University of Cincinnati

Cincinnati, Ohio 45221

COLLEGE OF ENGINEERING

It is hereby recommended that the thesis

prepared by Edward John Delp III

entitled SPECTRAL ANALYSIS AND SYNTHESIS USING WALSH FUNCTIONS.

be accepted as fulfilling this part of the requirements

for the degree MASTER OF SCIENCE.

Alfred W. Scheide

[illegible]

Carl H Osterbrock

June 4, 19 75

SPECTRAL ANALYSIS AND SYNTHESIS USING WALSH FUNCTIONS

A thesis submitted to the

Department of Electrical Engineering

College of Engineering

Division of Graduate Studies

UNIVERSITY OF CINCINNATI

in partial fulfillment of the
requirements for the degree of

Master of Science

1975

by

EDWARD JOHN DELP III

B.S.E.E. University of Cincinnati 1973

ACKNOWLEDGMENTS

The author wishes to thank Dr. A. W. Scheide (chairperson), Professor R. H. Engelmann, and Dr. C. H. Osterbrock, members of his thesis advisory committee for the interest which they contributed toward this thesis. The author is indebted to Dr. Scheide for introducing him to Walsh functions.

The author would also like to thank Mr. Michael Salas for his help in constructing the Walsh function generator. A special acknowledgment is extended to Mr. Delvan Ramey for his long, and sometimes loud, discussion sessions concerning digital signal processing.

The author also wishes to extend a very special acknowledgment to his wife, Maureen, for her help in preparing the manuscript and for her understanding and encouragement she gave to the completion of this project.

Statement of Object

The object of this thesis is to describe the usefulness of Walsh functions for spectral analysis and synthesis. Waveform synthesis will be accomplished using a Walsh function generator. The Fast Walsh Transform will be used to analyze a simple sequency-lowpass filter. A Discrete Walsh-Fourier Transform computer program package will also be described.

TABLE OF CONTENTS

List of Illustrations

Figure Number	Title	Page
4-14	A Time-Invariant Linear System	88
4-15	A Dyadic-Invariant Linear System	89
4-16	Impulse Response of a Zero-Order Hold.	92
4-17	An Ideal Sequency Lowpass Filter	93
4-18	Sampler and Zero-Order Hold System	95
4-19	Flow Chart for Analysis of ZOH Using Sequency Transfer Function.	98
4-20	Flow Chart for Analysis of ZOH Using Frequency Transfer Function.	99
4-21	Frequency Transfer Function of ZOH Described in Figure 4-16.	100
4-22	Output of ZOH Using Sequency Transfer Function (Sine-Wave Input).	102
4-23	Output of ZOH Using Frequency Transfer Function (Sine-Wave Input).	103
4-24	Output of ZOH Using Sequency Transfer Function . . (Ramp Input)	104
4-25	Output of ZOH Using Frequency Transfer Function (Ramp Input)	105
4-26	Output of ZOH Using Sequency Transfer Function (Half-Wave Rectified Sine-Wave Input).	106
4-27	Output of ZOH Using Frequency Transfer Function (Half-Wave Rectified Sine-Wave Input).	107
4-28	Output of ZOH Using Sequency Transfer Function (Full-Wave Rectified Sine Wave Input).	108
4-29	Output of ZOH Using Frequency Transfer Function (Full-Wave Rectified Sine-Wave Input).	109
4-30	Output of ZOH Using Sequency Transfer Function (Triangular Function Input).	110
4-31	Output of ZOH Using Frequency Transfer Function (Triangular Function Input).	111

List of Tables

Chapter 1

INTRODUCTION OF MATHEMATICAL CONCEPTS

1.0 Introduction

The method of Fourier series and Fourier transform analysis has been used extensively for over a hundred years in solving engineering and science-related problems. Fourier techniques have long been recognized as a powerful tool for the engineer in the study of everything from antenna construction to optics.[1] In recent years, engineers have been interested in applying Fourier techniques to other sets of functions besides the sine-cosine set that is usually used in Fourier analysis. The set of functions that seem to have received most of the publicity recently is the complete orthonormal set of square waves known as Walsh functions.

The purpose of this chapter is to quickly review why Fourier techniques work and briefly discuss some classical sets of orthogonal functions. The description of Walsh functions will then be discussed along with certain properties related to them.

1.1 Generalized Fourier Series

Given a set of real valued functions $\{g(i,x)\}$ which are defined as some finite half open interval, say $(a,b]$, these functions are said to be orthogonal on the above interval if:

$$\int_a^b g(i,x)g(j,x)\,dx = \begin{cases} K \text{ if } j = i \\ 0 \text{ if } i \neq j \end{cases} \quad i = 1,2,3,\ldots. \tag{1-1}$$

[1]For more detailed discussion of Fourier techniques, see Reference 17.

Furthermore, these functions are said to be orthonormal if $K = 1$. An important property of a set of orthogonal functions is whether it is complete or not. Completeness is very difficult to explain in simple words and even harder to prove mathematically, but suffice it to say a general function cannot be expanded in a series of incomplete orthogonal functions.[2]

The set of orthogonal functions presented above is a denumerable set, i.e. a set containing an infinite but countable number of functions. The variable i appears to serve only as a method of indexing the functions as used in equation 1-1. However, this variable usually is connected in some way with how a particular function of the set is described mathematically. In classical Fourier analysis using the sine-cosine set of functions the integer i is used to represent a multiple of the fundamental frequency. This type of argument also holds for other sets of orthogonal functions, including Walsh functions. This type of notation will also provide some continuity when discussing the generalized Fourier Transform.

The French physicist, Joseph Fourier, showed in the early 19th century that any "well behaved" function, $f(x)$, defined on the same interval as a to b can be represented by a series expansion of a linear weighted sum of a complete set of orthogonal functions, i.e.,

$$f(x) = \sum_{i=0}^{\infty} c(i)g(i,x) \tag{1-2}$$

[2]See Reference 92, pp. 311-327 or Reference 53, pp. 134-137 for a more detailed description of orthogonal functions and the generalized expansion problem.

where the sum has a vanishing mean square error. Fourier used this idea mainly to solve heat transfer problems and later it was used to solve other kinds of boundary value problems.

The value of the coefficients c(i) may be obtained by multiplying equation 1-2 by g(j,x) and integrating the products in the interval of orthogonality using the orthogonality principle of equation 1-1.

The results are:

$$f(x) = \sum_{i=0}^{\infty} c(i)g(i,x) \tag{1-3a}$$

$$c(i) = \frac{1}{K}\int_a^b g(i,x)f(x)dx \tag{1-3b}$$

c(i) is usually referred to as the ith generalized Fourier coefficient and equation 1-3a is referred to as the generalized Fourier series expansion of f(x).

Harmuth has discussed the generalized expansion problem stated above and has derived various properties of Fourier series such as Bessel's inequality and Parseval's Theorem in terms of generalized orthogonal functions.[3]

The set of orthogonal functions engineers are most familiar with is the set $\{1, \sqrt{2}\cos 2\pi ix, \sqrt{2}\sin 2\pi ix\}$. This set of functions is orthonormal in the interval (0,1]. These functions are sometimes combined using Euler's identity and are written using the complex exponential function exp(jix) where $j = \sqrt{-1}$.

---

[3]See Reference 43, pp. 14-16.

A function f(x) defined on the above interval will have the following sine-cosine expansion:

$$f(x) = a(0) + \sqrt{2} \sum_{i=1}^{\infty} [a(i)\cos 2\pi ix + b(i)\sin 2\pi ix]$$

$$a(0) = \int_0^1 f(x)dx \tag{1-4}$$

$$a(i) = \sqrt{2}\int_0^1 f(x)\cos 2\pi ix \, dx$$

$$b(i) = \sqrt{2}\int_0^1 f(x)\sin 2\pi ix \, dx$$

Other sets of orthogonal functions have found use in engineering problems; among these are Bessel functions and Legendre polynomials.

Bessel functions arrive from the solution of Bessel's differential equation and have many uses in solving problems with cylindrical symmetry such as circular waveguides. Another use of Bessel functions is to describe the spectral content of a frequency modulated carrier. Legendre polynomials are solutions to Legendre's differential equations and have uses in solving problems with spherical symmetry such as the Schrodinger wave equation in quantum mechanics.

Most "well behaved" functions can be expanded in a series of the three sets of orthogonal systems discussed so far. Whether a certain function f(x) can be expanded in a series of a particular orthogonal system cannot be told from such simple features of f(x) as its continuity or boundedness. For instance, the Fourier series of a continuous function does not have to converge at every point. A theorem due to Banach states that there are arbitrarily many orthogonal systems with the feature, that the orthogonal series of a continuous differentiable function diverges almost everywhere.[4]

1.2 Generalized Fourier Transform

The development of a generalized Fourier Transform is beyond the scope of this thesis. However, the results presented by Harmuth[5] will be quickly summarized below:

Fourier Transforms, sometimes referred to as orthogonal transforms, are closely related to the Fourier series developed above, one of the differences being that the Fourier series uses a set of denumerable functions whereas the Fourier transform uses a system of non-denumerable functions. In a heuristic sense, one can think of the Fourier transform coming about by increasing the interval of orthogonality to $(-\infty,\infty)$.

The generalized Fourier Transform does not use a set of orthogonal functions but rather a particular function, $g(y,x)$, called a Fourier kernel. The kernel function $g(y,x)$ is closely related to the set $\{g(i,x)\}$ used for Fourier series, one of the differences being that the variable y can take on any real value where i was limited to integer values only.[6]

Therefore, let $f(x)$ be any "well behaved" function; the generalized Fourier transform of $f(x)$ is defined as:

$$a(y) = \int_{-\infty}^{\infty} f(x)\ g(y,x)dx \qquad (1\text{-}5a)$$

[4]See Reference 30.

[5]See Reference 43, pp. 44-61

[6]For a discussion of Fourier kernels see Reference 17, pp. 250-251.

where a(y) is called the generalized Fourier spectra of f(x). The generalized inverse transform is defined as:

$$f(x) = \int_0^\infty a(y)\ g(y,x)dy \tag{1-5b}$$

Equation 1-5 is known as the generalized Fourier transform pair and should be compared to equation 1-3.

Sometimes the kernel function g(y,x) is divided into odd and even functions denoted $g_s(y,x)$ and $g_c(y,x)$ respectively. The Fourier transform pair becomes:

$$\begin{aligned} a_c(y) &= \int_{-\infty}^{\infty} f(x)\ g_c(y,x)dx \\ a_s(y) &= \int_{-\infty}^{\infty} f(x)\ g_s(y,x)dx \end{aligned} \tag{1-6a}$$

$$f(x) = \int_0^\infty [a_c(y)g_c(y,x) + a_s(y)g_s(y,x)]dy \tag{1-6b}$$

$a_c(y)$ and $a_s(y)$ are called the generalized odd and even Fourier spectra of f(x) respectively.

If the sine-cosine kernel functions are used the Fourier transform of f(x) becomes:

$$\begin{aligned} a_c(y) &= \sqrt{2} \int_{-\infty}^{\infty} f(x)\ \cos 2\pi yx dx \\ a_s(y) &= \sqrt{2} \int_{-\infty}^{\infty} f(x)\ \sin 2\pi yx dx \end{aligned} \tag{1-7a}$$

$$f(x) = \sqrt{2} \int_0^\infty [a_c(y)\cos 2\pi yx + a_s(y)\sin 2\pi yx]dy \tag{1-7b}$$

which are the standard equations presented for the Fourier transform, $a_c(y)$ being the cosine transform and $a_s(y)$ being the sine transform.

Harmuth has considered examples of the generalized Fourier transform using Legendre polynomials, the sine-cosine set, and Walsh functions.

1.3 Normalized Time and Frequency

In most signal processing work one is usually interested in finding Fourier series and Fourier transforms of time functions. The following definitions are to be used in Fourier work using time functions:

$$\theta = \frac{t}{T_n} \qquad \nu = fT_n \tag{1-8}$$

where t = time, seconds

T_n = normalizing time base, seconds

θ = normalized time, dimensionless

f = frequency, hertz

ν = normalized frequency, cycles per unit interval

Using the definitions of equation 1-8 the following sine-cosine set of functions are orthonormal on the interval $0<\theta\leq 1$, $\{1, \sqrt{2}\cos 2\pi i\theta, \sqrt{2}\sin 2\pi i\theta\}$.

The following is the Fourier series expansion of any well behaved function $h(\theta)$ on the above interval.

$$h(\theta) = a(0) + \sqrt{2}\sum_{i=1}^{\infty}[a(i)\cos 2\pi i\theta + b(i)\sin 2\pi i\theta]$$

$$a(0) = \int_0^1 h(\theta)d\theta \tag{1-9}$$

$$a(i) = \sqrt{2}\int_0^1 h(\theta)\cos 2\pi i\theta d\theta$$

$$b(i) = \sqrt{2}\int_0^1 h(\theta)\sin 2\pi i\theta d\theta$$

In most engineering work, when one is talking about Fourier series expansion of time functions one usually assumes the function is periodic with some known period. The theory of the generalized Fourier series does not mention the concept of periodicity and in fact nothing is implied or stated about what happens to the function $h(\theta)$ or even the orthogonal functions themselves outside the interval of orthogonality. Applied mathematicians using the concept of periodic sine-cosine elements have "extended" Fourier series techniques to be used with "periodic" functions. Harmuth has commented on this when extending Walsh functions, but one should realize the concept of periodic continuation is really an artificial extension of Fourier series techniques used very successfully in engineering work.

The Fourier series described in equation 1-9 may look odd to those who are used to the periodic expansion concept as usually described in an initial presentation of Fourier analysis. However, if the period of the particular time function is chosen to be the normalizing time base T_n, equation 1-9 reduces to the standard Fourier series for periodic time functions. The normalized frequency therefore becomes the integer i or:

$$\begin{aligned} i &= fT_n \\ f &= \frac{i}{T_n} \end{aligned} \tag{1-10}$$

which are standard results.

This concept of normalized frequency and normalized time can be extended to the Fourier transform; however, as was discussed in section 1.2 the parameter i is allowed to take on any real value, hence the normalized frequency is denoted by ν and takes non-integer values as does the unnormalized frequency f.

The most important result to come out of this discussion is that the sine-cosine set of orthogonal functions are <u>independent</u> of the normalizing time base T_n, i.e.:

$$\cos 2\pi i\theta = \cos 2\pi (fT_n) \frac{t}{T_n} = \cos 2\pi ft$$

$$\text{or} \tag{1-11}$$

$$\cos 2\pi \nu\theta = \cos 2\pi (fT_n) \frac{t}{T_n} = \cos 2\pi ft$$

This is not so for other systems of orthogonal functions whose primary describing parameter, f and t in this case, are not connected by multiplication.

1.4 Walsh Functions

Most engineers are familiar with square waves and their various uses. Mathematically, square waves, known as Rademacher functions, are an incomplete set of orthonormal functions which were developed in 1922.[7] The Rademacher functions of index m, denoted by rad(m,θ), are a train of rectangular pulses with 2^{m-1} cycles in the half-open interval [0,1), with the exception of the rad(0,θ) which is the unit step (see Figure 1-1). If the Rademacher functions are periodically extended, then they satisfy the relation:

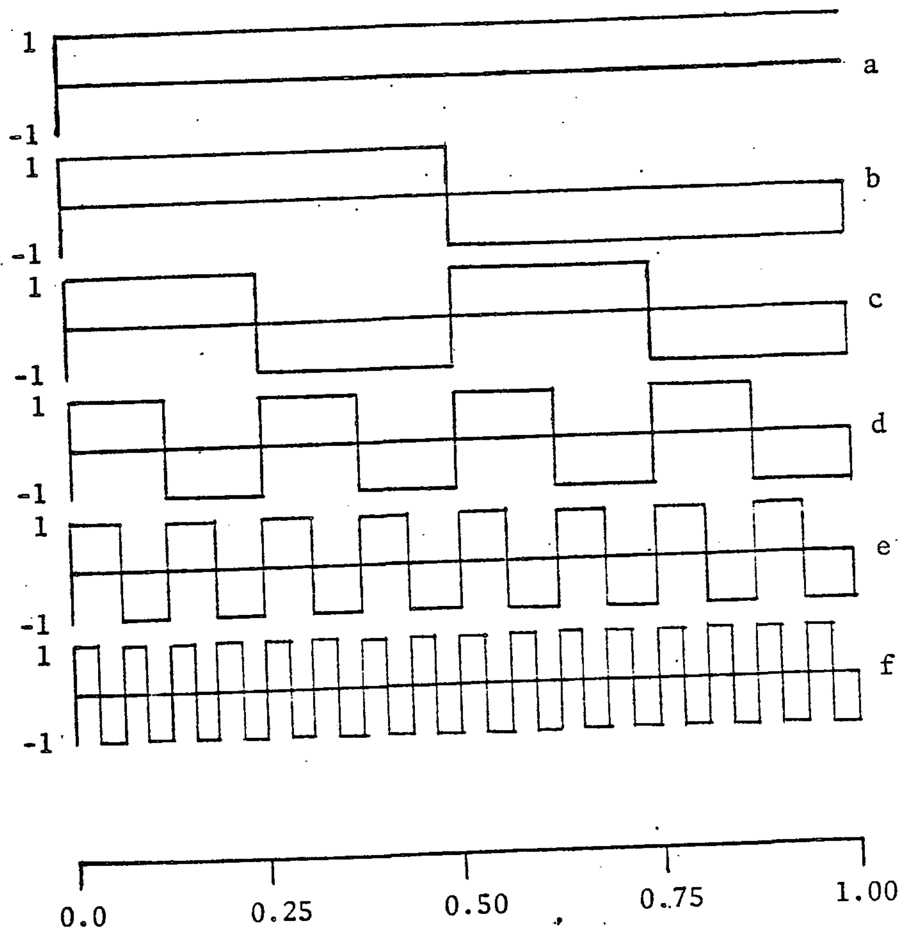
$$\text{rad}(m,\theta + n2^{1-m}) = \text{rad}(m,\theta) \tag{1-12}$$

$$m = 1,2,3,\ldots.$$

$$n = \pm 1,\pm 2,\ldots.$$

$$\theta = \text{normalized time}$$

[7]See Reference 75.



θ
Normalized Time

a	rad(0,θ)
b	rad(1,θ)
c	rad(2,θ)
d	rad(3,θ)
e	rad(4,θ)
f	rad(5,θ)

Figure 1-1 The First Six Rademacher Functions

Rademacher functions can be generated using the recurrence relation

$$\mathrm{rad}(m,\theta) = \mathrm{rad}(1,2^{m-1}\theta)$$

with

$$\mathrm{rad}(1,\theta) = \begin{cases} 1, \theta\varepsilon[0,1/2) \\ -1, \theta\varepsilon[1/2,1) \end{cases} \qquad (1\text{-}13)$$

Hence, Rademacher functions are just ordinary square waves whose frequencies differ by two, but most importantly, they are orthonormal and satisfy equation 1-1. Rademacher functions are an incomplete set of functions, hence they cannot be used for Fourier series expansions. This can be seen intuitively by noting that the Rademacher functions are odd functions; hence it would be impossible to expand an even function as a series of odd functions.

In 1923, J. L. Walsh[8] completed the set of Rademacher functions and described the properties of these new functions. He also discussed Fourier series expansions using these functions. The functions have become known as Walsh functions and are denoted as wal(n,θ). The first eight Walsh functions are shown in Figure 1-2.

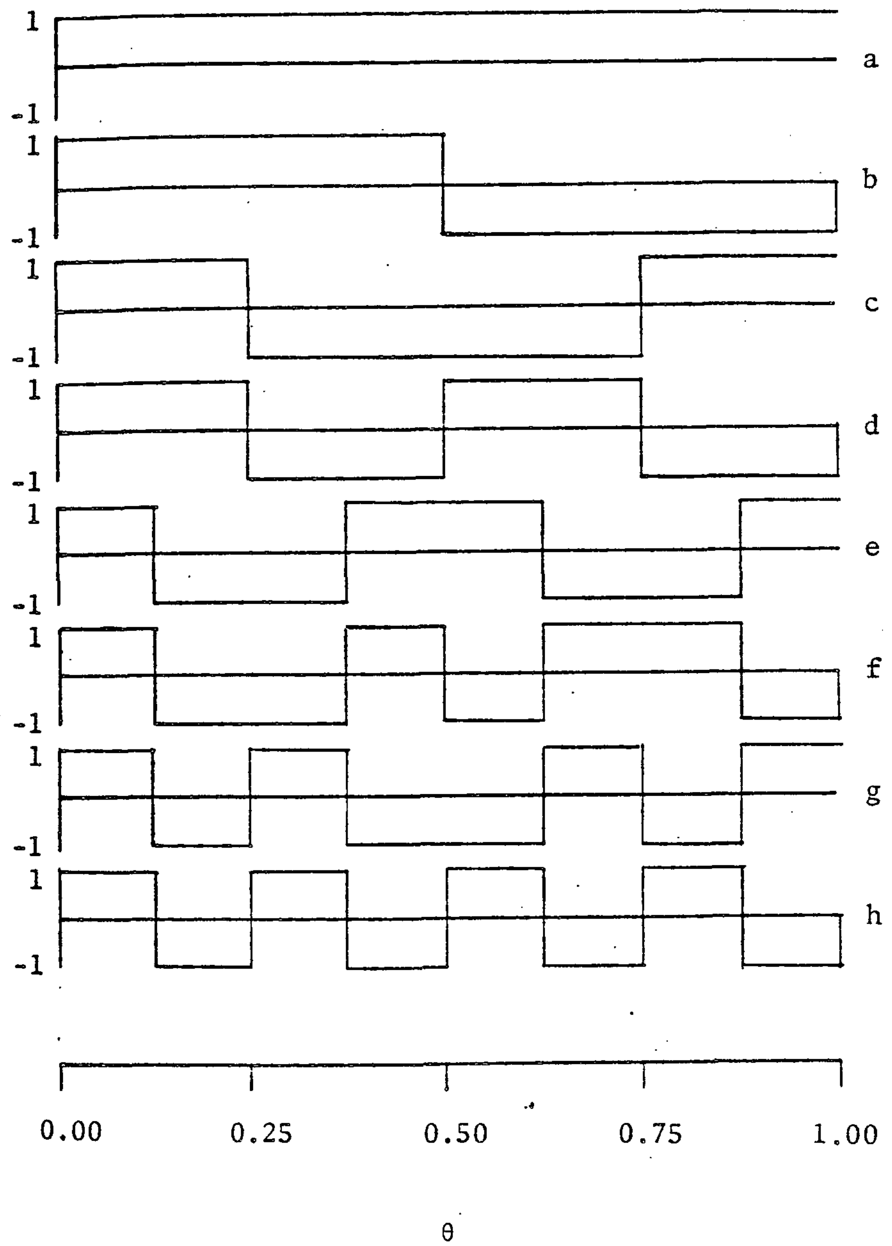
Various people have commented on how a set of Walsh functions can be generated. Harmuth[9] uses a difference equation to generate a set of Walsh functions while Lackey and Maltzer[10] use the Gray code representation of n and a set of Rademacher functions.

