal shaping," Proc. IRE, vol. 49, pp. 1195-1203, July 1961.

12. J. M. Aein and J. C. Hancock, "Reducing the effects of intersymbol interference with correlation receivers," IEEE Trans. Information Theory, vol. IT-9, pp. 167-175, July 1963.

13. D. A. George, "Matched filters for interfering signals," IEEE Trans. Information Theory, vol. IT-11, pp. 153-154, January 1965.

14. R. A. Gibby and J. W. Smith, "Some extensions of Nyquist's telegraph transmission theory," Bell System Tech. J., vol. 44, pp. 1487-1510, September 1965.

15. D. W. Tufts, "Nyquist's problem - the joint optimization of transmitter and receiver in pulse amplitude modulation," Proc. IEEE, vol. 53, pp. 248-259, March 1965.

16. M. R. Aaron and D. W. Tufts, "Intersymbol interference and error probability," IEEE Trans. Information Theory, vol. IT-12, pp. 26-34, January 1966.

17. J. W. Smith, "The joint optimization of transmitted signal and receiving filter for data transmission systems," Bell System Tech. J., vol. 44, pp. 2363-2392, December 1965.

18. T. Berger and D. W. Tufts, "Optimum pulse amplitude modulation parts I & II," IEEE Trans. Information Theory, vol. IT-13, pp. 196-216, April 1967.

19. H. B. Voelcker, "Generation of digital signaling waveforms," IEEE Trans. Communication Technology, vol. COM-16, pp. 81-93, February 1968.

20. P. J. Van Gerwen and P. Van Der Wurf, "Data modems with integrated digital filters and modulators," IEEE Trans. Communication Technology, vol. COM-18, pp. 214-222, June 1970.

21. D. J. Nowak and P. E. Schmid, "A nonrecursive digital filter for data transmission," IEEE Trans. Audio and Electroacoustics, vol. AU-16, pp. 343-349, September 1968.

22. A. Croisier and J. D. Pierret, "The digital echo modulation," IEEE Trans. Communication Technology, vol. COM-18, pp. 367-376, August 1970.

23. D. A. Spaulding, "Synthesis of pulse-shaping networks in the time domain," Bell System Tech. J., vol. 48, pp. 2425-2444, September 1969.

24. R. J. Tracey, "Optimization of pulse shaping filters for data transmission," 1970 IEEE Conference on Communications Record, vol. 6, pp. 24-7 - 24-12, June 1970.

25. D. A. Shnidman, "Optimum phase equalization of filters for digital signals," Bell System Tech. J., vol. 49, pp. 1531-1554, September 1970.

26. T. Yamabuchi, T. Takagi, and K. Mano, "Matched filters for rectangular, raised cosine, and half sine wave signals," IEEE Trans. Communication Technology, vol. COM-19, pp. 369-371, June 1971.

27. A. P. Brogle, "A new transmission method for pulse-code modulation communication systems," IEEE Trans. Communications Systems, vol. CS-8, pp. 155-160, September 1960.

28. A. Lender, "The duobinary technique for high-speed data transmission," IEEE Trans. Communication and Electronics, vol. 82, pp. 214-218, May 1963.

29. A. Lender, "Correlative level coding for binary-data transmission," IEEE Spectrum, vol. 3, pp. 104-115, February 1966.

30. J. M. Sipress, "A new class of selected ternary pulse transmission plans for digital transmission lines," IEEE Trans. Communication Technology, vol. COM-13, pp. 366-372, September 1965.

31. E. R. Kretzmer, "Generalization of a technique for binary data communication," IEEE Trans. Communication Technology, vol. COM-14, pp. 67-68, February 1966.

32. F. K. Becker, E. R. Kretzmer, and J. R. Sheehan, "A new signal format for efficient data transmission," Bell System Tech. J., vol. 45, pp. 755-758, May 1966.

33. R. D. Howson, "An analysis of the capabilities of polybinary data transmission," IEEE Trans. Communication Technology, vol. COM-13, pp. 312-319, September 1965.

34. F. K. Becker, L. N. Holzman, R. W. Lucky, and E. Port, "Automatic equalization for digital communication," Proc. IEEE, vol. 53, pp. 96-97, January 1965.

35. K. E. Schreiner, H. L. Funk, and E. Hopner, "Automatic distortion correction for efficient pulse transmission," IBM J., pp. 20-30, January 1965.

36. R. W. Lucky, "Automatic equalization for digital communication," Bell System Tech. J., vol. 44, pp. 547-588, April 1965.

37. R. W. Lucky, "Techniques for adaptive equalization of digital communication systems," Bell System Tech. J., vol. 45, pp. 255-286, February 1966.

38. C. Kaiteris and W. L. Rubin, "Generalized automatic equalization for communication channels," Proc. IEEE, vol. 54, pp. 439-440, March 1966.

39. A. Gersho, "Automatic time-domain equalization with transversal filters," Proc. National Electronics Conference, vol. 22, pp. 928-932, October 1966.