---

[8]Walsh Functions are a complete orthonormal set of square waves, while Rademacher Functions are an incomplete orthonormal set of square waves. See Reference 90.

[9]See Reference 43, p. 23.

[10]See Reference 55.



a	wal(0,θ)	[cal(0,θ)]
b	wal(1,θ)	[sal(1,θ)]
c	wal(2,θ)	[cal(2,θ)]
d	wal(3,θ)	[sal(3,θ)]
e	wal(4,θ)	[cal(4,θ)]
f	wal(5,θ)	[sal(5,θ)]
g	wal(6,θ)	[cal(6,θ)]
h	wal(7,θ)	[sal(7,θ)]

Figure 1-2 The First Eight Walsh Functions

The product of two Walsh functions yields another Walsh function:

$$\text{wal}(h,\theta)\ \text{wal}(k,\theta) = \text{wal}(r,\theta)$$

This relation may readily be proved using the previous difference equation; however, determination of the value of r from the difference equation is somewhat cumbersome. The result is that r equals the modulo 2 sum of h and k:

$$\text{wal}(h,\theta)\ \text{wal}(k,\theta) = \text{wal}(h\oplus k,\theta) \tag{1-14}$$

where $\oplus$ indicates modulo 2 addition. k and h are written as binary numbers and added bitwise according to the rules $0\oplus 1 = 1$, $1\oplus 0 = 1$, $0\oplus 0 = 0$ (no carry). Equation 1-14 is sometimes referred to as the multiplication law or the modulation relation and can be easily verified using Figure 1-2. The concept of modulo 2 addition, sometimes referred to as dyadic addition, is used extensively with Walsh functions. Other properties of Walsh functions using dyadic addition will be developed later.[11]

One notes from Figure 1-2 that Rademacher functions are a subset of Walsh functions:

$$\text{rad}(m,\theta) = \text{wal}(2^m-1,\theta) \quad m = 1,2,3,\ldots. \tag{1-15}$$

This fact, along with the multiplication law will be used in Chapter 2 to develop a Walsh function generator.

[11]The modulo 2 addition arises because, mathematically speaking, the group of Walsh functions is isomorphic to the discrete dyadic group. The dyadic group is the topologic group derived from the set of binary representations of the real numbers. See Reference 43, pp. 24-28.

Using equation 1-2 the following Walsh-Fourier series can be written for a function f(θ) defined on the unit interval:

$$f(\theta) = \sum_{n=0}^{\infty} c(n)\,\text{wal}(n,\theta)$$
$$c(n) = \int_0^1 f(\theta)\,\text{wal}(n,\theta)\,d\theta \qquad (1\text{-}16)$$

where c(n) is referred to as the nth Walsh coefficient.

It is appropriate at this point to discuss the general notation used with Walsh functions and how it is related to sine-cosine functions. Harmuth has grouped the Walsh functions by odd or evenness and by the number of sign changes per interval. Harmuth calls the even Walsh functions <u>cal</u> functions and the odd Walsh functions <u>sal</u> functions. This notation is analogous to the even sine-cosine functions being called cosine functions and the odd ones called sine functions. The functions are further ordered by the number of sign changes in the half open interval (0,1].[12] The number of sign changes per unit interval divided by two is called the normalized <u>sequency</u> of the Walsh function, sequency being analogous to frequency. Sequency will be discussed later. Therefore,

$$\begin{aligned} &\text{sal}\,(i,\theta) = \text{wal}(2i-1,\theta) \\ &\text{cal}\,(i,\theta) = \text{wal}(2i,\theta) \\ &i = 1,2,3,.... \\ &\text{sal}\,(0,\theta) \text{ is undefined} \\ &\text{cal}\,(0,\theta) = \text{wal}(0,\theta) = 1 \\ &\text{where } i = \text{normalized sequency} \\ &\qquad \theta = \text{normalized time} \end{aligned} \qquad (1\text{-}17)$$

[12]The integer n is equal to the number of sign changes in the open unit interval (0,1) of the function wal(n,θ).

The normalized sequency i takes on only integer values and is similar to the normalized frequency discussed in section 1.3. The unnormalized sequency is:

$$\phi = \frac{i}{T_n} \tag{1-18}$$

and is measured in zero crossings per second or ZPS.

As defined above, the normalized sequency is one half the number of sign changes in the half-open interval (0,1]. Harmuth has used the concept of sequency as a generalization of frequency; in fact, one can talk about the sequency of a sine or cosine function. Frequency for sine or cosine functions is measured in cycles per second or hertz; thus a 100 Hz sine wave has 100 cycles per second or 200 sign changes per second, hence a 100 Hz sine wave has a sequency of 100 zps. One can see that for sine waves, the concept of sequency and frequency is identical; however, for Walsh functions the idea of frequency has no meaning and one can only talk about its sequency. In fact, the concept of a generalized frequency or sequency is extremely important and has many applications. Harmuth has extended the concept of sequency to other sets of orthogonal functions, particularly Bessel functions and Legendre polynomials.

Using the notation introduced by equation 1-17 one can rewrite equation 1-16 as follows:

$$f(\theta) = a(0) + \sum_{i=1}^{\infty} [a(i)\mathrm{cal}(i,\theta) + b(i)\mathrm{sal}(i,\theta)]$$

$$a(0) = \int_0^1 f(\theta)d\theta \tag{1-19}$$

$$a(i) = \int_0^1 f(\theta)\mathrm{cal}(i,\theta)d\theta$$

$$b(i) = \int_0^1 f(\theta)\mathrm{sal}(i,\theta)d\theta \qquad (1\text{-}19)$$

This should be compared to equation 1-9.

The Fourier kernel required for the Walsh-Fourier transform is due to Fine[13], who also pointed out the existence of such a transform. The correct mathematical theory of the Walsh-Fourier transform using sal and cal functions, which are somewhat different from the system used by Fine, is due to Pichler.[14]

The Walsh-Fourier transform of a function $f(\theta)$ and its inverse have the following form:

$$a_c(\mu) = \int_{-\infty}^{\infty} f(\theta)\mathrm{cal}(\mu,\theta)d\theta$$

$$a_s(\mu) = \int_{-\infty}^{\infty} f(\theta)\mathrm{sal}(\mu,\theta)d\theta \qquad (1\text{-}20a)$$

$$f(\theta) = \int_0^{\infty} [a_c(\mu)\mathrm{cal}(\mu,\theta) + a_s(\mu)\mathrm{sal}(\mu,\theta)]d \qquad (1\text{-}20b)$$

where $a_c(\mu)$ and $a_s(\mu)$ are the even and odd Walsh-Fourier transforms of $f(\theta)$ respectively. μ is the normalized sequency and it is a non-negative real number.

It may seem at first that a new symbol, μ, has been introduced for normalized sequency, however the reader is reminded that this "new" normalized sequency is a generalization of the sequency, i, defined

[13]See Reference 34.

[14]See Reference 68.

above in equation 1-17, This generalization was necessary to allow the normalized sequency to take on non-integer values as required by the Fourier transform. This "new" sequency can be unnormalized as before:

$$\phi = \frac{\mu}{T_n}$$

where ϕ = sequency, zps.

This is consistent with the normalized frequency, ν, as discussed in section 1.3.

One may ask what does a Walsh function look like when the normalized sequency is not an integer? In other words, what does, say, cal(1.386,θ) look like? The function cos(1.386,θ) is easily defined; can any parallels be drawn from this? The answer is no. It is beyond the scope of this thesis to discuss non-integer normalized sequency. When μ takes on non-integer values, the problem is very difficult and one has to determine if μ is dyadic rational or not.[15] In fact, cal(μ,θ) and sal(μ,θ) are not periodic if μ is not dyadic rational, but the interpretation of μ holds true.[16]

The Walsh-Fourier transform of equation 1-20 can be used in the following generalization of the convolution integral. Let f(θ) and g(θ) be defined on $(-\infty,\infty)$ then the dyadic convolution product $f \circledast g$ is defined as:

$$f \circledast g(\theta) = \int_{-\infty}^{+\infty} f(\theta \oplus \tau) g(\tau) d\tau \qquad (1\text{-}21)$$

[15]A number, μ, is called dyadic rational if μ, when written as a binary number, has a finite number of terms to the right of the binary decimal point.

[16]See Reference 43, pp. 26-28.

There is really no need to distinguish between dyadic correlation and dyadic convolution since addition and subtraction modulo 2 are an identical operation.[17] In the literature, equation 1-21 is sometimes referred to as the <u>logical convolution</u> integral. The similarity with standard or <u>arithmetic</u> convolution or correlation is obvious.

Using the above definition, the following theorem is stated without proof.[18] Let $f(\theta)$, $g(\theta)$ be defined on $(-\infty,\infty)$ and $h(\theta) = f(\theta) \circledast g(\theta)$, let F_c, F_s, G_c, G_s, H_c, H_s, denote the even and odd Walsh-Fourier transforms of f, g, and h respectively, therefore:

$$H_c = F_cG_c \text{ and } H_s = F_sG_s \tag{1-22}$$

This theorem is analogous to the theorem used in conjunction with arithmetic convolution and sine-cosine Fourier transforms. As was previously mentioned, the concept of dyadic addition is extremely important when using Walsh functions. The discrete version of equations 1-21 and 1-22 will be investigated as part of this thesis.

For the reasons stated above, concerning non-integer values of μ, calculations involving the Walsh-Fourier transform of equation 1-20a are usually avoided; even calculation of the Walsh-Fourier series is somewhat cumbersome to evaluate as compared to the sine-cosine Fourier series.[19] However, the beauty and ease of Walsh-Fourier analysis will be seen when using discrete data.

---

[17]See Reference 43, pp. 26-28.

[18]See Reference 70.

[19]Another reason why the Walsh-Fourier transform and Walsh-Fourier series are difficult to evaluate is that the integration theorems are not

1.5 Comparison of Describing Parameters between Walsh and Sine Functions

In this section a review of the describing parameters of Walsh and sine functions will be presented and discussed briefly.

The following are usual describing parameters for sine functions:

$$V_m \sin(2\pi\nu\theta + \alpha)$$

where V_m = maximum amplitude

ν = normalized frequency, cycles

θ = normalized time, dimensionless

α = phase shift, radians

$f = \frac{\nu}{T_n}$, frequency in hertz

$t = \theta T_n$, time in seconds

T_n = normalizing time base in seconds

Let the normalized variables ν and θ in $\sin 2\pi\nu\theta$ be replaced by the non-normalized variables f and t:

$$V_m \sin(2\pi\nu\theta + \alpha) = V_m \sin(2\pi f t + \alpha)$$

The result is independent of the time base T_n, as was previously pointed out.

The following are the usual describing parameters for Walsh functions sal (μ,θ) and cal(μ,θ):

$$V_m \mathrm{sal}(\mu,\theta)$$

where V_m = maximum amplitude

μ = normalized sequency

θ = normalized time

---

available to evaluate the integrals of equations 1-20a and 1-19. Fine did develop some of the required integration theorems which were later used by Johnson to develop a very limited set of Walsh-Fourier transform pairs. See References 34 and 50.

If the parameters are unnormalized and a time delay is inserted for the purpose of generality, the function $\mathrm{sal}(\mu,\theta)$ becomes:

$$V_m \mathrm{sal}(\phi T_n, \frac{t-t_o}{T_n})$$

where V_m = maximum amplitude

ϕ = sequency, zps.

t = time, seconds

t_o = time delay, seconds

T_n = normalizing time base, seconds

μ = ϕT, normalized sequency, zeroes

θ = $\frac{t}{T_n}$, normalized time

θ_o = $\frac{t_o}{T_n}$, normalized time delay

The result for the unnormalized case is <u>not</u> independent of the time base, T_n, because the sequency and time are not connected by multiplication as with sine waves. This property of Walsh functions is used extensively with electromagnetic Walsh waves relative to the Doppler shift. Harmuth has suggested that the concepts of period of oscillation and wavelength be generalized for nonsinusoidal functions as Walsh functions but these concepts have no use in this thesis.

In summary, a sine function is described by its amplitude, frequency and phase angle,[20] while a Walsh function is described by its amplitude, sequency, time delay <u>and</u> time base.

[20]The phase angle could be considered as a time delay.

Chapter 2

WALSH FUNCTION GENERATOR

2.0 Introduction

One of the objectives of this thesis is the design of a device that will generate Walsh functions. Many designs have been discussed in the literature and are based on the various properties of Walsh functions. Some of these generators are programmable and are of true laboratory quality costing hundreds of dollars.

The design configuration chosen is one that is usually described in most of the literature concerning Walsh function generators. It generates sets of synchronized Walsh functions simultaneously.

This generator is used in some waveform synthesis experiments described in Chapter 4.

2.1 Description of Walsh Function Generator Configuration

The generator configuration used is based on the multiplication law as described in Chapter 1:

$$\mathrm{wal}(h \oplus k, \theta) = \mathrm{wal}(h, \theta)\mathrm{wal}(k, \theta) \qquad (2\text{-}1)$$

By applying this law to two previously generated Walsh functions, a new Walsh function can be generated. Hence, it would be possible to generate a limited set of Walsh functions.[1] A generator configuration

[1]See Reference 43, pp. 90-91.

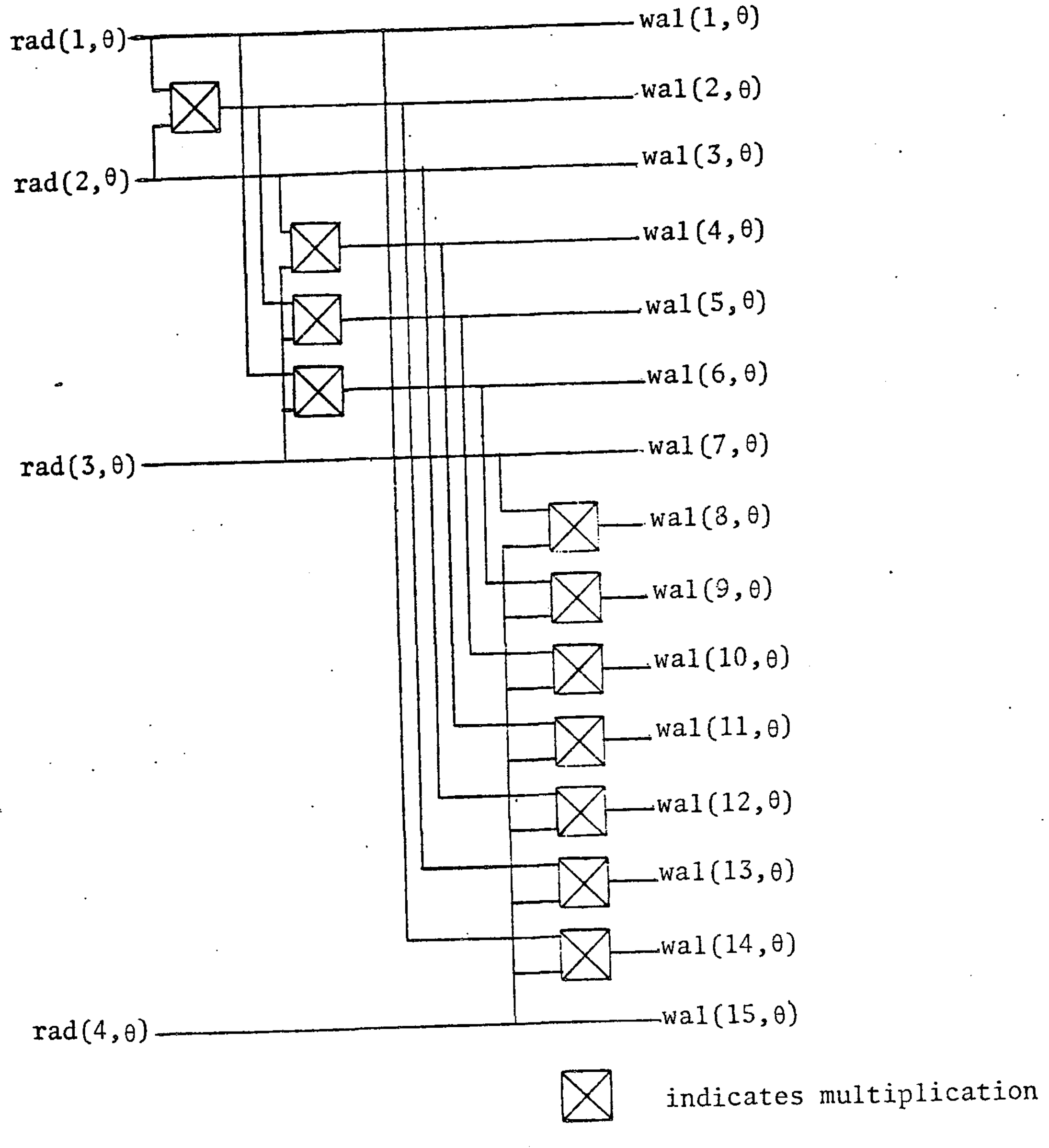


Figure 2-1 Analog Walsh Function Generator

based on this law for the first 15 Walsh functions is shown in Figure 2-1.[2,3]

From Figure 2-1 one sees that by using the functions wal(1,θ), wal(3,θ), wal(7,θ) and wal(15,θ), the rest of the Walsh functions can be generated directly or indirectly from previously generated functions. From Figure 2-2 through Figure 2-4, the generating set [wal(1,θ), wal(3,θ), wal(7,θ) and wal(15,θ)] are really the first four (non dc) Rademacher functions. As discussed in Chapter 1, Rademacher functions, a subset of Walsh functions, are just ordinary variable frequency square waves.

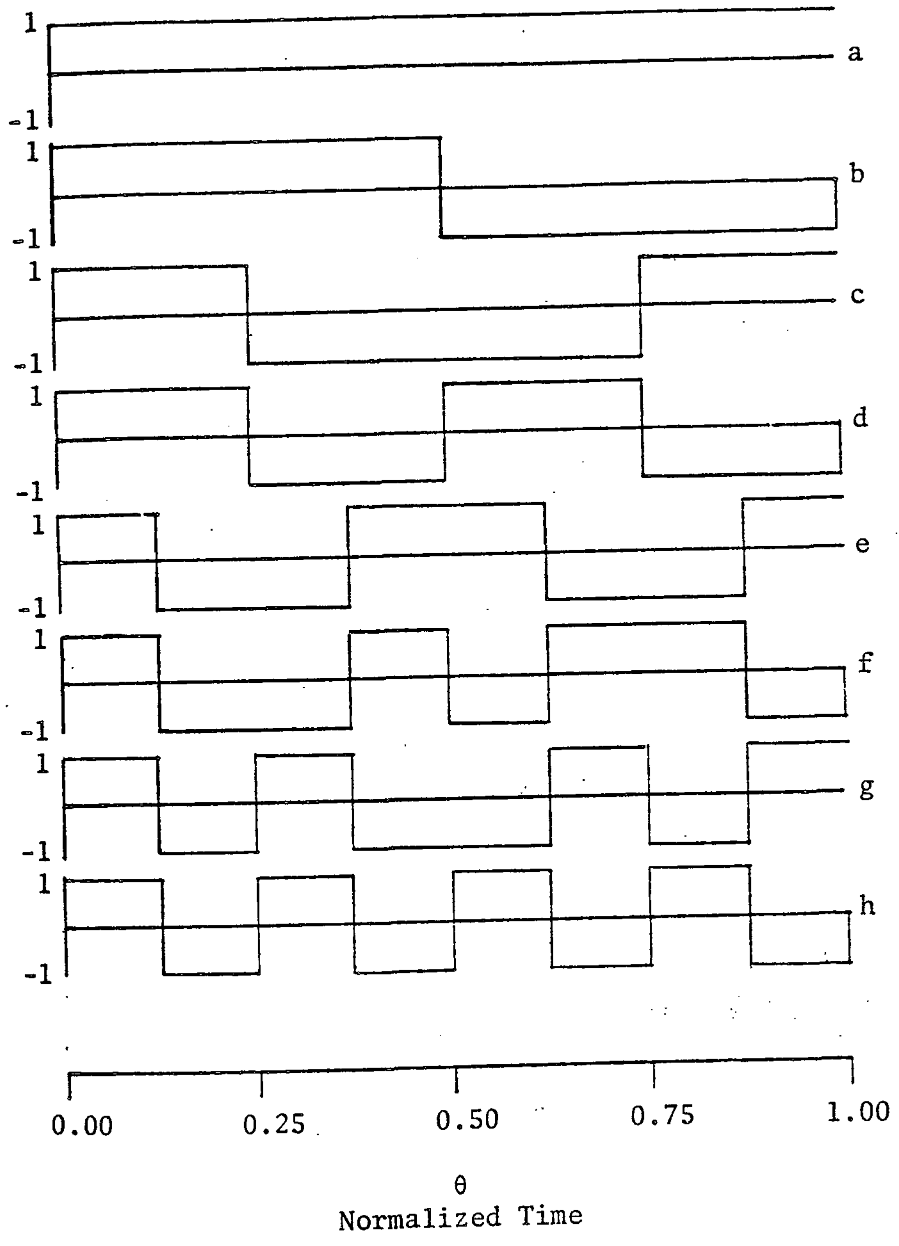
Hence, to generate a set of synchronized Walsh functions simultaneously, one must first generate a set of synchronized Rademacher functions simultaneously. Then by applying equation 2-1 as shown in Figure 2-1 the set of Walsh functions can be realized. To build a truly analog Walsh function generator capable of generating functions with levels between, say, ±1 volt, 11 analog multipliers would have to be obtained, not to mention the hardware needed to generate and phase-lock the analog Rademacher functions. Although this task is not impossible, there is a more efficient way to approach this problem, by binary coding the Walsh functions.

The first requirement of the generator of Figure 2-1 is that of obtaining the Rademacher functions. By examining Figure 2-4 one notes that each Rademacher function has a frequency that is twice the

---

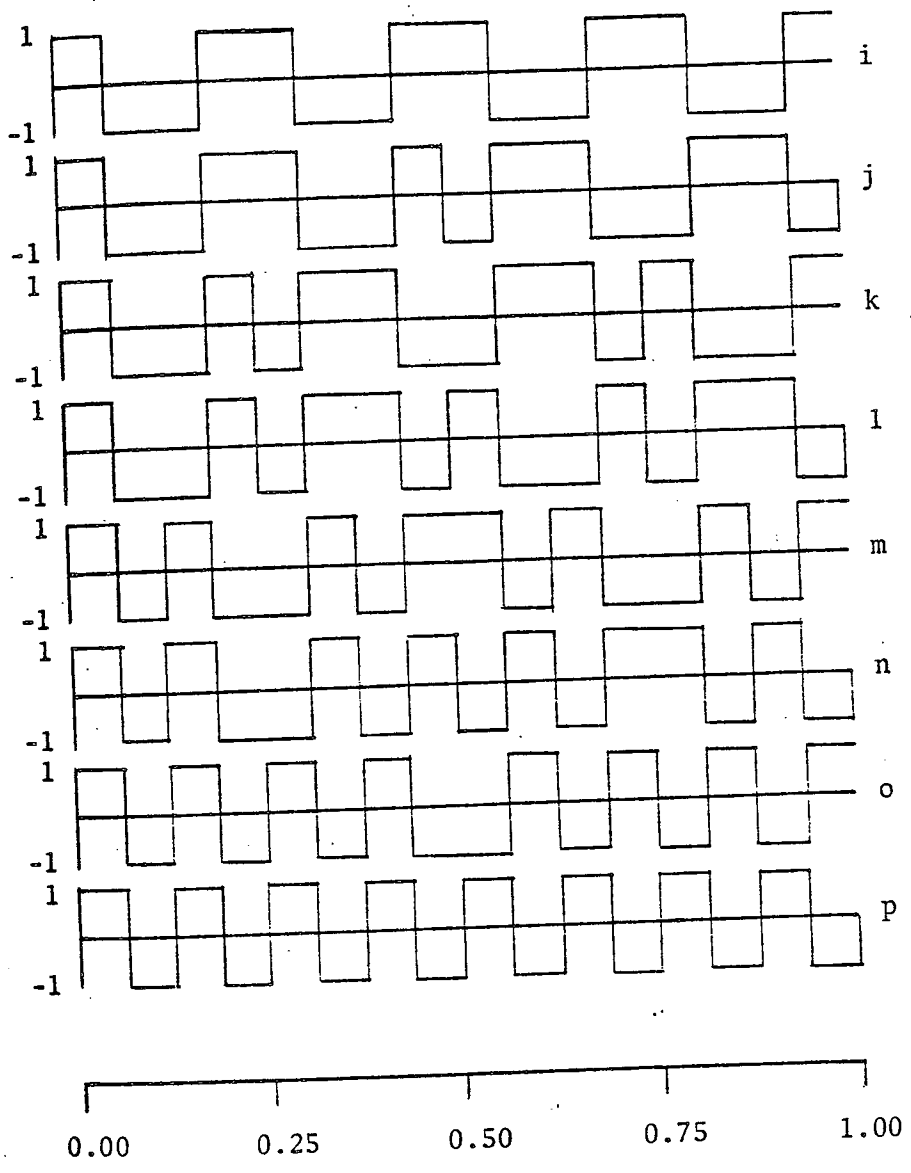
[2]The dc or wal(0,θ) function is not generated.

[3]The functions generated are periodically continued Walsh functions. See Section 1.3.



a	wal(0,θ)
b	wal(1,θ)
c	wal(2,θ)
d	wal(3,θ)
e	wal(4,θ)
f	wal(5,θ)
g	wal(6,θ)
h	wal(7,θ)

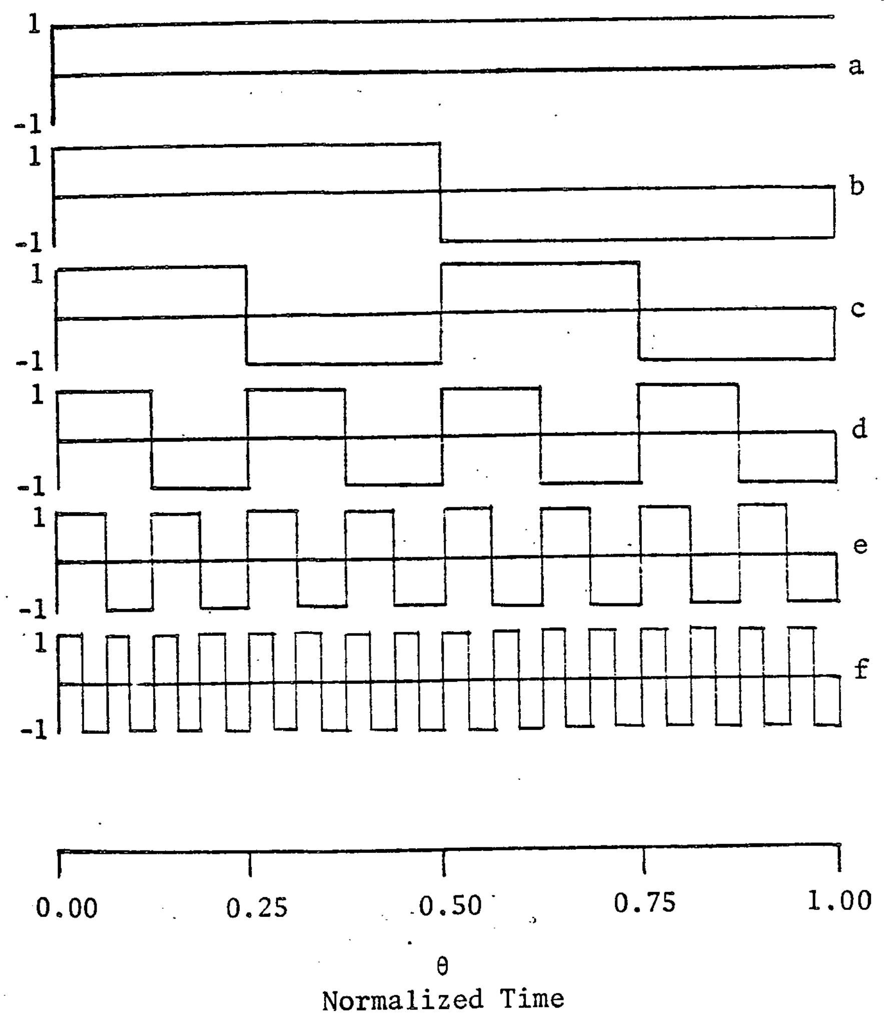
Figure 2-2 Walsh Functions



θ
Normalized Time

i	wal(8,θ)
j	wal(9,θ)
k	wal(10,θ)
l	wal(11,θ)
m	wal(12,θ)
n	wal(13,θ)
o	wal(14,θ)
p	wal(15,θ)

Figure 2-3 Walsh Functions (Continued)



a	rad(0,θ)
b	rad(1,θ)
c	rad(2,θ)
d	rad(3,θ)
e	rad(4,θ)
f	rad(5,θ)

Figure 2-4 The First Six Rademacher Functions

previous one. The functions themselves are two-valued, either 1 or -1[4]; hence they could be coded as binary functions. The +1 could be coded as logical 1 and the -1 coded as logical 0. This coding scheme results in what is known in the literature as logical Walsh functions, denoted WAL(n,θ), and logical Rademacher functions, denoted RAD(m,θ).

By using this coding scheme and the previously discussed frequency division property of Rademacher functions, it is possible to generate the needed Rademacher functions using a clock source and four toggle (T-type) flip-flops. The flip-flops have the property of dividing the frequency of the signal at its toggle input by two. Hence the output of a four-bit binary counter could be thought of as a generator for the first four logical Rademacher functions.

The next problem is what kind of logic element can be used to replace the multipliers of Figure 2-1? Since the Walsh functions of Figures 2-2 and 2-3 are assumed to be two valued functions, either 1 or -1, the multipliers of Figure 2-1 would have the "truth" table shown in Table 2-1. If the functions are coded according to the above scheme, the truth table for the "logical" multipliers are shown in Table 2-2. This is the truth table of the logic element known as an exclusive-NOR gate. Therefore the multiplication law of equation 2-1 reduces to:

$$\mathrm{WAL}(h \oplus k, \theta) = \overline{\mathrm{WAL}(h,\theta) \oplus \mathrm{WAL}(k,\theta)} \qquad (2\text{-}2)$$

for logical Walsh functions. Using equation 2-2, the analog Walsh function generator of Figure 2-1 reduces the logical Walsh function

[4]In a strictly mathematical sense this is not true. Walsh defined the functions to be identically zero at the zero crossings; however, by assuming two-valued functions there is no loss in generality.

TABLE 2-1

a	b	y
-1	-1	1
-1	1	-1
1	-1	-1
1	1	1

$y = ab$

"Truth" Table of Analog Multiplier

TABLE 2-2

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

$Y = \overline{A \oplus B}$

Truth Table of Exclusive-NOR Gate

generator of Figure 2-5. A clock signal is counted down to generate the logical Rademacher functions and the exclusive-NOR gates are used to operate on pairs of these functions to generate the Walsh functions.

This type of design could be used to generate larger sets of logical Walsh functions by using a larger counter (i.e., more flip-flops) and more exclusive-NOR gates.

2.2 Realization of Logical Walsh Function Generator

The logical Walsh generator of Figure 2-5 was built using standard transistor-transistor logic (TTL) integrated circuits. The first step of the design was to build a four-bit binary counter to generate the four logical Rademacher functions. This counter could have been built using four T-type flip-flops. This type of counter is known as an asynchronous or ripple counter. Ripple counters are susceptable to counting errors (commonly known as counting spikes or hazards) due to the fact that all flip-flops are not clocked simultaneously. Therefore, a synchronous counter was chosen over a ripple counter for the generator. The 74191 Synchronous Up/Down four-bit counter was chosen for the generator. The 74191 is a standard medium speed TTL integrated circuit. Since there is no standard TTL exclusive-NOR integrated circuit, the 7486, a quad two input exclusive OR gate was used. Two gates of each of the 7486's were hardwired as inverters. Therefore, the exclusive-NOR operation was obtained for the generator.

The generator was built using the TTL integrated circuits discussed above and is shown in Figure 2-6. A Hewlett-Packard function generator was used for the clock source. The outputs of the generator were connected to standard 9-pin plugs so that the generator could be

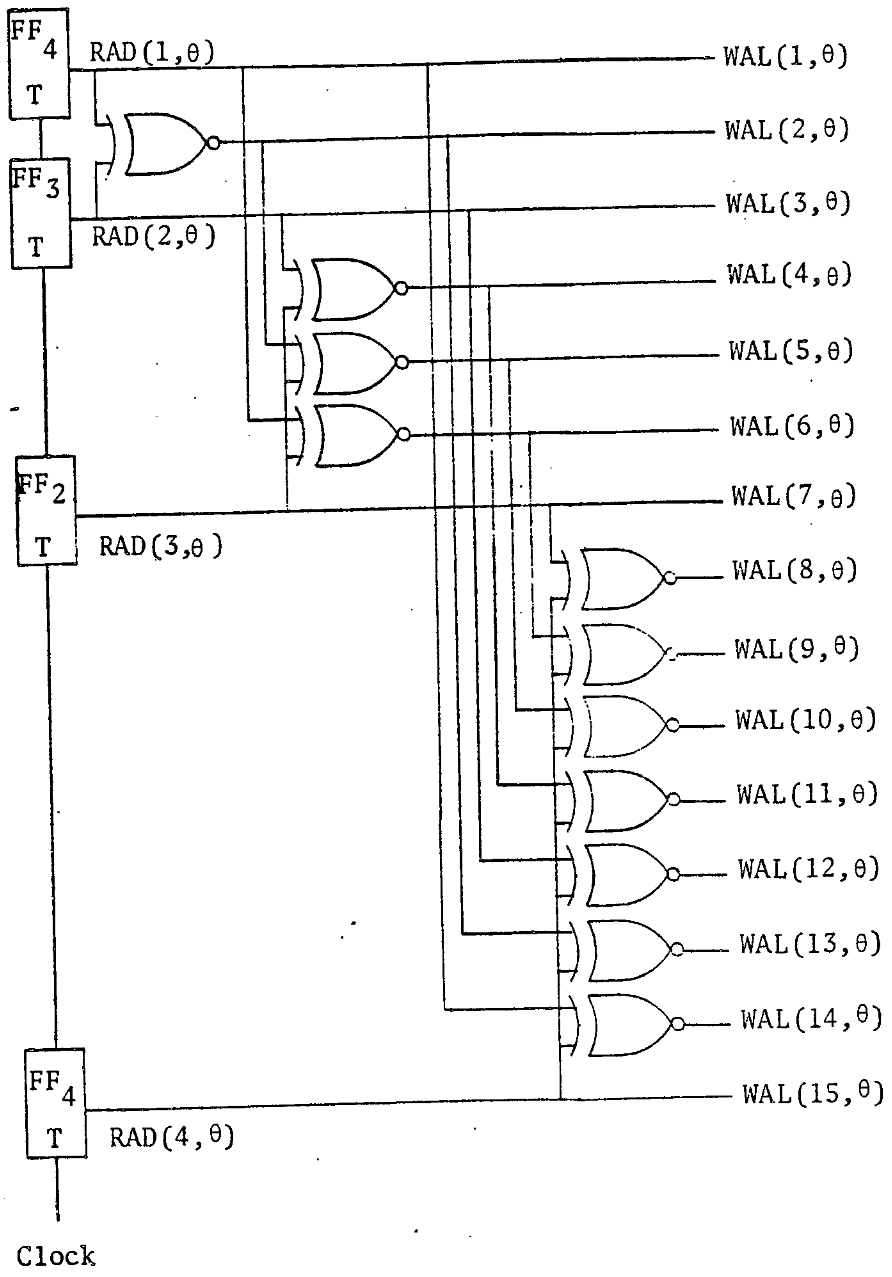


Figure 2-5 Logical Walsh Function Generator

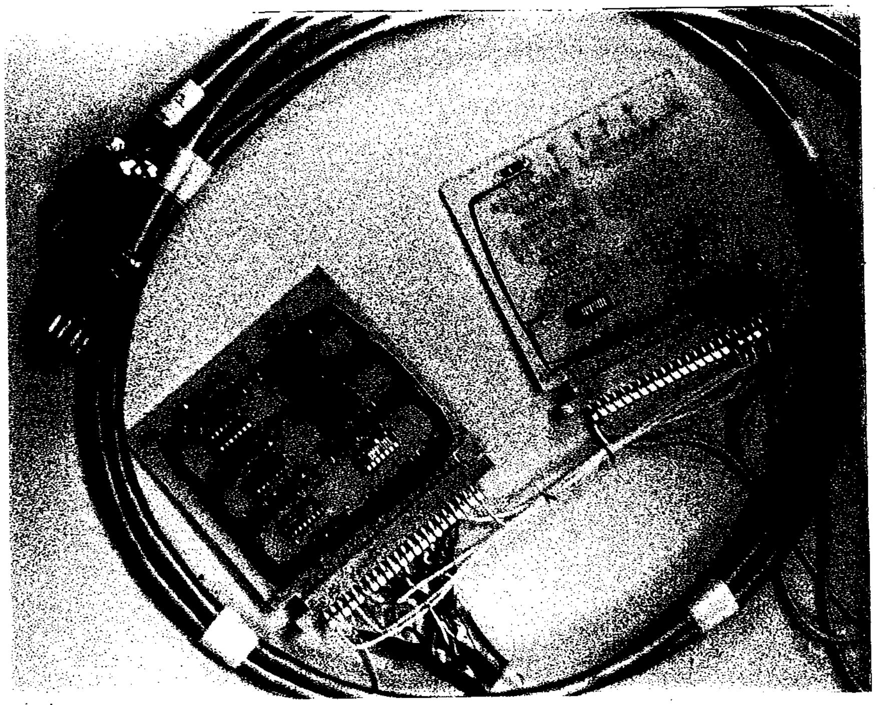


Figure 2-6 Photograph of Walsh Function Generator

connected to the analog trunk lines of the AD-4 analog computer. The outputs of the generator were then displayed on the Brush-8 strip chart recorder, as shown in Figures 2-7 through 2-9 and are identical to Walsh waveshapes shown in Figures 2-2 and 2-3.

By using a synchronous counter the only source of errors were the propogation delays of the exclusive NOR gates. This presented no real problem as long as the clock frequency was below 1 MHZ. The clock frequency for outputs shown in Figure 2-7 through Figure 2-9 was 1 HZ. The outputs were of course standard TTL waveforms; logical 1 is about 4.5 volts and logical 0 is at ground potential, assuming positive logic. No real differences were observed among the voltage levels between any two functions. These waveforms could be thought of as analog Walsh functions that are dc shifted and slightly amplified.

The beauty of this type of generator is that it is possible to generate 15 synchronized Walsh functions simultaneously. Think of the problems involved with generating 15 synchronized sine or cosine functions simultaneously! This type of design could be used to generate larger sets of Walsh functions and in fact Harmuth has suggested a variation on this design using a 20-bit binary counter and 19 exclusive-NOR gates capable of generating 1,048,576 different Walsh functions.[5]

One final comment should be made concerning the sequencies of the generator outputs. The normalized sequencies realized are the first eight integer sequency Walsh functions. The unnormalized sequency can

---

[5]Harmuth's design only generates one Walsh function at a time. See Reference 43, p. 91.