40. H. Rudin, Jr., "Automatic equalization using transversal filters," IEEE Spectrum, vol. 4, pp. 53-59, January 1967.

41. A. Gersho, "Adaptive equalization of highly dispersive channels for data transmission," Bell System Tech. J., vol. 48, pp. 55-70, January 1969.

42. M. G. Taylor, "Adaptive digital filtering for PAM data transmission," 1970 IEEE Conference on Communications Record, vol. 6, pp. 26-9 - 26-12, June 1970.

43. J. G. Proakis, "Adaptive digital filters for equalization of telephone channels," IEEE Trans. Audio and Electroacoustics, vol. AU-18, pp. 195-199, June 1970.

44. C. W. Niessen, "Adaptive equalizer for pulse transmission," IEEE Trans. Communication Technology, vol. COM-18, pp. 377-395, August 1970.

45. A. Lender, "Decision-directed digital adaptive equalization technique for high-speed data transmission," IEEE Trans. Communication Technology, vol. COM-18, pp. 625-632, October 1970.

46. A. Gersho, "Adaptive equalization for data transmission," Proc. National Electronics Conference, vol. 26, pp. 154-157, December 1970.

47. W. M. Cowan and J. G. Proakis, "Automatic telephone line equalization," 1971 IEEE Conference on Communications Record, vol. 7 pp. 38-12 - 38-17, June 1971.

48. H. E. Kallman, "Transversal Filters," Proc. IRE, vol. 28, pp. 302-310, July 1940.

49. R. W. Lucky, "Bell turns to MOS/LSI in digital equalizer," Electronics, vol. 44, p. 39, June 7, 1971.

50. T. G. Stockham, Jr., "High speed convolution and correlation," AFIPS Proc., 1966 Spring Joint Computer Conference, vol. 28, pp. 229-233, 1966.

51. J. F. Kaiser, "Digital Filters" in Systems Analysis by Digital Computer, F. F. Kuo and J. F. Kaiser, Eds., New York: John Wiley & Sons, 1966, pp. 218-285.

52. R. C. Houts and D. W. Burlage, "Computer-aided design of digital filters," University of Alabama Bureau of Engineering Research Technical Report No. 128-102, March 1971.

53. M. A. Martin, "Frequency domain applications to data processing," IRE Trans. Space Electronics and Telemetry, vol. SET-5, pp. 33-41, March 1959.

54. R. J. Graham, "Determination and analysis of numerical smoothing weights," NASA Technical Report No. TR R-179, December 1963.

55. E. B. Anders, et al., "Digital filters," NASA Contract Report No. CR-136, December 1964.

56. H. D. Helms, "Nonrecursive digital filters: design methods for achieving specifications on frequency response," IEEE Trans. Audio and Electroacoustics, vol. AU-16, pp. 336-342, September 1968.

57. A. J. Monroe, Digital Processes for Sampled Data Systems, New York: John Wiley & Sons, 1962, Chapters 9 and 12.

58. R. K. Cavin, C. H. Ray, and V. T. Rhyne, "The design of optimal convolutional filters via linear programming," IEEE Trans. Geoscience Electronics, vol. GE-7, pp. 142-145, July 1969.

59. C. M. Rader and B. Gold, "Digital filter design techniques in the frequency domain," Proc. IEEE, vol. 55, pp. 149-171, February 1967.

60. B. Gold and K. L. Jordan, Jr., "A note on digital filter synthesis," Proc. IEEE, vol. 56, pp. 1717-1718, October 1968.

61. C. J. Weinstein, "Quantization effects in digital filters," M.I.T. Lincoln Laboratory Technical Report 468, November 1969, DDC AD706862.

62. B. Gold and K. L. Jordan, Jr., "A direct search procedure for designing finite duration impulse response filters," IEEE Trans. Audio and Electroacoustics, vol. AU-17, pp. 33-36, March 1969.

63. L. R. Rabiner, B. Gold, and C. A. McGonegal, "An approach to the approximation problem for nonrecursive digital filters," IEEE Trans. Audio and Electroacoustics, vol. AU-18, pp. 83-106, June 1970.

64. L. R. Rabiner, "Techniques for designing finite-duration impulse-response digital filters," IEEE Trans. Communication Technology, vol. COM-19, pp. 188-195, April 1971.

65. G. B. Dantzig, "Maximization of a Linear Function of Variables Subject to Linear Inequalities" in Activity Analysis of Production and Allocation, T. C. Koopmans, Ed., New York: John Wiley & Sons, 1951, Chapter XXI.

66. S. T. Gass, Linear Programming, New York: McGraw-Hill, 1964.

67. G. Hadley, Linear Programming, Reading, Mass.: Addison-Wesley, 1962.

68. N. Ricker, "Wavelet contraction, wavelet expansion, and the control of seismic resolution," Geophysics, vol. 18, pp. 769-792, October 1953.

69. A. E. Drake and E. Streit, "University of Alabama Statistical Monitor," University of Alabama Computer Center Manual, September 1969.

COMMUNICATION SYSTEMS GROUP

RECENT REPORTS