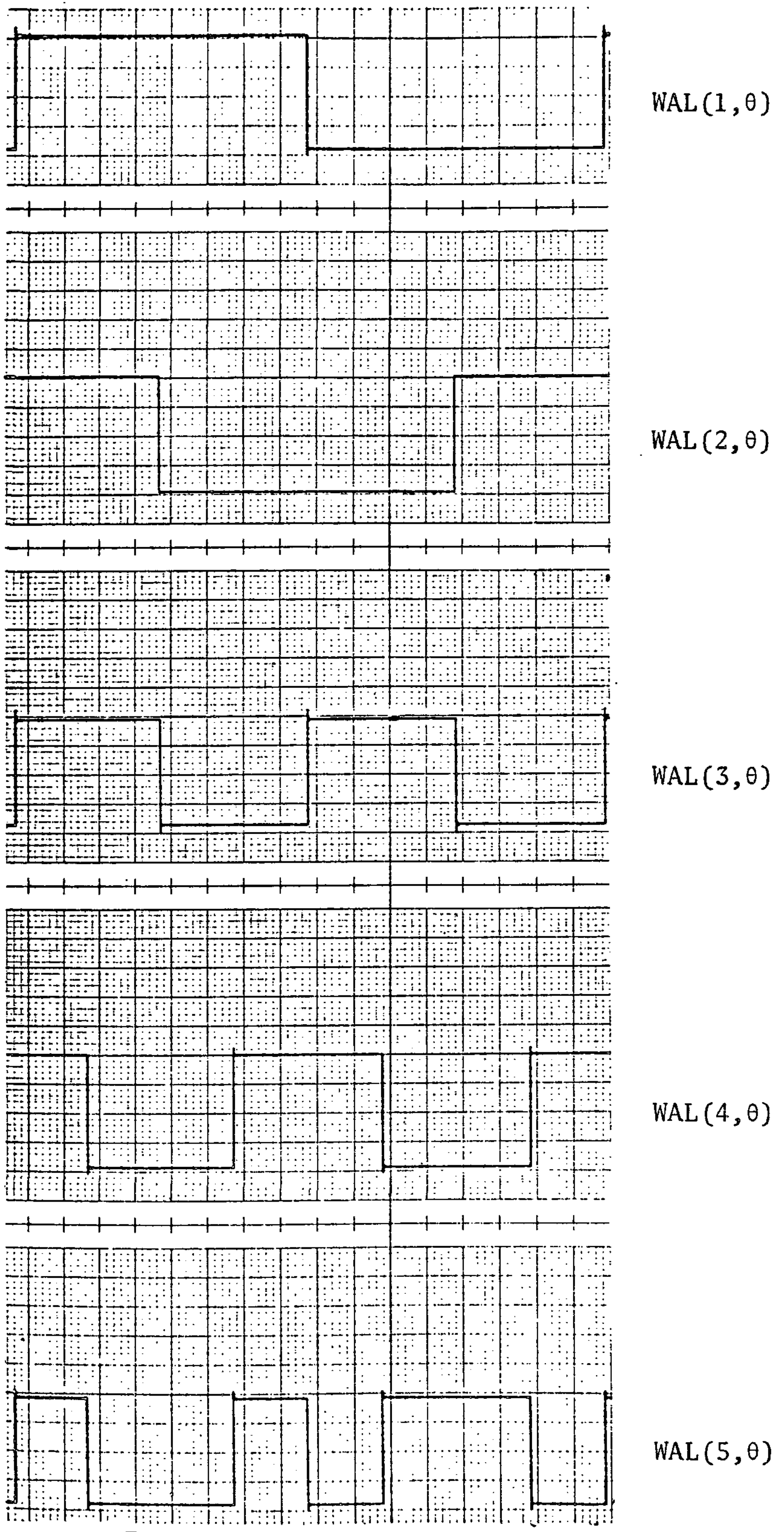


Figure 2-7 Output of Logical Walsh Function Generator

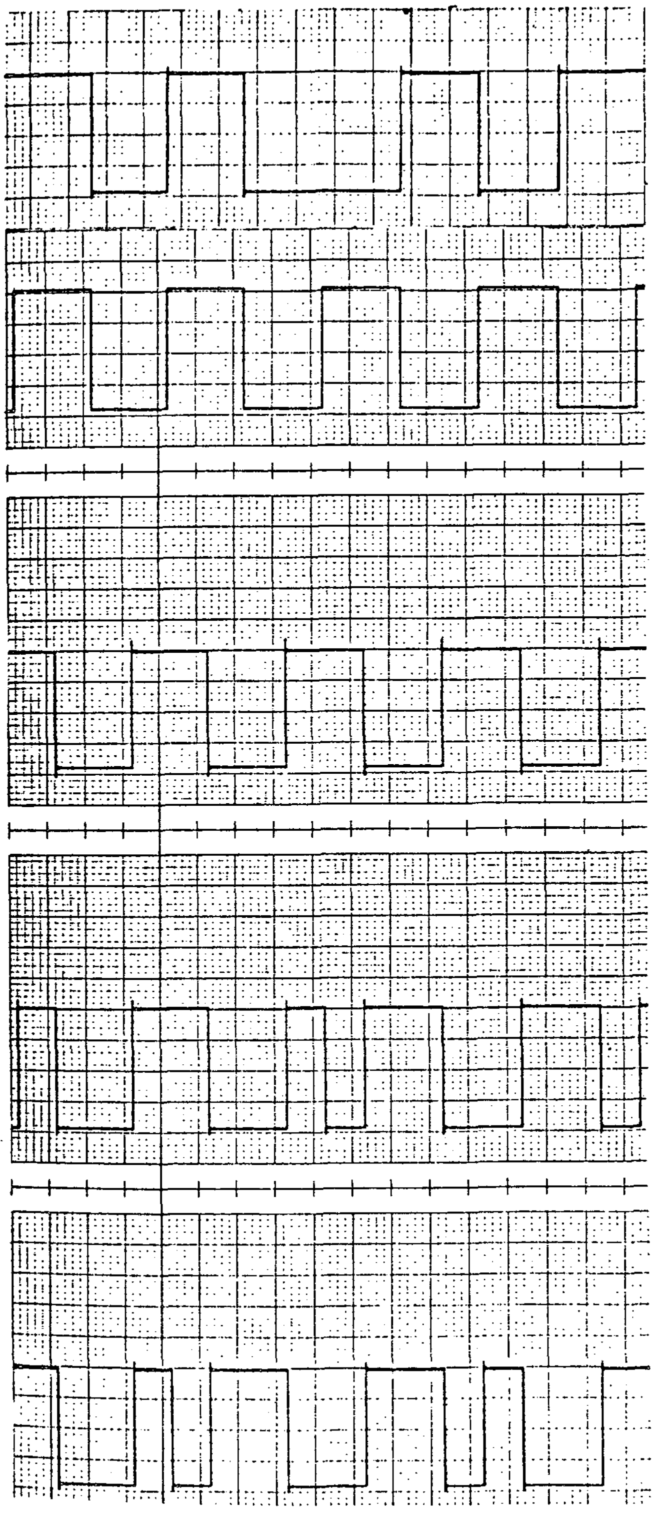


Figure 2-8 Output of Logical Walsh Function Generator

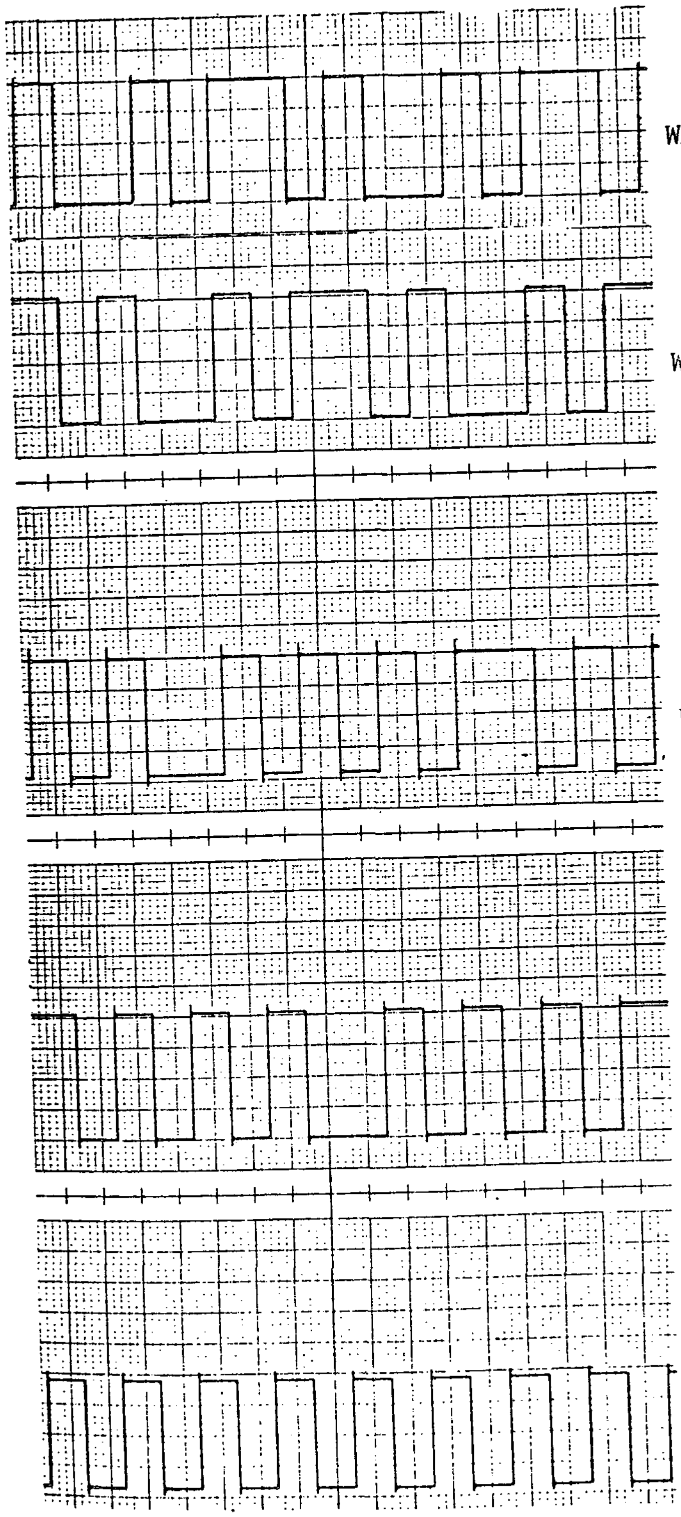


Figure 2-9 Output of Logical Walsh Function Generator

be related easily to the frequency of the clock, f_c. This can easily be seen by noting from Figure 2-2 and the discussion of Chapter 1 that the normalizing time base, T_n, is chosen so that the period of WAL(1,θ) is equal to T_n. Hence, using this fact and the fact that each successive stage frequency of the counter is the previous stage frequency divided by two. The time base is related to the clock frequency by:

$$T_n = 2^m T_c \tag{2-3}$$

where m = number of bits of the binary counter

$T_c = \frac{1}{f_c}$

f_c = clock frequency, Hz

the unnormalized sequencies remain:

$$\phi = \frac{\mu}{T_n} \tag{2-4}$$

Therefore by changing the clock frequency the unnormalized sequencies change accordingly but the normalized sequencies remain the same.

[6]For the generator of Figure 2-5 m = 4.

Chapter 3

DISCRETE WALSH TRANSFORM

3.0 Introduction

The purpose of this chapter is to describe the Finite Discrete Walsh-Fourier Transform[1] (DWT). The DWT will be compared to the Finite Discrete Fourier Transform[2] (DFT) and a computer program will be described that computes the DWT. The discrete dyadic convolution property of the DWT will be demonstrated and briefly discussed. The three types of ordering of Walsh functions will be described as an introduction to the DWT.

3.1 Walsh Function Ordering

The ordering of the Walsh functions, as introduced in Chapter 1, is not the only possible ordering that has appeared in the literature.[3] Two other types of ordering known as dyadic ordering and Hadamard ordering are discussed in the literature.

The ordering discussed in Chapter 1 is known as Walsh or sequency ordering. By ordering the functions such that each Walsh function, denoted wal(i,θ), has one more zero crossing in the interval $(0,1)$ than the previous function, the concept of sequency can be introduced

[1]In the literature, this is sometimes referred to as the Finite Walsh Transform, or the Discrete Walsh Transform.

[2]In the literature this is sometimes referred to as the Finite Fourier Transform, or the Discrete Fourier Transform.

[3]See Reference 8.

as a generalization of frequency. The concept of sequency can then be used by the engineer in ways similar to using frequency when working with sine-cosine functions. The sequency-ordered Walsh functions are shown in Figure 3-1.

The dyadic type of ordering is due to Paley[4] who first pointed out the existence of such an ordering. Walsh functions ordered this way are sometimes referred to as Paley-ordered Walsh functions. The Paley-ordered Walsh functions, as shown in Figure 3-2, are denoted by $wal_p(i,\theta)$ where p denotes Paley ordering.

Paley and Fine[5] used this type of ordering in their work concerning Walsh functions. As shall be discussed later, this type of ordering appears automatically when using the Discrete Walsh Transform. Yuen[6] has presented a strong case that dyadic ordering is superior to sequency ordering when using the DWT.

This set of functions is related to the sequency-ordered functions by the relation:

$$wal_p(i,\theta) = wal(b(i),\theta) \qquad (3\text{-}1)$$

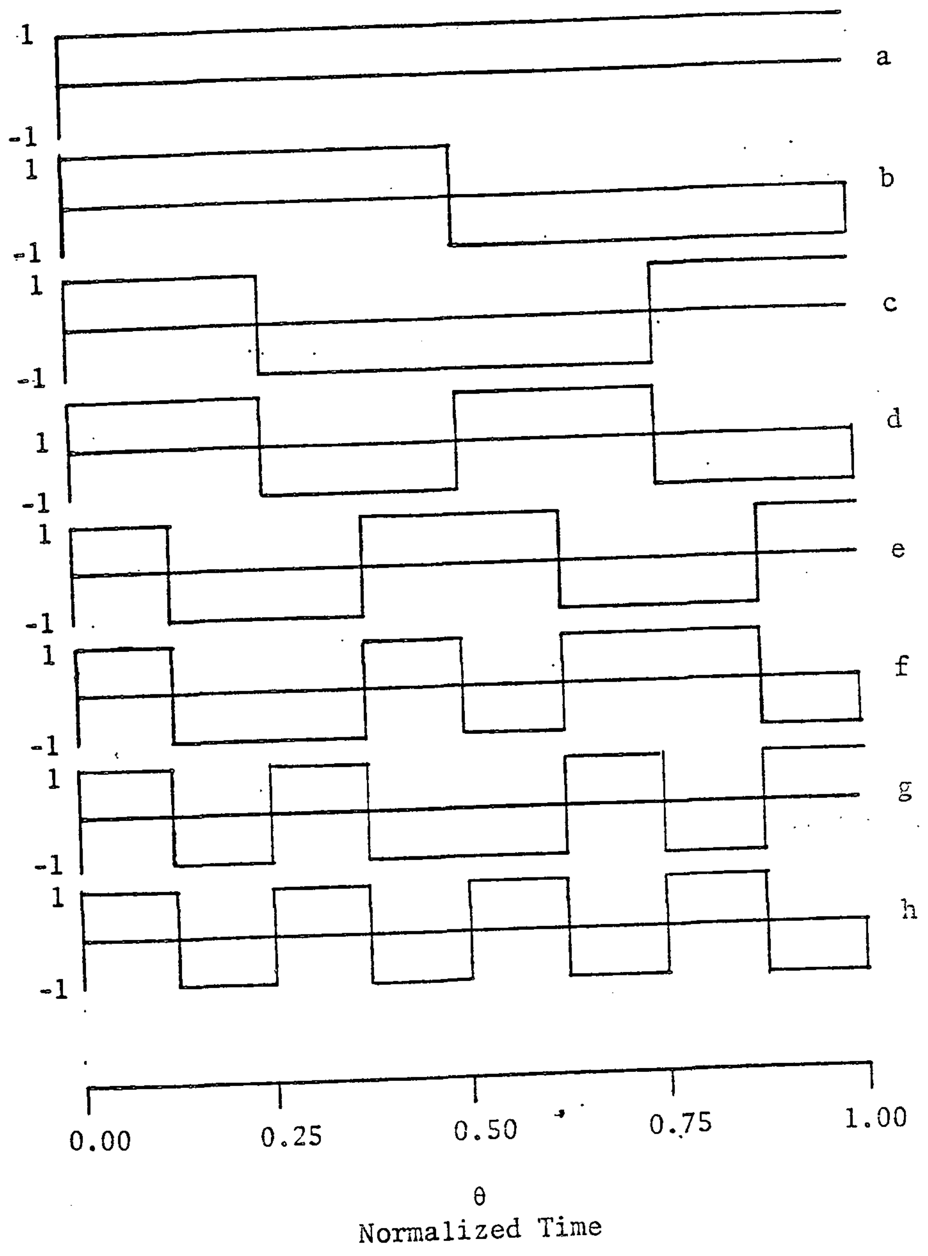
where b(i) is the Gray code-to-binary conversion of i.

The third type of ordering is known as Hadamard or natural ordering. The Hadamard ordered Walsh functions are shown in Figure 3-3 and are denoted as $wal_H(i,\theta)$. This type of ordering arises because of

[4]See Reference 66.

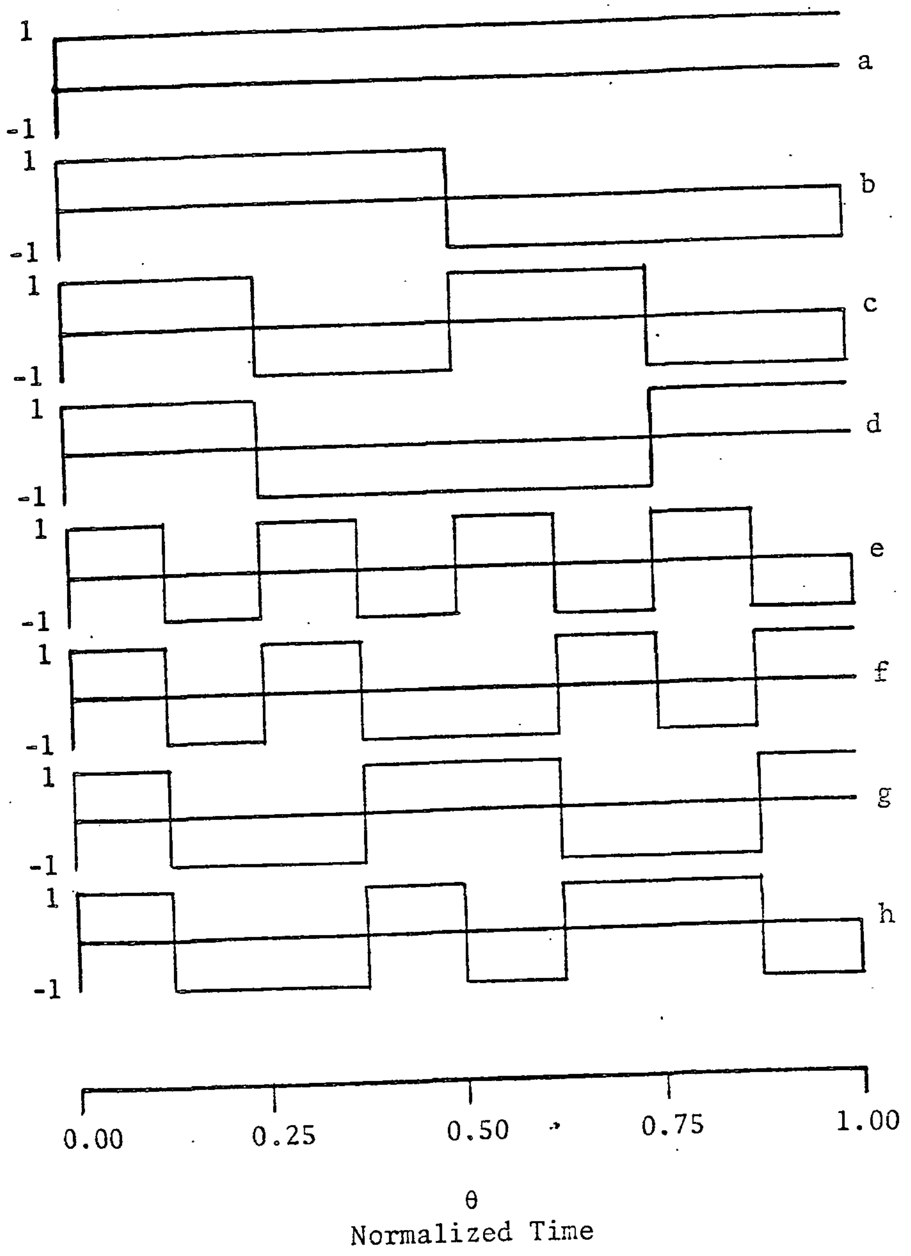
[5]See Reference 34.

[6]See Reference 93 and 94.



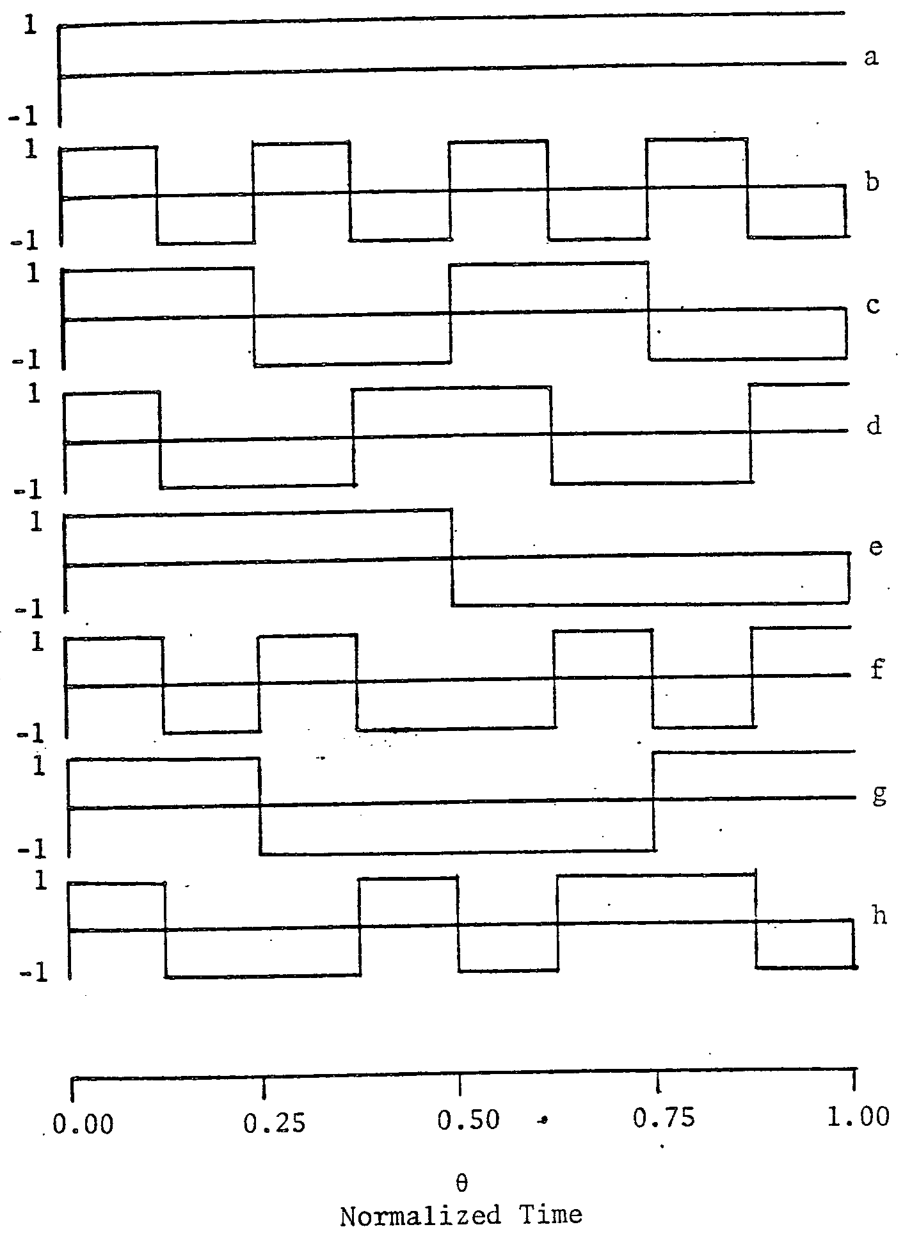
a	wal(0,θ)
b	wal(1,θ)
c	wal(2,θ)
d	wal(3,θ)
e	wal(4,θ)
f	wal(5,θ)
g	wal(6,θ)
h	wal(7,θ)

Figure 3-1 The First Eight Sequency Ordered Walsh Functions



a	$wal_p(0,\theta)$	$[wal(0,\theta)]$
b	$wal_p(1,\theta)$	$[wal(1,\theta)]$
c	$wal_p(2,\theta)$	$[wal(3,\theta)]$
d	$wal_p(3,\theta)$	$[wal(2,\theta)]$
e	$wal_p(4,\theta)$	$[wal(7,\theta)]$
f	$wal_p(5,\theta)$	$[wal(6,\theta)]$
g	$wal_p(6,\theta)$	$[wal(4,\theta)]$
h	$wal_p(7,\theta)$	$[wal(5,\theta)]$

Figure 3-2 The First Eight Paley Ordered Walsh Functions



a	$wal_H(0,\theta)$	$[wal(0,\theta)]$
b	$wal_H(1,\theta)$	$[wal(7,\theta)]$
c	$wal_H(2,\theta)$	$[wal(3,\theta)]$
d	$wal_H(3,\theta)$	$[wal(4,\theta)]$
e	$wal_H(4,\theta)$	$[wal(1,\theta)]$
f	$wal_H(5,\theta)$	$[wal(6,\theta)]$
g	$wal_H(6,\theta)$	$[wal(2,\theta)]$
h	$wal_H(7,\theta)$	$[wal(5,\theta)]$

Figure 3-3 The First Eight Hadamard Ordered Walsh Functions

the similarities between a sampled set of these functions and the Hadamard matrix of linear algebra.[7]

This set of functions is related to the sequency-ordered functions by the relation:

$$\mathrm{wal}_H(i,\theta) = \mathrm{wal}(b(j),\theta) \qquad (3\text{-}2)$$

where j is obtained by the bit-reversal of i and b(j) is the Gray code-to-binary conversion of j.

Of all three types of ordering, only the sequency ordering allows the introduction of sal and cal functions. All three types of ordering will be used when discussing the DWT.

3.2 Discrete Fourier Transform

For many years engineers have been interested in computing Fourier series or Fourier transforms using the digital computer. This has presented some problems in that the digital computer can handle only a finite number of discrete points. Since only a finite number of points can be handled, computing a true Fourier transform is out of the question. Also, since only discrete points can be evaluated, a true Fourier series cannot be computed either. In 1965, Cooley and Tukey[8] "rediscovered" a method that can be used in certain cases to approximate very accurately Fourier series and Fourier transforms. Cooley and Tukey's initial work lead to the vigorous definition of what has become known as the Discrete Fourier Transform (DFT)[9].

[7]See References 40 and 41.

[8]See Reference 28.

[9]The Discrete Fourier Transform is very closely related to a truncated Fourier series. For an excellent treatment of the DFT see References 26, 27 and 11.

Let X(i), i = 0,1,...., N-1 be a sequence of N finite valued complex numbers. The DFT of X(i) is defined as:

$$A_F(n) = \frac{1}{N}\sum_{i=0}^{N-1} X(i)e^{-2\pi inj/N} \tag{3-3a}$$

where $j = \sqrt{-1}$ and $A_F(n)$ is known as the discrete Fourier coefficients of X(i). The inverse DFT is defined as:

$$X(i) = \sum_{n=0}^{N-1} A_F(n)\ e^{2\pi inj/N} \tag{3-3b}$$

Brigham[10] has shown that equation 3-3 is a transform pair possessing many of the properties related to standard Fourier transforms and Fourier series. The most important property of the DFT is the discrete convolution theorem to be discussed later.

The sequence X(i) used in equation 3-3 can be thought of as an ideally (impulse) sampled time signal. The sequence $A_F(n)$ can be thought of as frequency coefficients of the time signal X(i).[11] The real part of $A_F(n)$ corresponds to the discrete cosine transform of X(i) and the imaginary part of $A_F(n)$ corresponds to the discrete sine transform. If the above interpretation of X(i) is used, one may ask why the time variable and frequency variable does not show up explicitly in equation 3-3. The reason is that for convenience the time and frequency variables are assumed to be normalized such that the N samples of the time function are described on the unit interval. Therefore, the frequency coefficients are defined for normalized

[10]See Reference 16, pp. 91-130.

[11]Other interpretations are possible since X(i) contains, in general, complex numbers. One possible interpretation is that of X(i) being

frequencies of 0 to N-1. These values can be easily unnormalized as discussed in Chapter 1 when the sampling period is known.[12]

Theilheimer[13] has shown that equation 3-3 can be written in matrix formulation:

$$\overline{a_F} = \frac{1}{N} [F] \overline{x} \tag{3-4a}$$

$$\overline{x} = [F]^{-1}\overline{a_F} \tag{3-4b}$$

where $\overline{x}$ and $\overline{a_F}$ are N dimensional column vectors corresponding to the time function X(i) and the Fourier coefficients $A_F(n)$ respectively. [F] is an NxN matrix known as the DFT matrix.[14,15] The [F] matrix is a complex-valued matrix and can be thought of as containing essentially sampled sine and cosine functions.[16]

To evaluate the DFT of a given time function $\overline{x}$ one notes, from equation 3-4a, that N^2 complex additions and complex multiplications

---

a complex-valued time function. This type of interpretation will not be possible for the Discrete Walsh Transform.

[12]See Reference 16, pp. 91-99.

[13]See Reference 89.

[14]In the literature, the DFT matrix is usually represented as [W]. This notation will not be used to avoid confusion with the DWT matrix to be developed later.

[15]Equation 3-4 is recognized as a true matrix orthogonal transformation. This was not obvious from equation 3-3.

[16]See Reference 16, pp. 148-149.

are required. Cooley and Tukey[17] have shown that the number of operations (complex additions and multiplications) needed to complete the DFT are greatly reduced if the number of points to be transformed (N) are some power of two, i.e., $N=2^m$ where m is an integer. This algorithm for computing the DFT has become known as the Fast Fourier Transform (FFT). Cooley and Tukey showed that the number of operations needed to compute the DWT, using the FFT is $N \log_2 N$ where $N=2^m$. For relatively large numbers of points, the number of computations is greatly reduced. Glassman[18] has generalized the FFT such that the number of points does not need to be of radix two but can take on other values. A FORTRAN subroutine, FFT1, that computes a forward and inverse FFT appears in the Appendix. As mentioned at the beginning of this section, the DFT in some cases gives only an approximation to the true Fourier coefficients of a particular time function;[19], therefore one should not blindly use the FFT.

3.3 Discrete Walsh Transform

Based upon the work presented in section 3.2, the Discrete Walsh Transform[20] will now be discussed.

Let X(i), i = 0,1,.... N-1, be a sequence of N finite valued real numbers. The DWT of X(i) is defined as:

---

[17]See Reference 28.

[18]See Reference 38.

[19]For a more detailed discussion of how the FFT is used, see References 26, 27, 29 and 16.

[20]The Discrete Walsh Transform is related very closely to a truncated Walsh-Fourier series. For an excellent presentation of the DWT and

$$A_W(n) = \frac{1}{N} \sum_{i=0}^{N-1} X(i)\,\mathrm{wal}(n,i) \qquad (3\text{-}5a)$$

where $A_W(n)$ is known as the discrete Walsh-Fourier coefficients of X(i). The inverse DWT is defined as:

$$X(i) = \sum_{n=0}^{N-1} A_W(n)\,\mathrm{wal}(n,i) \qquad (3\text{-}5b)$$

Equation 3-5 also form a transform pair possessing many of the properties related to Walsh-Fourier transforms and series.[21] Since the Walsh function used above are sequency-ordered, equation 3-5 is known as the sequency-ordered DWT.

The sequence X(i) used in equation 3-5 can also be thought of as an ideally sampled time function; however, the sequence $A_W(n)$ cannot be thought of as exactly the sequency coefficients of X(i) because the functions used in the transform pair are not sal and cal functions but the wal function. Equation 3-5 could be rewritten to force the sal and cal notation to appear. However, by doing this, the transform equations would not lend themselves to a fast algorithm similar to the Cooley-Tukey FFT. Therefore, one must remember when using the sequency-ordered DWT that the transform coefficients, $A_W(n)$, correspond to wal coefficients and not sal and cal coefficients, e.g. $A_W(2)$ corresponds to the coefficient of wal(2,i) or cal(1,i). The time and sequency variable are also assumed to be normalized.[22]

---

how it is related to the DFT, see References 52, 56 and 84.

[21]Ibid.

[22]The θ notation was not used for normalized time for either the DFT or the DWT because time (or normalized time) does not appear explicitly in either set of equations. This is the standard approach used in the literature.

The DWT can be also defined using the other orderings of Walsh functions.[23] The Paley ordered DWT is defined as:

$$A_{W_p}(n) = \frac{1}{N} \sum_{i=0}^{N-1} X(i) \mathrm{wal}_p(n,i) \tag{3-6a}$$

$$X(i) = \sum_{n=0}^{N-1} A_{W_p}(n) \mathrm{wal}_p(n,i) \tag{3-6b}$$

where equation 3-6b is the inverse transform. The Hadamard-ordered DWT[24] is defined as:

$$A_{W_H}(n) = \frac{1}{N} \sum_{i=0}^{N-1} X(i) \mathrm{wal}_H(n,i) \tag{3-7a}$$

$$X(i) = \sum_{n=0}^{N-1} A_{W_H}(n) \mathrm{wal}_H(n,i) \tag{3-7b}$$

where equation 3-7b is the inverse transform. The coefficient $A_W(n)$ and $A_{W_H}(n)$ can be related by the bit-reversal and Gray code conversion discussed in section 3.1. One should note that by transforming the same time function, X(i), using all three types of DWT's the values of the resultant transform coefficient will be identical but they will be shuffled in different order relative to each other.

A matrix formulation for equation 3-5 through 3-7 is:

$$\overline{a_W} = \frac{1}{N} [W]\overline{x} \tag{3-8a}$$

---

[23]Other types of discrete transforms have been discussed in the literature. The generalized discrete-Fourier transform is due to Ahmed, et. al.and it is similar to the generalized Fourier series and transforms discussed in Chapter 1. See References 4 and 6.

[24]The Hadamard-ordered DWT is sometimes referred to as the Hadamard transform or BIFORE transform. See Reference 7.

$$\bar{x} = [W]^{-1}\bar{a}_W \qquad (3\text{-}8b)$$

$$\bar{a}_{W_P} = \frac{1}{N}[W_P]\bar{x} \qquad (3\text{-}9a)$$

$$\bar{x} = [W_P]^{-1}\bar{a}_{W_P} \qquad (3\text{-}9b)$$

$$\bar{a}_{W_H} = \frac{1}{N}[W_H]\bar{x} \qquad (3\text{-}10a)$$

$$\bar{x} = [W_H]^{-1}\bar{a}_{W_H} \qquad (3\text{-}10b)$$

where $\bar{x}$ and $\bar{a}_W$ are N-dimensional column vectors corresponding to the sampled time function X(i) and the correctly ordered Walsh coefficients respectively. [W] is an NxW matrix known as the DWT matrix or the Walsh matrix. The matrix [W] is a real-valued matrix and can be thought of as containing sampled Walsh functions. The three types of Walsh matrices are shown in Figures 3-4 through 3-6 respectively. One should note the similarity to the Walsh functions of Figures 3-1 through 3-3.

From equation 3-8 through 3-10 it is obvious that it requires N^2 additions and multiplications to evaluate the DWT of a given function. Upon closer inspection of the Walsh matrix, the number of operations required to evaluate the transform is really N^2 additions (or subtractions) since the matrix contain only 1's and -1's. The DWT requires fewer operations than the DFT, which requires N^2 complex operations (complex addition and complex multiplication). The DWT is therefore simpler to compute (i.e., faster) than the DFT.

Shanks[25] has shown that it is possible to evaluate the DWT using a Cooley-Tukey type of algorithm such that the number of operations

[25]See Reference 83.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

Figure 3-4 The Sequency Ordered Walsh Matrix [W]

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Figure 3-5 The Paley Ordered Walsh Matrix $[W_P]$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Figure 3-6 The Hadamard Ordered Walsh Matrix $[W_H]$

(additions) required is $N\log_2 N$ where $N = 2^m$. This algorithm for computing the DWT has become known as the Fast Walsh Transform (FWT). The FWT is "faster" than the FFT since only addition is required for computing the FWT.

The actual derivation of the FWT algorithm will not be presented. However, methods that can be used to modify an existing FFT computer program will be discussed in the next section. The derivation of FWT algorithm is very "similar" to the classical derivation initially presented by Cooley and Tukey. The reader is referred to Shanks for more details.[26]

3.4 Programming the Fast Walsh Transform

One of the objectives of this thesis is to describe a package of computer programs that can be used to evaluate the Discrete Walsh Transform. The computer programs are listed in the Appendix along with full documentation on how to use them. The procedures used in obtaining these programs are discussed below.

Of the three types of FWT's discussed in section 3.3, the Hadamard ordered FWT was programmed using a slightly different algorithm than the standard Cooley-Tukey algorithm.[27] This FWT program is due to Shum and Elliott[28] who initially used this type of ordering for speech processing work. Their algorithm is based on the factoring of the W_H matrix

[26]Ibid.

[27]Hadamard-ordered FWT is sometimes referred to in the literature as the Fast Hadamard Transform (FHT).

[28]See References 84 and 85.

using the Kronecker product[29] operation of linear algebra. This algorithm requires the same number of operations ($N\log_2 N$) as the Cooley-Tukey algorithm; however the approach is somewhat different.[30] The disadvantages of this algorithm are that it does not arrive easily from a modification of an existing FFT program, and that the inverse transformation is not as easily accomplished.

Two FORTRAN subroutines, FWT1 and FWT2, were obtained that compute the Hadamard ordered FWT based on Shum and Elliott's algorithm. One should remember that the coefficients obtained using these subroutines are the Hadamard-ordered coefficients and do lend themselves easily to the concept of sequency.

Computer programs for the other two types of ordering were written by modifying an existing FFT subroutine. The modifications are due to Shanks and Manz.[31] The FORTRAN subroutine, FFT1, was modified to compute the Paley ordered and sequency ordered FWT.

The Paley-ordered modification involves just setting all the trigonometric values to be 1.0 + j0.0 in the FFT program. Since the FWT is defined for only real-valued time signals, all calculations involving the imaginary part of the FFT program can be removed. The subroutine FFT1 was modified accordingly and renamed FWT3. Again, one

---

[29]The Kronecker product is sometimes referred to as the tensor product.

[30]Glassman has shown that the Cooley-Tukey algorithm is really based on the Kronecker product operation. In fact, the Cooley-Tukey algorithm is one of the many fast algorithms that can be used to evaluate discrete transforms. See References 38 and 6.

[31]See References 83 and 61.

should remember the transform coefficients obtained from FWT3 are Paley-ordered and not sequency ordered.

The sequency-ordered modification was suggested by Manz and it involves modifications to a Paley-ordered FWT. The modifications involve inverting the sign of every other dual node pair[32] in the transform program. The subroutine FWT3 was modified using this scheme and renamed FWT4. This subroutine performs a sequency-ordered transform and will be discussed later.

All five of the transform programs discussed above perform their operations "in place", i.e. extra array space is not needed to store intermediate calculations. The Hadamard-ordered and Paley-ordered transform programs were written to demonstrate the feasibility of performing these transformations on a digital computer. These two types of transforms will not be used for the experimental work to be discussed in Chapter 4 since they do not compute sequency-ordered coefficients. It would be necessary to reshuffle the output coefficients to sequency order them which would require more computation time. These subroutines could be used by others for different applications of Walsh functions.[33]

---

[32]For a discussion of the dual node concept, see Reference 16, pp. 154-156.

[33]Pitassi has used the Paley-ordered FWT to develop a fast algorithm for discrete arithmetic convolution where the number of convolved points is less than 1024. See Reference 72.

3.5 Miscellaneous Walsh Function Programs[34]

Hutchins[35] has suggested an algorithm for obtaining the sequency-ordered Walsh matrix [W] (see equation 3-8 and Figure 3-4). This algorithm is based on the recursive relationship for Rademacher functions (see equation 1-13). A FORTRAN subroutine, called WALSH, was written based on this algorithm and is listed in the Appendix. WALSH will construct a Walsh matrix of any order that is a power of two, i.e. $N = 2^m$. This subroutine was used initially to check out the FWT subroutines by actually performing the matrix calculations of equations 3-8 through 3-10.

An algorithm due to Bhagavan and Polge[36] was used to write a program that would shuffle the Hadamard-ordered transform coefficients so that they would be in sequency order. This shuffling operation performs the bit-reversals and Gray code conversions discussed in section 3.1. The subroutine SORT1 is based on this alborithm; the shuffling is performed "in place". One notes from examining SORT1 that the shuffling operation takes extra time to obtain the sequency-ordered coefficients.

3.6 Comparison of the Interpretation of the Fast Fourier Transform With The Fast Walsh Transform

From section 3.3, one observes that the FWT is "faster" than the FFT

[34]These programs were written very early in the research for this thesis and are discussed here for general information purposes only.

[35]See Reference 48.

[36]See Reference 14.

due to the fact that the FWT requires no multiplications to compute the transform.[37] However, there are more differences in interpreting the output coefficients of the FWT that one should be aware of. The differences discussed below pertain to the subroutine FWT4 specifically and to the other subroutines in general.

The output coefficients of the Fast Fourier Transform, denote $\overline{a}_F$, are in general complex numbers representing the normalized frequency coefficients of the time function $\overline{\overline{x}}$[38]. The real part of $\overline{a}_F$ corresponds

[37]To illustrate the relative speed of the FWT, the times required to compute a 1024 point transform using the subroutines FWT4 and FFT1 were observed on the IBM 1130 computing system. The Fast Walsh Transform took 14 seconds to compute; the Fast Fourier Transform took 80 seconds to compute the same 1024 point transform. This claim is made with the knowledge that the FFT subroutine used (FFT1) is not the "fastest" FFT subroutine available. The subroutine FFT1 does not use "Twiddle Factors" to increase its computing speed nor is the subroutine "pruned" or "blocked". The speed claim is made on the basis that the subroutine FWT4 is a <u>modified</u> version of FFT1 and hence it too could be made faster using some of the techniques discussed above. For a discussion on how to make the FFT faster, see Reference 16, pp. 184-197 and Reference 26.

[38]The matrix version of the DFT and DWT will be used in this discussion; see equations 3-4a and 3-8a. One should not lose sight of the fact that the FFT and FWT are just "fast" procedures used to calculate the DFT and DWT respectively.

to the discrete cosine transform and the imaginary part corresponds to the discrete sine transform respectively. By computing an N point transform one only resolves $\frac{N}{2}$ frequency coefficients; i.e., the dc coefficient and the first $\frac{N}{2}$-1 frequency coefficients. The output coefficients are ordered such that the first $\frac{N}{2}$ terms of $\bar{a}_F$ corresponds to the first $\frac{N}{2}$ Fourier coefficient and the last $\frac{N}{2}$ term of $\bar{a}_F$ corresponds to the "folded" frequency spectrum. Essentially the Nyquist folding phenomena is observed in the output coefficients[39]. The important point to be remembered from this discussion is that by computing an N point transform only the first $\frac{N}{2}$ Fourier cosine and sine coefficients are obtained[40].

The output coefficients of the Fast Walsh Transform, denoted $\bar{a}_W$, are real numbers that represent the normalized sequency coefficients of $\bar{x}$. The ordering of the output coefficients for an eight point transform is shown in Figure 3-7. As discussed in section 3.8 the sequencies of the output coefficients are not in sequential order. The reason for this is that the DWT uses the functions wal and not sal or cal functions. Hence, the output coefficients are slightly shuffled in that even index coefficients correspond to cal coefficients and the odd index coefficients correspond to sal coefficients. Therefore, by computing an N point transform one only resolves the first $\frac{N}{2}$ sequency and coefficients, i.e., the

---

[39]Another interpretation is that the last $\frac{N}{2}$ term corresponds to the "negative" frequency terms of the Fourier transform. However, this interpretation is not rigorous and one may lose sight of the fact that the FFT computes discrete Fourier spectra.

[40]For a detailed discussion of how to interpret the DFT see Reference 16.

$$\begin{bmatrix} \text{wal}(0,\theta) \\ \text{wal}(1,\theta) \\ \text{wal}(2,\theta) \\ \text{wal}(3,\theta) \\ \text{wal}(4,\theta) \\ \text{wal}(5,\theta) \\ \text{wal}(6,\theta) \\ \text{wal}(7,\theta) \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \text{dc term} \\ \text{sal}(1,\theta) \\ \text{cal}(1,\theta) \\ \text{sal}(2,\theta) \\ \text{cal}(2,\theta) \\ \text{sal}(3,\theta) \\ \text{cal}(3,\theta) \\ \text{sal}(4,\theta) \end{bmatrix}$$

The ordering of the output data corresponds to the coefficients of these Walsh functions.

Figure 3-7 The Ordering of the Transform Vector $\bar{a}_W$ for an Eight Point Sequency Ordered Discrete Walsh Transform

dc coefficient and the first $\frac{N}{2}-1$ sequency coefficients. This result is similar to the FFT; however, there is a difference in that an "extra" coefficient is resolved using the FWT. For the eight point transform of Figure 3-7, the "extra" coefficient corresponds to the term sal$(4,\theta)$. For larger transforms, i.e. larger values of N, the extra coefficient is always a sal coefficient corresponding to sal $(\frac{N}{2},\theta)$. This extra coefficient presents no problem in interpreting the results of DWT. However, the user should be aware of this fact. Since the output coefficients are real numbers and only consist of the first $\frac{N}{2}$ sequencies (plus the extra term) the Nyquist folding phenomena is not observed in the output coefficients[41].

Despite the differences discussed above, the Fast Fourier Transform and Fast Walsh Transform subroutines are used in a similar manner. The user does not need to prepare the input data to be transformed in any special way and the calling procedures for the FWT subroutines are almost identical to that of the FFT subroutine. The FWT subroutines in general do not need any FORTRAN library marco programs such as the library subprograms SIN and COS needed for the FFT. Through very slight modifications to the FWT subroutines it is possible to perform all the transform calculations using integer arithmetic. This is in general not possible with the FFT. The advantages of using integer arithmetic is that it is faster and usually when computing the FWT on a digital

[41]The DWT spectrum does "fold" and exhibit periodicity properties similar to the DFT. The output of the FWT does not show this, however. See References 50 pp. 72-89, 2, 16, 26 and 52.

computer the input data to be transformed is in reality the output of an A/D converter. Most "fast" A/D converters operate using the integer mode such that the output of the converter is an integer word.

Finally, one should always remember when using either the FFT or the FWT that these subroutines compute discrete transforms that are analogous to it, but <u>not</u> identically equal to the Fourier series and Fourier Transforms discussed in Chapter 1.

3.7 Discrete Dyadic Convolution

Let x(i) and h(i), i = 0,1,2,......N-1, be sequences of N finite valued real numbers, the discrete arithmetic convolution of x(i) and h(i) is defined as:

$$y(k) = \sum_{i=0}^{N-1} x(i)h(k-i) \qquad (3\text{-}11)$$

as denoted as[42]

$$y(k) = x(k) * h(k) \qquad (3\text{-}12)$$

The following theorem is stated without proof.[43] Let y(k), x(k), h(k), k = 1,2,....N-1, be sequences of N finite valued real numbers and A_{F_h}, A_{F_x} and A_{F_y} denote the Discrete Fourier Transform of y, x, and h respectively. Then,

$$A_{F_y} = A_{F_x} A_{F_h} \qquad (3\text{-}13)$$

The results are similar to the standard integral convolution theorem.

[42]When denoting the discrete convolution product by the symbol * one should not confuse it with the standard integral convolution. For a detailed discussion of discrete convolution, see Reference 16, pp. 110-122.

[43]Ibid.

Similarly for the DWT, let x(i) and h(i), i = 0,1,2,....N-1, be sequences of N finite valued real numbers, the <u>discrete dyadic convolution</u> of x(i) and h(i) is defined as:

$$y(k) = \sum_{i=0}^{N-1} x(i)h(k \oplus i) \qquad (3\text{-}14)$$

and is denoted as

$$y(k) = x(k) \circledast h(k) \qquad (3\text{-}15)$$

The similarity to equation 1-21 is obvious.

The following discrete dyadic convolution theorem is stated without proof.[44] Let y(k), x(k), and h(k), k = 1,2,....N-1, be sequences of N finite valued real numbers. Let $y(k) = x(k) \circledast h(k)$ and A_{W_y}, A_{W_x} and A_{F_h} denote the Discrete Walsh Transform of y, x, and h respectively. Then,

$$A_{W_y} = A_{W_x} A_{W_h} \qquad (3\text{-}16)$$

The results are similar to the integral dyadic convolution theorem of equation 1-22.

The DWT therefore possesses a convolution property that is different than standard convolution. The discrete dyadic convolution of equation 3-14 does not lend itself to physical interpretations similar to the arithmetic convolution, i.e., the "folding" of one function and "sliding" of the folded function with respect to the other function. The importance of the discrete dyadic convolution will be discussed in Chapter 4 relative to the definition of dyadic invariant linear systems.

[44]For a proof of this theorem and a discussion of discrete dyadic convolution, see Reference 52.

A relatively simple example of discrete dyadic convolution is shown in Figures 3-8 through 3-10. The rectangular pulse shown in Figure 3-8[45] is convolved with itself using the FFT and the discrete arithmetic convolution theorem of equation 3-13, the results are shown in Figure 3-9. The discrete dyadic convolution of the rectangular pulse was computed using the FWT and the convolution theorem of equation 3-16. The results are shown in Figure 3-10. It is obvious that the two convolution products yield totally different results leading to different interpretations.

[45]Figures 3-8 through 3-10 are presented as bar graphs to emphasize that they are discrete time functions.

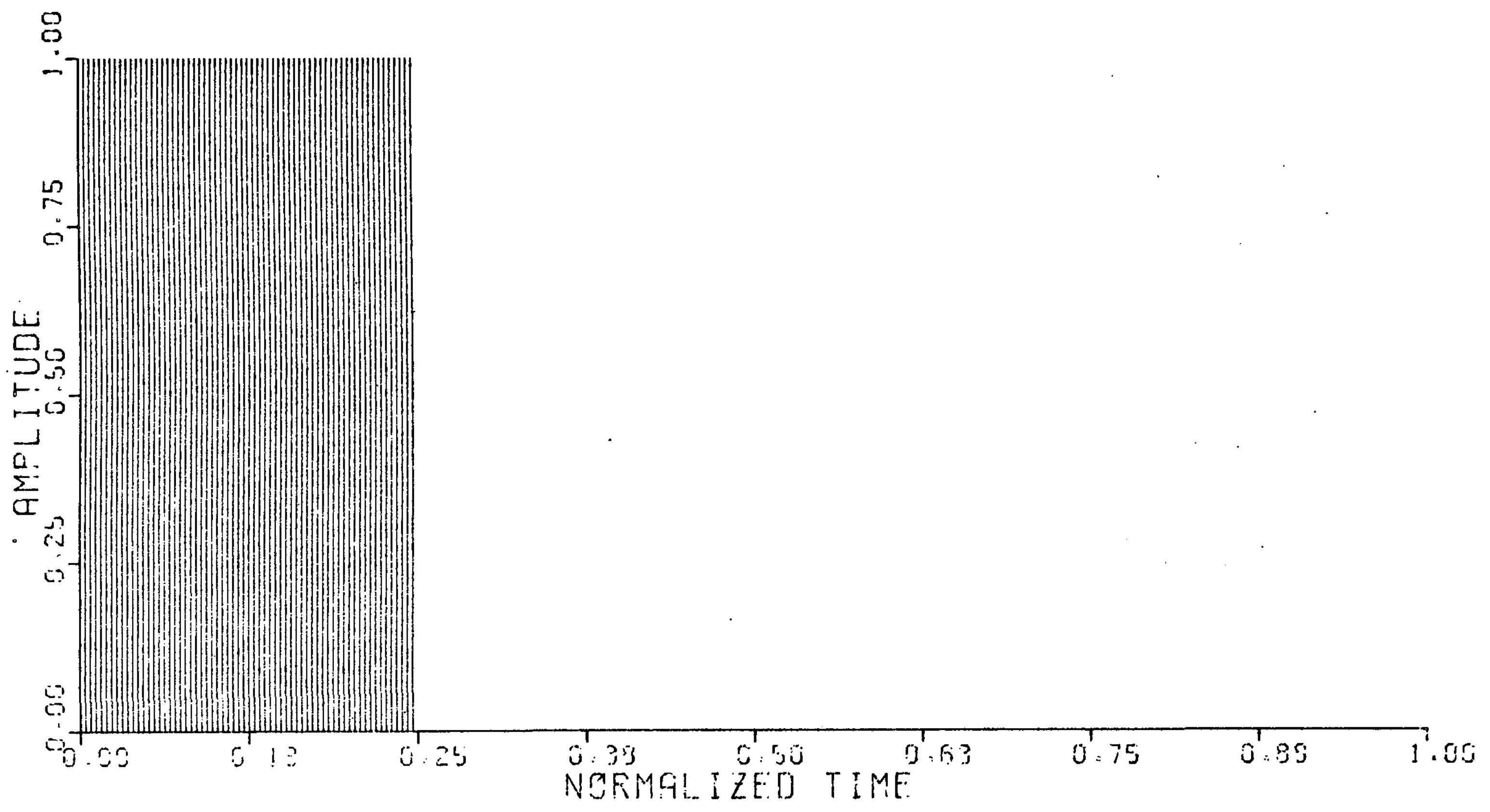


Figure 3-8 Discrete Rectangular Pulse

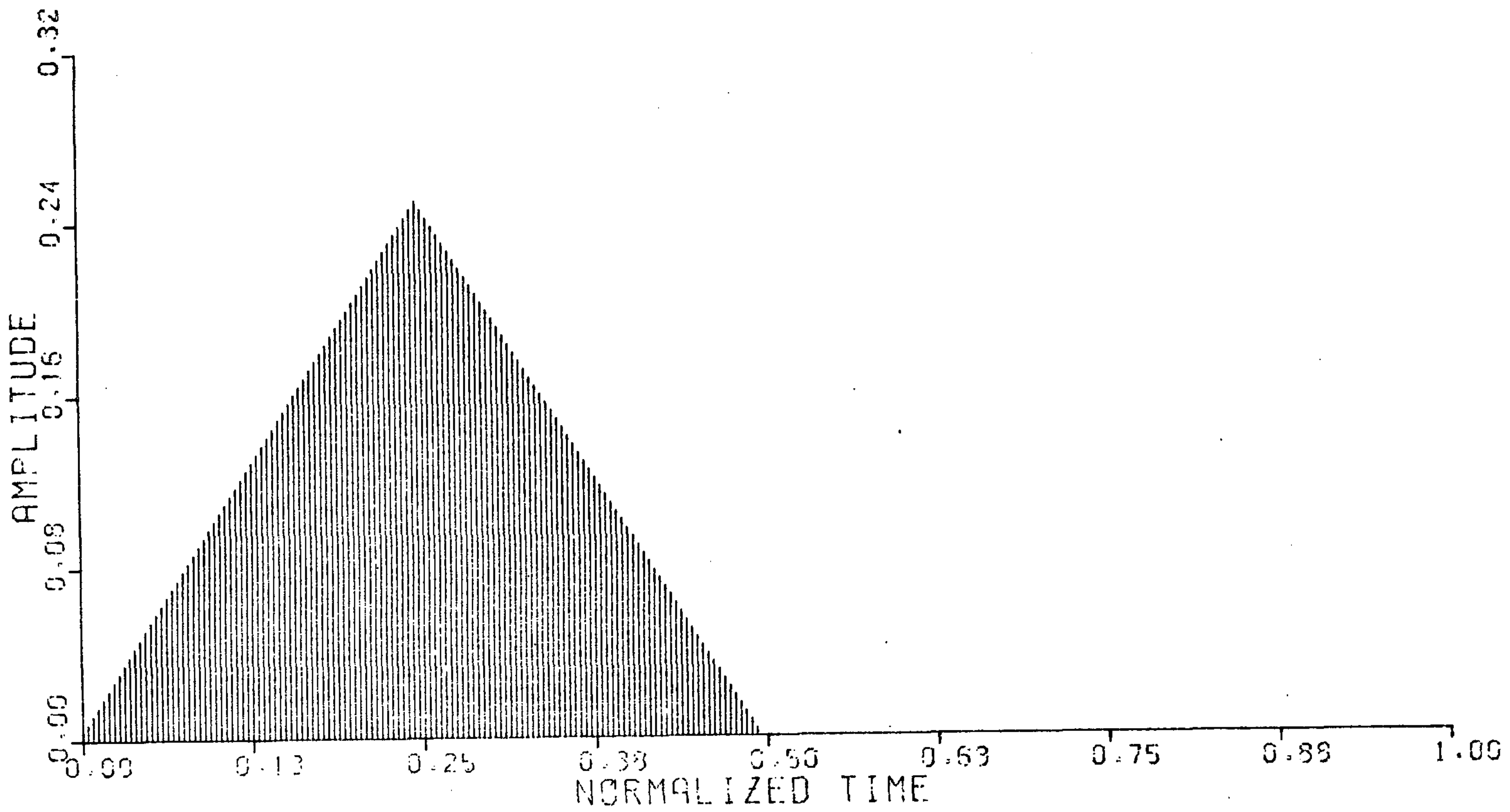


Figure 3-9 Result of Discrete Arithmetic Convolution of Two Rectangular Pulses Shown in Figure 3-8

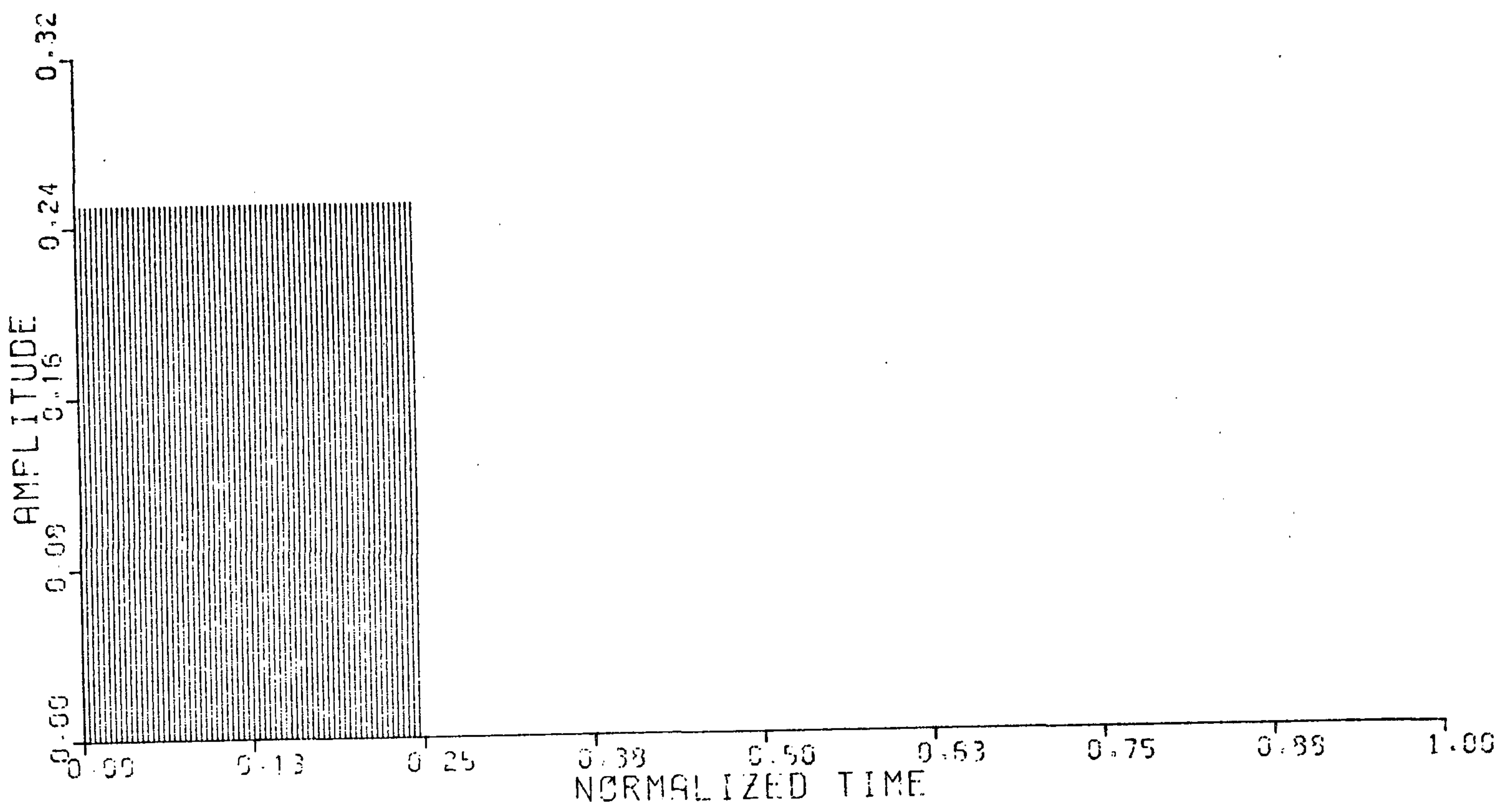


Figure 3-10 Result of Discrete Dyadic Convolution of Two Rectangular Pulses Shown in Figure 3-8

Chapter 4

EXPERIMENTAL WORK

4.0 Introduction

The purpose of this chapter is to demonstrate a technique for waveform synthesis using the Walsh function generator discussed in Chapter 2. The input-output relationships of a relatively simple sequency filter will also be analyzed using the Discrete Walsh Transform. A brief introduction will be presented to discrete dyadic-invariant linear systems.

4.1 Waveform Synthesis Using Walsh Functions

The Walsh function generator discussed in Chapter 2 was used to construct some relatively simple waveforms. This was accomplished by connecting the 15 outputs of the generator to the AD-4 analog computer via the analog trunk lines of the University Hybrid Computer System. This then allowed access to the outputs of the generator directly on the AD-4 analog patchboard[1]. The first 15 (non-dc) Walsh coefficients were calculated using the subroutine FWT4 (see Chapter 3) for various waveforms. The procedure used was one of writing a FORTRAN mainline routine that evaluated the desired time function; then a 1024 point Fast Walsh Transform was obtained using FWT4 resulting in a sequency resolution of the first 512 sequencies. The time function was assumed to have a period of one (normalized time was used)[2].

[1]The TTL outputs of generator were treated as true analog signals.

[2]The Discrete Fourier Transform and Discrete Walsh Transform assume the function to be transformed is periodic, with the period being the unit interval. See Chapter 3 and References 16, 52 and 83.

A valid question at this point is, why use so many points if only the 15 coefficients are desired? Why not use a 16 point transform? The answer is simply more accuracy was obtained by taking a larger transform. It was desired to resolve the first 15 coefficients as accurately as possible; therefore a larger transform was taken to reduce truncation errors (see Chapter 5)[3]. A 1024 point transform was chosen for its relatively fast computing time (14 seconds) on the IBM system. Even with such a large transform some errors were noted in the coefficients due to the larger truncation errors for the FWT as compared to the FFT (see Chapter 5).

Once the first 15 coefficients were obtained, the nonzero coefficients were used to set coefficient potentiometers (pots) connected to the respective Walsh function generator outputs (see Figure 4-1). The outputs of the pots were connected to a unity gain summing amplifier to sum the weighted Walsh functions. The output of the amplifier was displayed on the Brush-8 strip chart recorder. In other words, a particular waveform was constructed by weighting the outputs of the Walsh function generator with its respective coefficient and then the weighted Walsh functions were added together. Since the original time function and the Walsh function generator outputs are periodic one can think of this as a truncated "inverse" Walsh-Fourier series.

The above method was used to synthesize five waveforms; the results are shown in Figure 4-2. Tables 4-1 through 4-5 show the first 15 Walsh coefficients (and the dc terms) used in the synthesis procedure as

[3]For a discussion of truncation errors, see References 13 and 16.

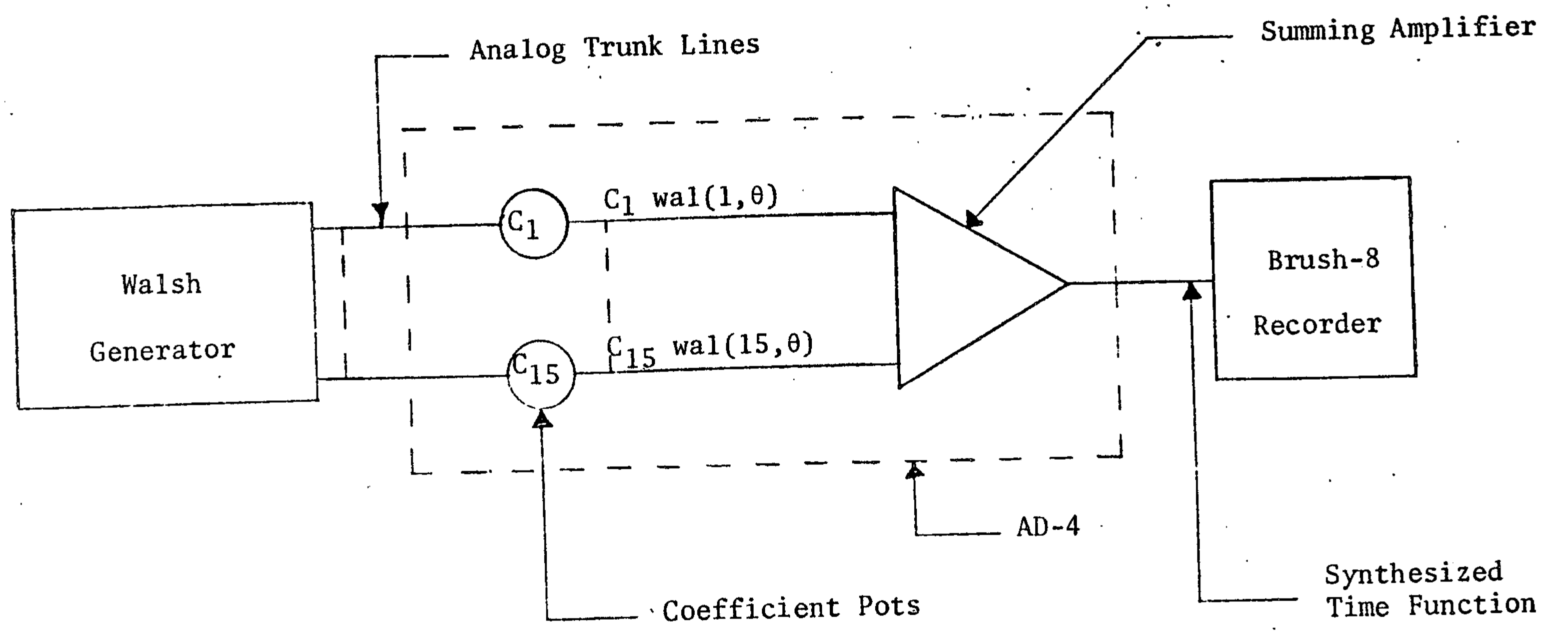
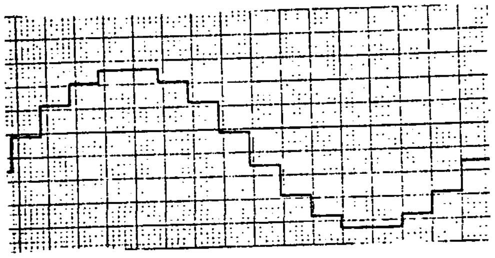
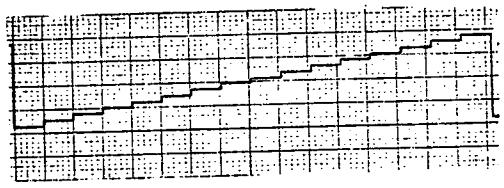


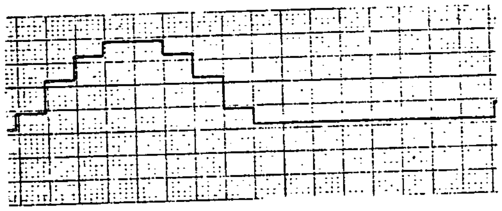
Figure 4-1 Block Diagram of Walsh Generator/AD-4 Setup



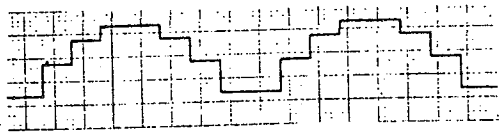
Sine Wave



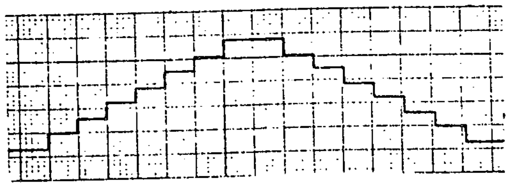
Ramp Function



Half-Wave Recified Sine Wave



Full-Wave Rectified Sine Wave



Triangular Function

Figure 4-2 Synthesized Functions

Table 4-1

First 16 Walsh Coefficients for Sine Wave of Figure 4-3

Walsh Function	Coefficient Value
wal(0,θ) [cal(0,θ)]	0.00000
wal(1,θ) [sal(1,θ)]	0.63661
wal(2,θ) [cal(1,θ)]	-0.00195*
wal(3,θ) [sal(2,θ)]	0.00000
wal(4,θ) [cal(2,θ)]	0.00000
wal(5,θ) [sal(3,θ)]	-0.26369
wal(6,θ) [cal(3,θ)]	-0.00080*
wal(7,θ) [sal(4,θ)]	0.00000
wal(8,θ) [cal(4,θ)]	0.00000
wal(9,θ) [sal(5,θ)]	-0.05245
wal(10,θ) [cal(5,θ)]	0.00016*
wal(11,θ) [sal(6,θ)]	0.00000
wal(12,θ) [cal(6,θ)]	0.00000
wal(13,θ) [sal(7,θ)]	-0.12663
wal(14,θ) [cal(7,θ)]	-0.00038*
wal(15,θ) [sal(8,θ)]	0.00000

*Assumed to be zero.

Table 4-2

First 16 Walsh Coefficients for Ramp Function of Figure 4-5

Walsh Function	Coefficient Value
wal(0,θ) [cal(0,θ)]	0.49951
wal(1,θ) [sal(1,θ)]	-0.25000
wal(2,θ) [cal(1,θ)]	0.00000
wal(3,θ) [sal(2,θ)]	-0.12500
wal(4,θ) [cal(2,θ)]	0.00000
wal(5,θ) [sal(3,θ)]	0.00000
wal(6,θ) [cal(3,θ)]	0.00000
wal(7,θ) [sal(4,θ)]	-0.06250
wal(8,θ) [cal(4,θ)]	0.00000
wal(9,θ) [sal(5,θ)]	0.00000
wal(10,θ)[cal(5,θ)]	0.00000
wal(11,θ)[sal(6,θ)]	0.00000
wal(12,θ)[cal(6,θ)]	0.00000
wal(13,θ)[sal(7,θ)]	0.00000
wal(14,θ)[cal(7,θ)]	0.00000
wal(15,θ)[sal(8,θ)]	-0.03125

Table 4-3

First 16 Walsh Coefficients for Half-Wave Rectified Sine-Wave Shown in Figure 4-7

Walsh Function	Coefficient Value
wal(0,θ) [cal(0,θ)]	0.31830
wal(1,θ) [sal(1,θ)]	0.31830
wal(2,θ) [cal(1,θ)]	-0.00097*
wal(3,θ) [sal(2,θ)]	-0.00097*
wal(4,θ) [cal(2,θ)]	-0.13184
wal(5,θ) [sal(3,θ)]	-0.13184
wal(6,θ) [cal(3,θ)]	-0.00040*
wal(7,θ) [sal(4,θ)]	-0.00040*
wal(8,θ) [cal(4,θ)]	-0.02622
wal(9,θ) [sal(5,θ)]	-0.02622
wal(10,θ)[cal(5,θ)]	0.00008*
wal(11,θ)[sal(6,θ)]	0.00008*
wal(12,θ)[cal(6,θ)]	-0.06331
wal(13,θ)[sal(7,θ)]	-0.06331
wal(14,θ)[cal(7,θ)]	-0.00019*
wal(15,θ)[sal(8,θ)]	-0.00019

*Assumed to be zero.

Table 4-4

First 16 Walsh Coefficients for Full-Wave Rectified Sine-Wave Shown in Figure 4-10

Walsh Function	Coefficient Value
wal(0,θ) [cal(0,θ)]	0.63661
wal(1,θ) [sal(1,θ)]	0.00000
wal(2,θ) [cal(1,θ)]	0.00000
wal(3,θ) [sal(2,θ)]	-0.00195*
wal(4,θ) [cal(2,θ)]	-0.26369
wal(5,θ) [sal(3,θ)]	0.00000
wal(6,θ) [cal(3,θ)]	0.00000
wal(7,θ) [sal(4,θ)]	-0.00080*
wal(8,θ) [cal(4,θ)]	-0.05245
wal(9,θ) [sal(5,θ)]	0.00000
wal(10,θ) [cal(5,θ)]	0.00000
wal(11,θ) [sal(6,θ)]	0.00016*
wal(12,θ) [cal(6,θ)]	-0.12663
wal(13,θ) [sal(7,θ)]	0.00000
wal(14,θ) [cal(7,θ)]	0.00000
wal(15,θ) [sal(8,θ)]	-0.00038*

*Assumed to be zero.

Table 4-5

First 16 Walsh Coefficients for Triangular Function Shown in Figure 4-12

Walsh Function	Coefficient Value
wal(0,θ) [cal(0,θ)]	0.50000
wal(1,θ) [sal(1,θ)]	-0.00097*
wal(2,θ) [cal(1,θ)]	-0.25000
wal(3,θ) [sal(2,θ)]	0.00000
wal(4,θ) [cal(2,θ)]	0.00000
Wal(5,θ) [sal(3,θ)]	0.00000
wal(6,θ) [cal(3,θ)]	-0.12500
wal(7,θ) [sal(4,θ)]	0.00000
wal(8,θ) [cal(4,θ)]	0.00000
wal(9,θ) [sal(5,θ)]	0.00000
wal(10,θ) [cal(5,θ)]	0.00000
wal(11,θ) [sal(6,θ)]	0.00000
wal(12,θ) [cal(6,θ)]	0.00000
wal(13,θ) [sal(7,θ)]	0.00000
wal(14,θ) [cal(7,θ)]	-0.06250
wal(15,θ) [sal(8,θ)]	0.00000

*Assumed to be zero.

calculated using FWT4. Figures 4-3 through 4-13 show plots of the original time functions used to calculate the Walsh coefficients and relevant nonzero sal and/or cal spectrum plots. The dc or wal(0,θ) coefficient was ignored since only waveshapes and not absolute signal levels were desired.

All of the waveforms synthesized possessed some sort of symmetric properties such as evenness or oddness or quarter wave symmetry, etc. The symmetry rules for the Walsh-Fourier series are identical with the Sine-Cosine Fourier series; i.e. an odd function has only odd Walsh function coefficients, etc. In this context, one notes from Table 4-1, that a sine wave, which is an odd function, appears to have a nonzero coefficient for wal(2,θ), or cal(1,θ), which is an even function. This is due to truncation errors mentioned above since a finite number of points (1024) were used for computing that coefficient. This is a problem that has been discussed in the literature and is due to the nature of the discrete transform[4]. In any event these types of coefficients were assumed to be zero for the waveform synthesis experiment and are so labeled in Tables 4-1 through 4-5.

The synthesized waveforms of Figure 4-2 have a staircase-like appearance similar to the output of a hold device used in sampled-data systems. These waveforms are sequency-limited functions, i.e., their Walsh spectra is limited to the first 15 Walsh coefficients. If the generator was capable of generating a large set of Walsh functions the "jumps" or steps would be less pronounced and the waveform would approach (in the limit) a smooth function.

[4]Ibid.

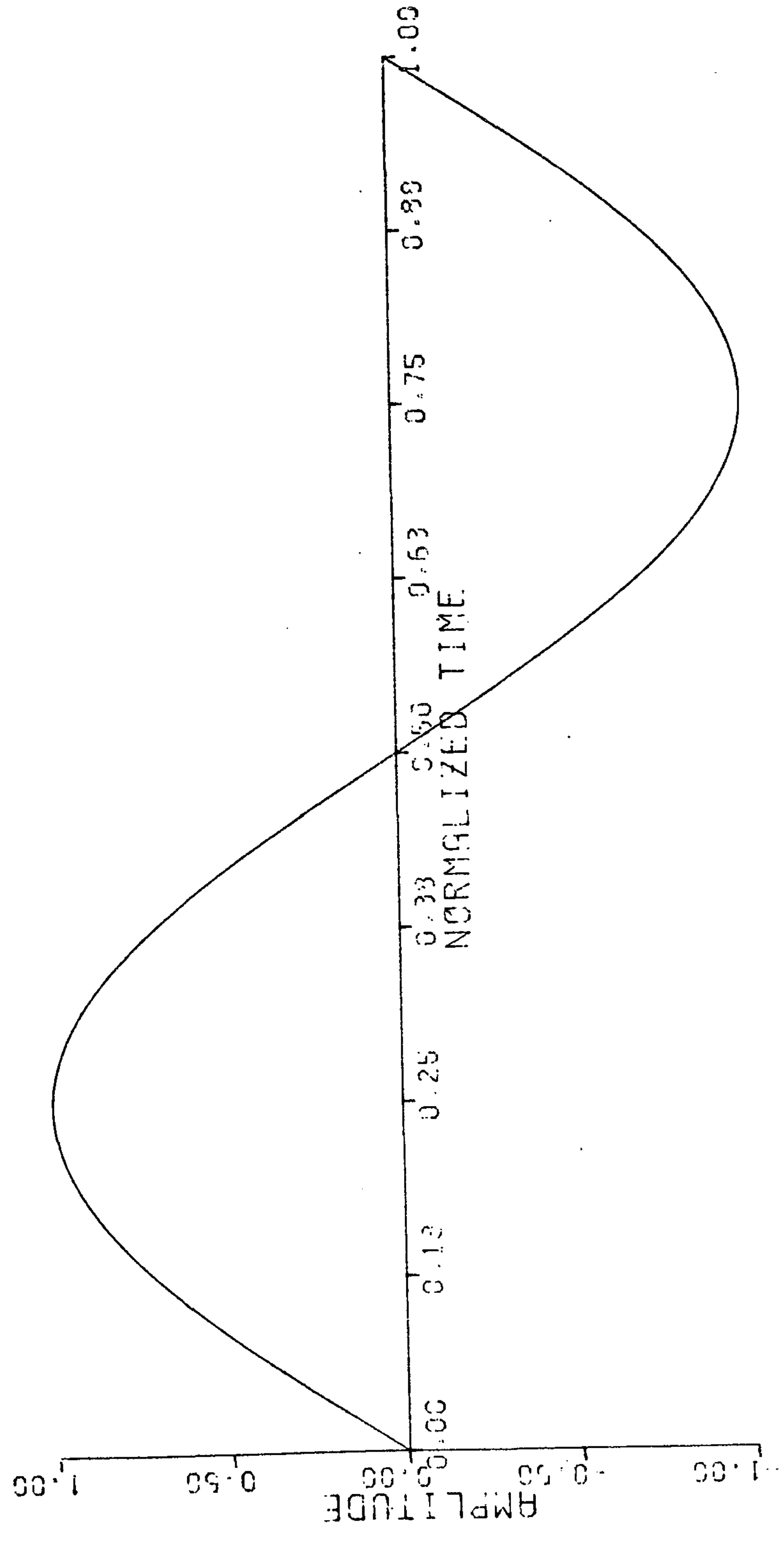


Figure 4-3 Sine Wave

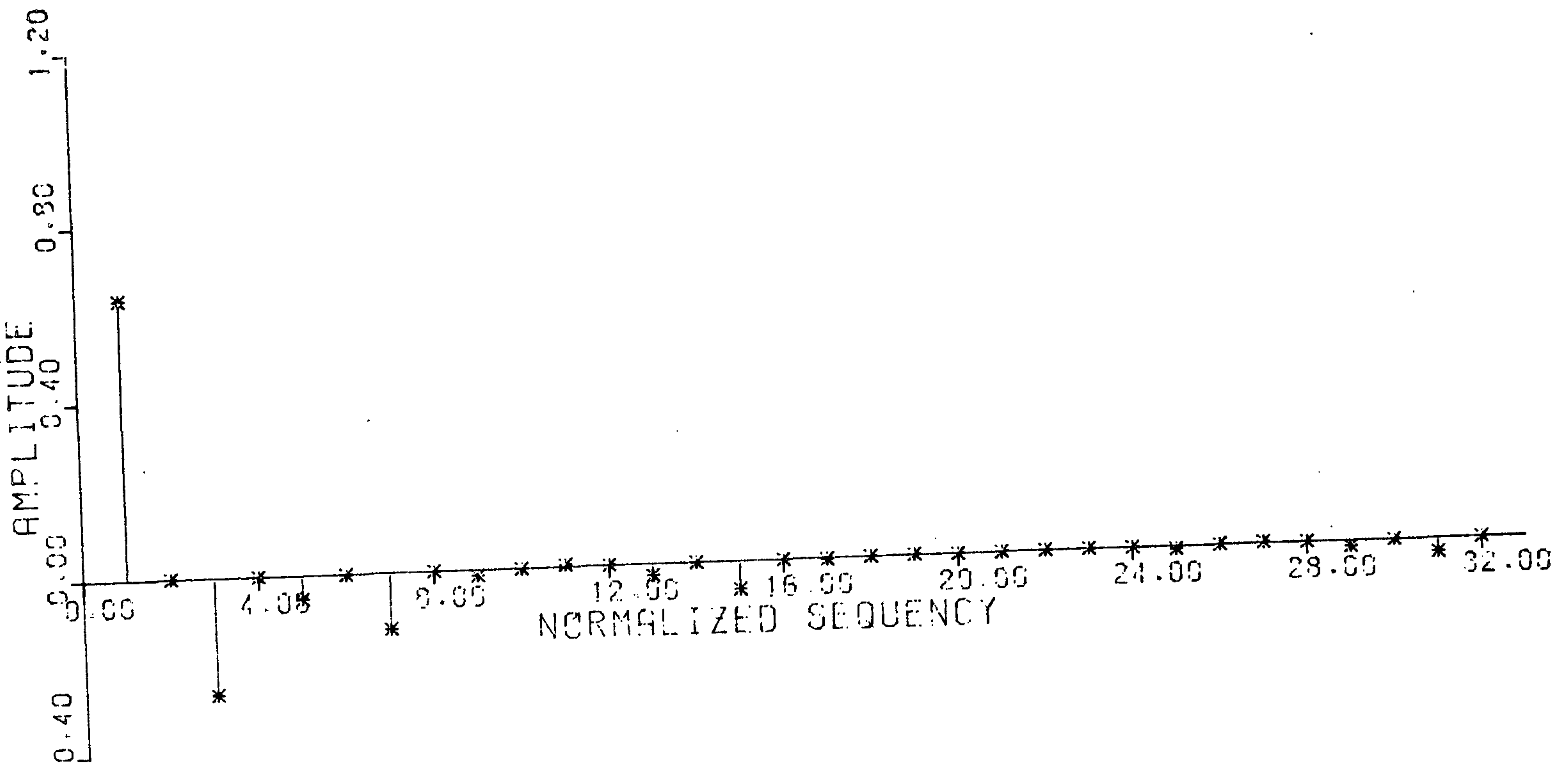


Figure 4-4 Sal Domain Description of Sine Wave

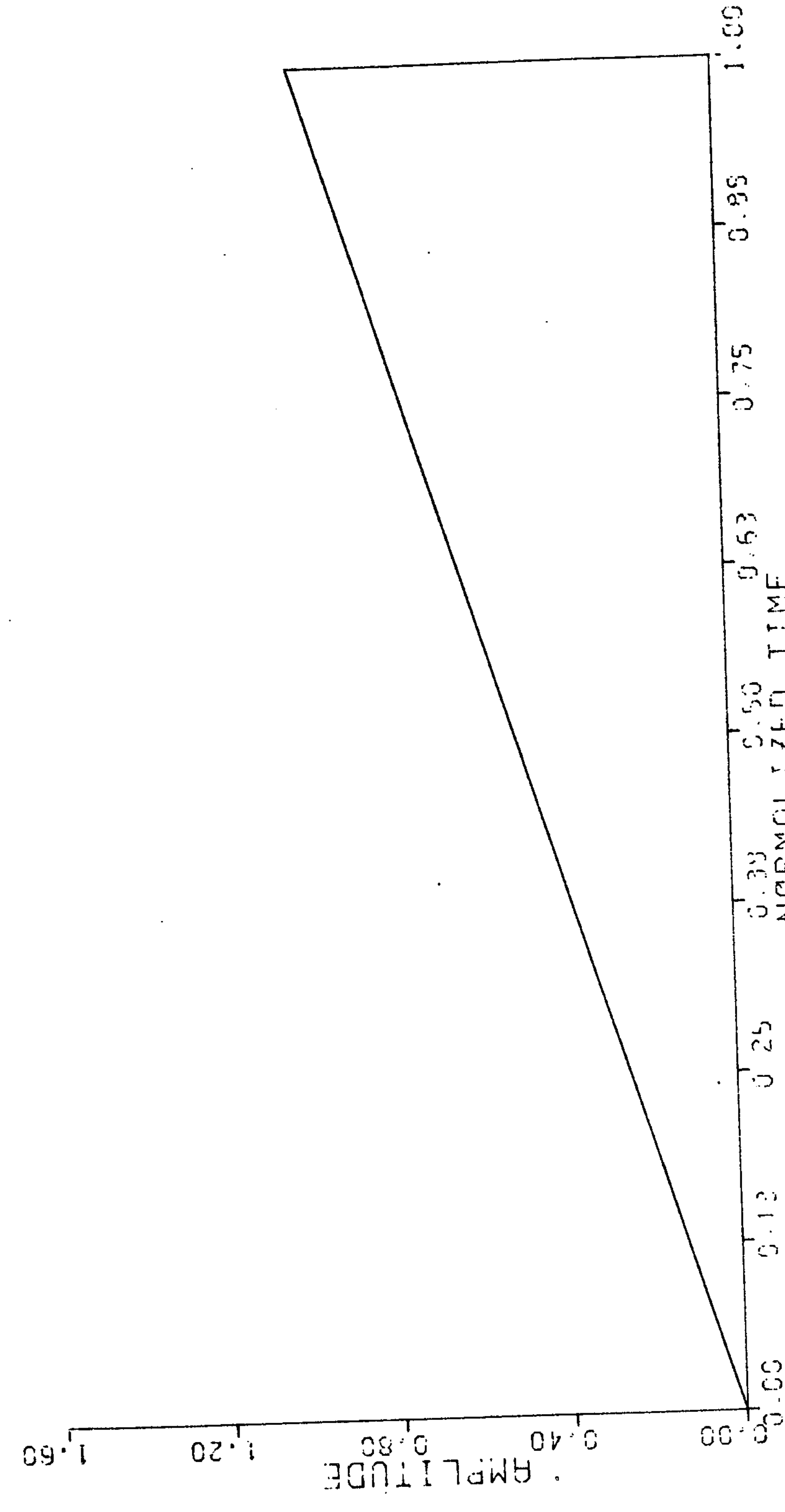


Figure 4-5 Ramp Function

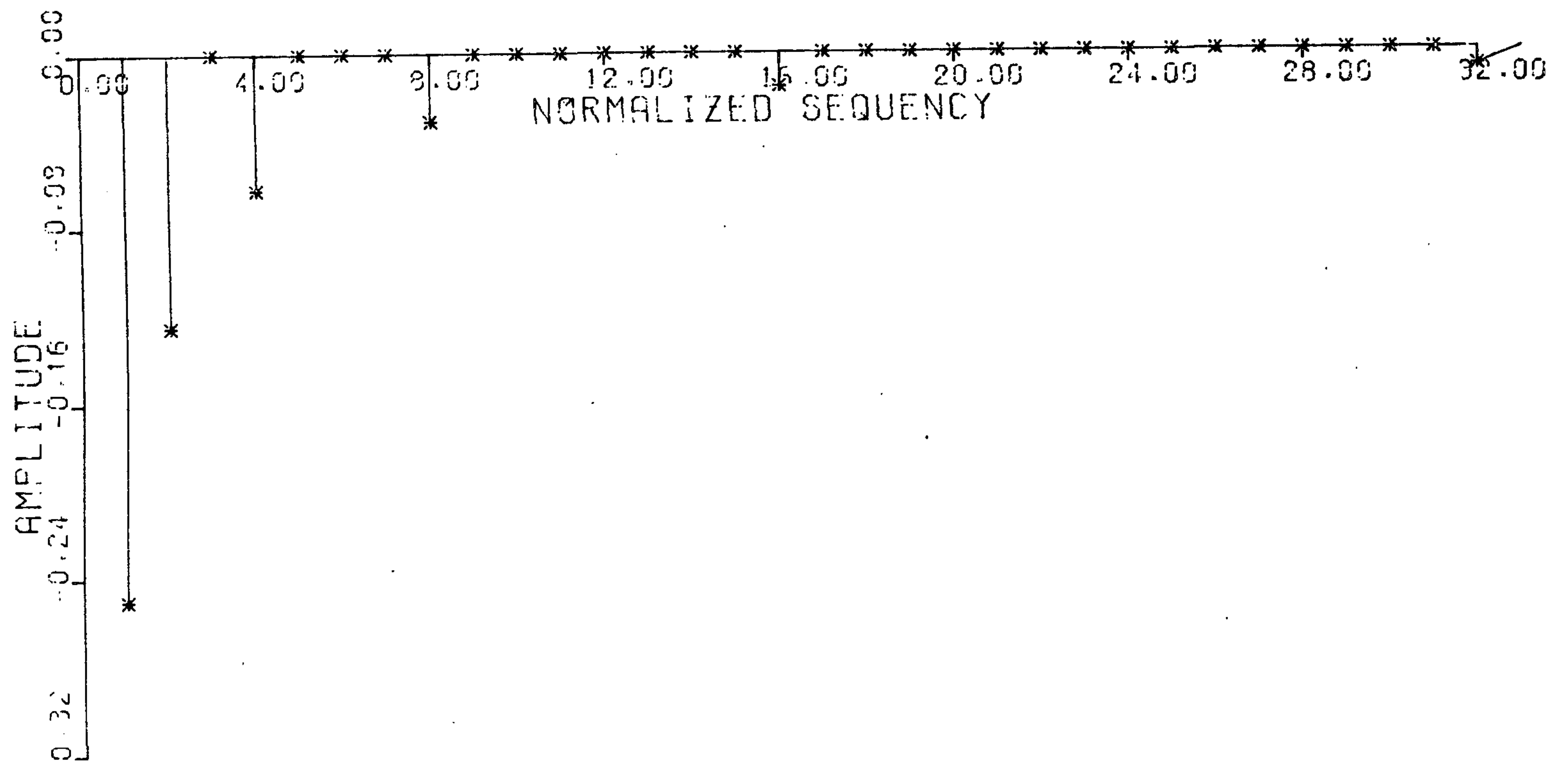


Figure 4-6 Sal Domain Description of Ramp Function

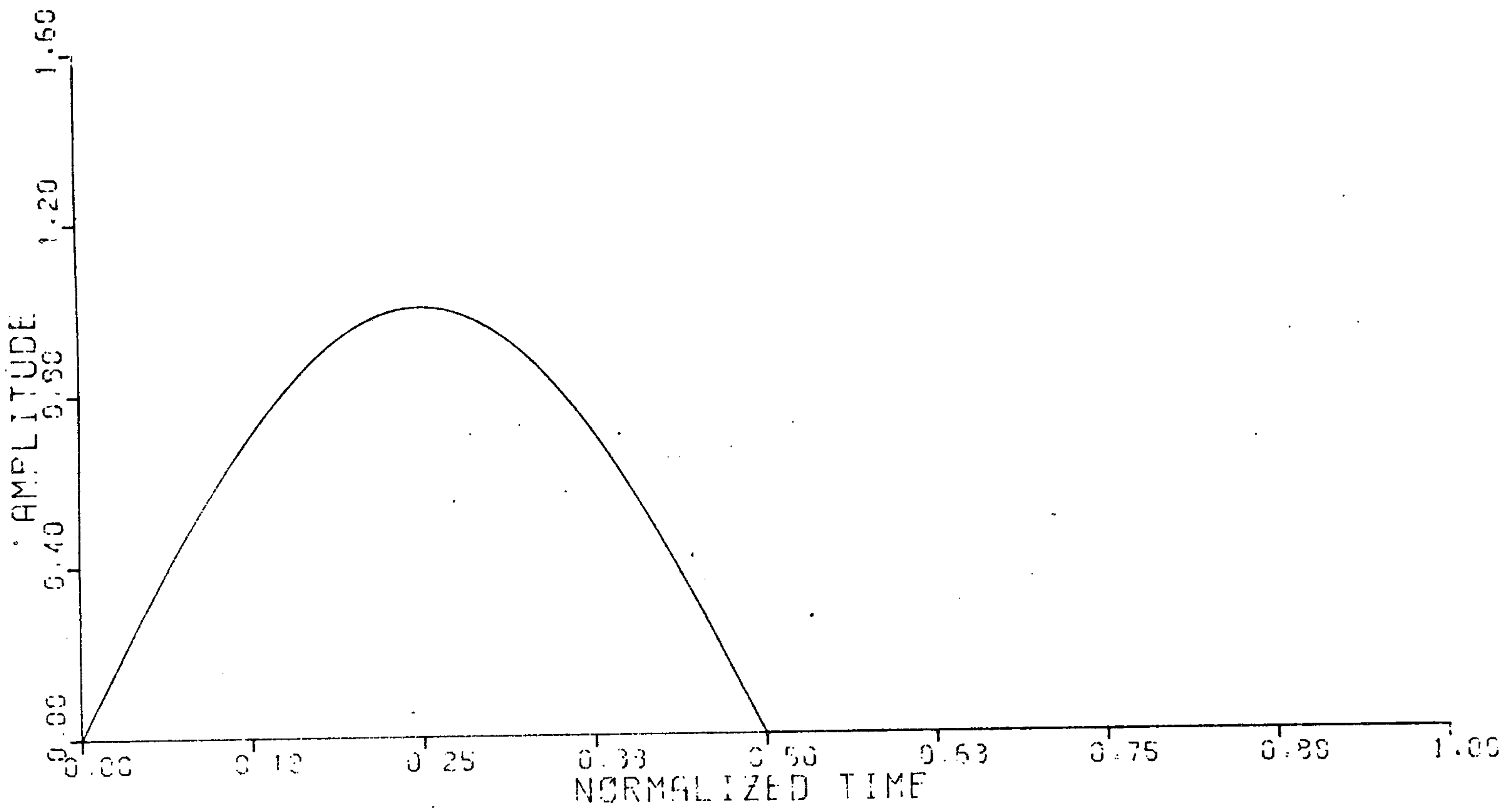


Figure 4-7 Half-Wave Rectified Sine-Wave

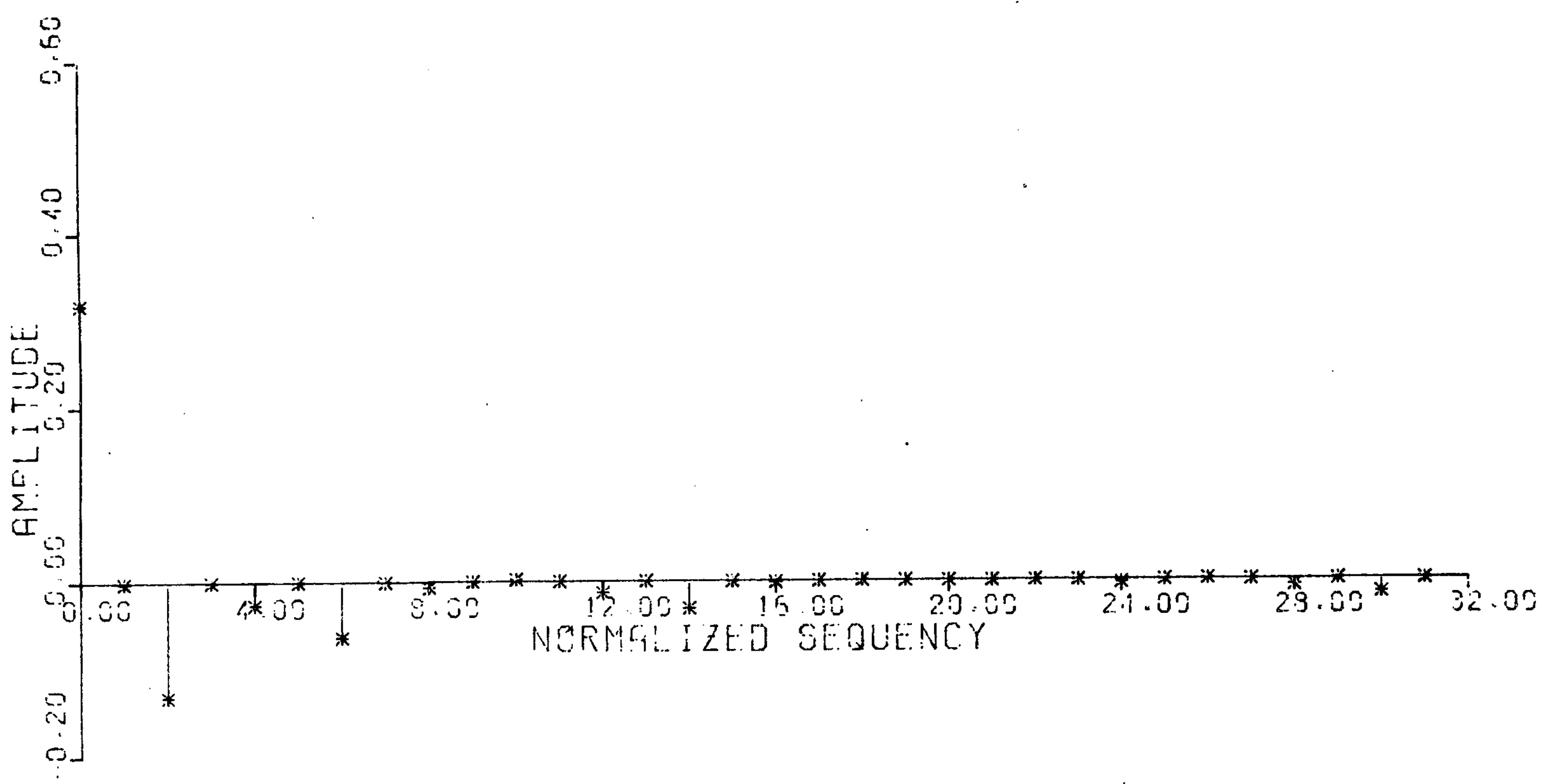


Figure 4-8 Cal Domain Description of Half-Wave Rectified Sine-Wave

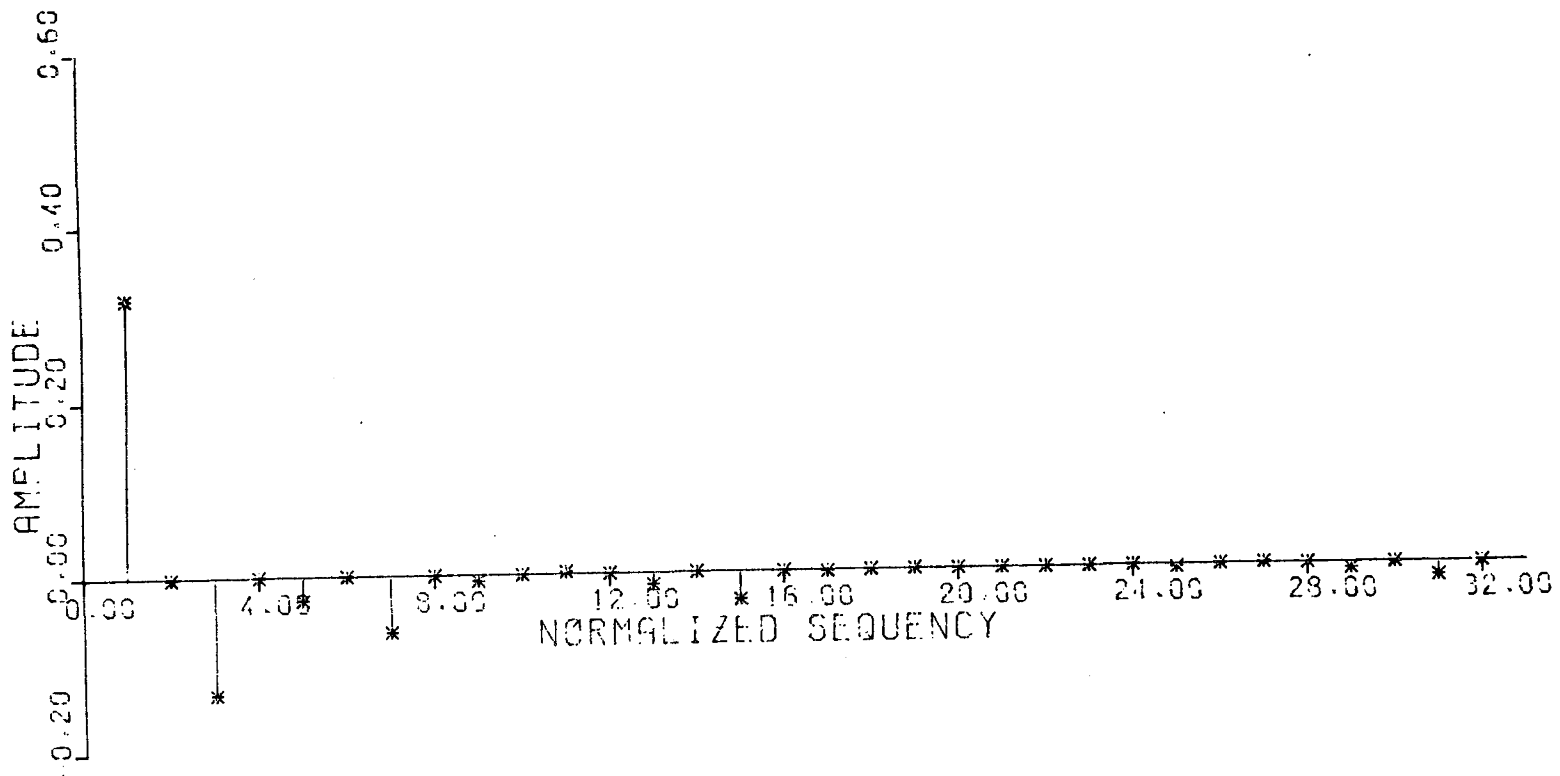


Figure 4-9 Sal Domain Description of Half-Wave Rectified Sine-Wave

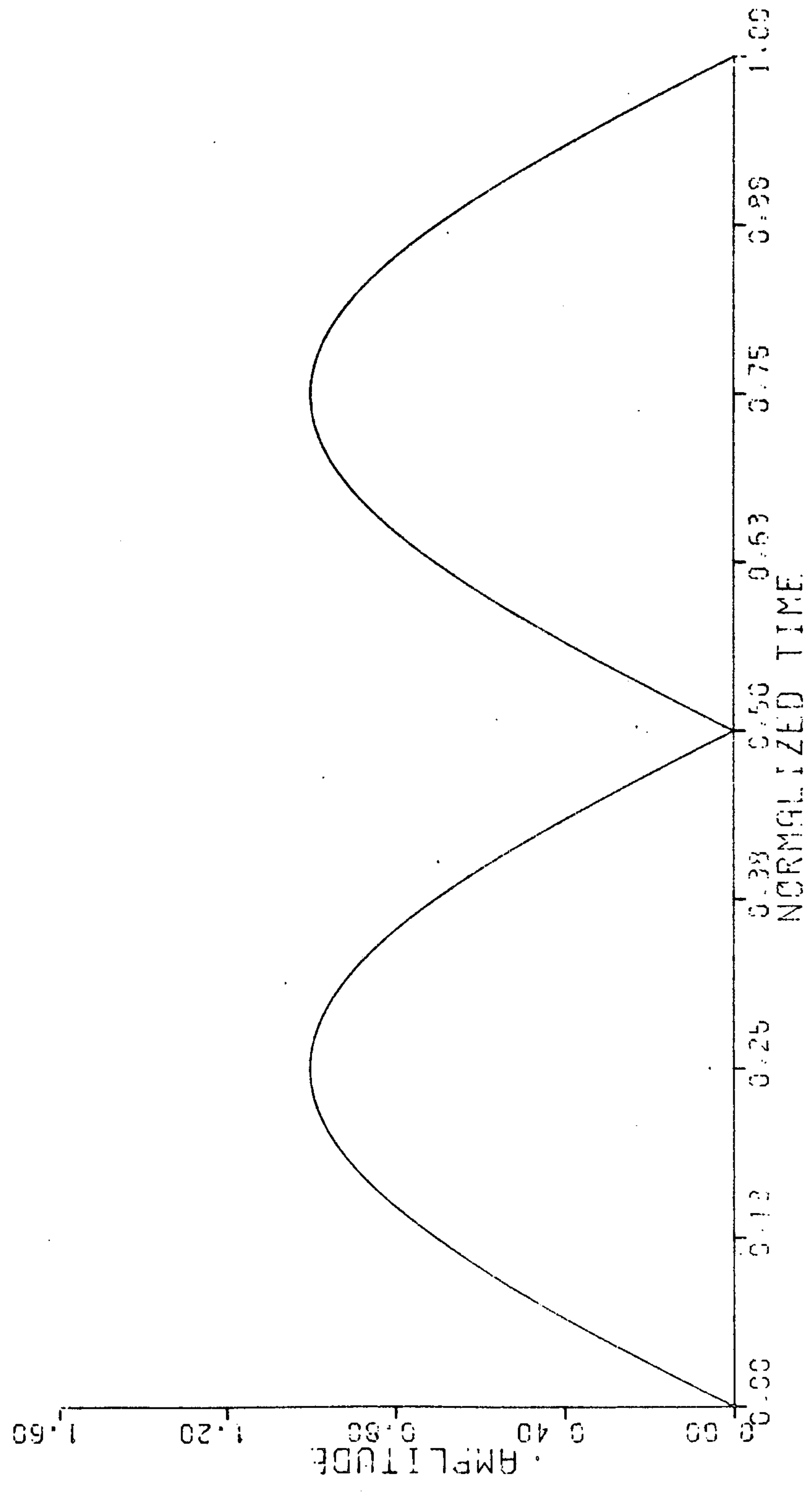


Figure 4-10 Full-Wave Rectified Sine-Wave

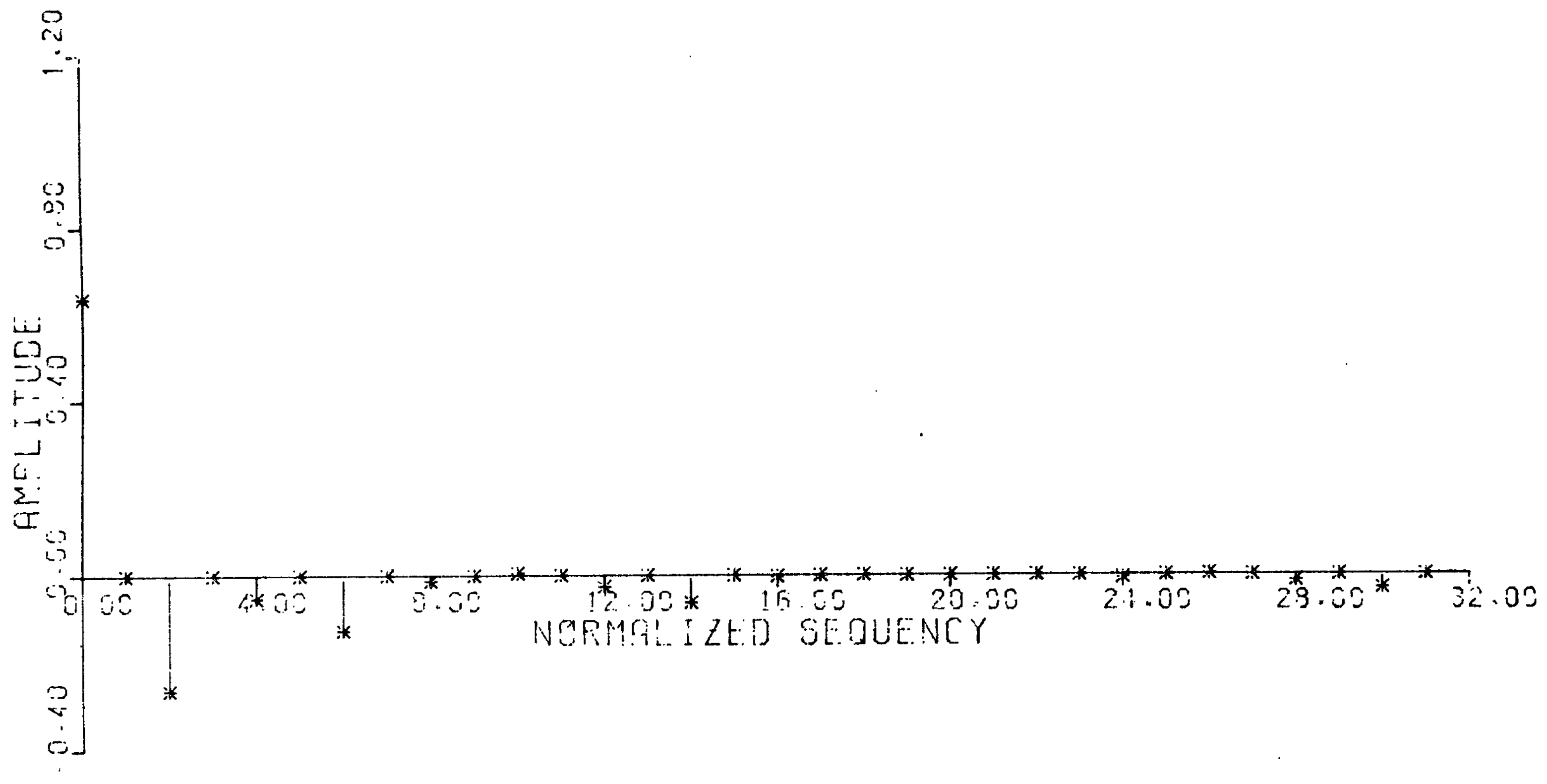


Figure 4-11 Cal Domain Description of Full-Wave Rectified Sine-Wave

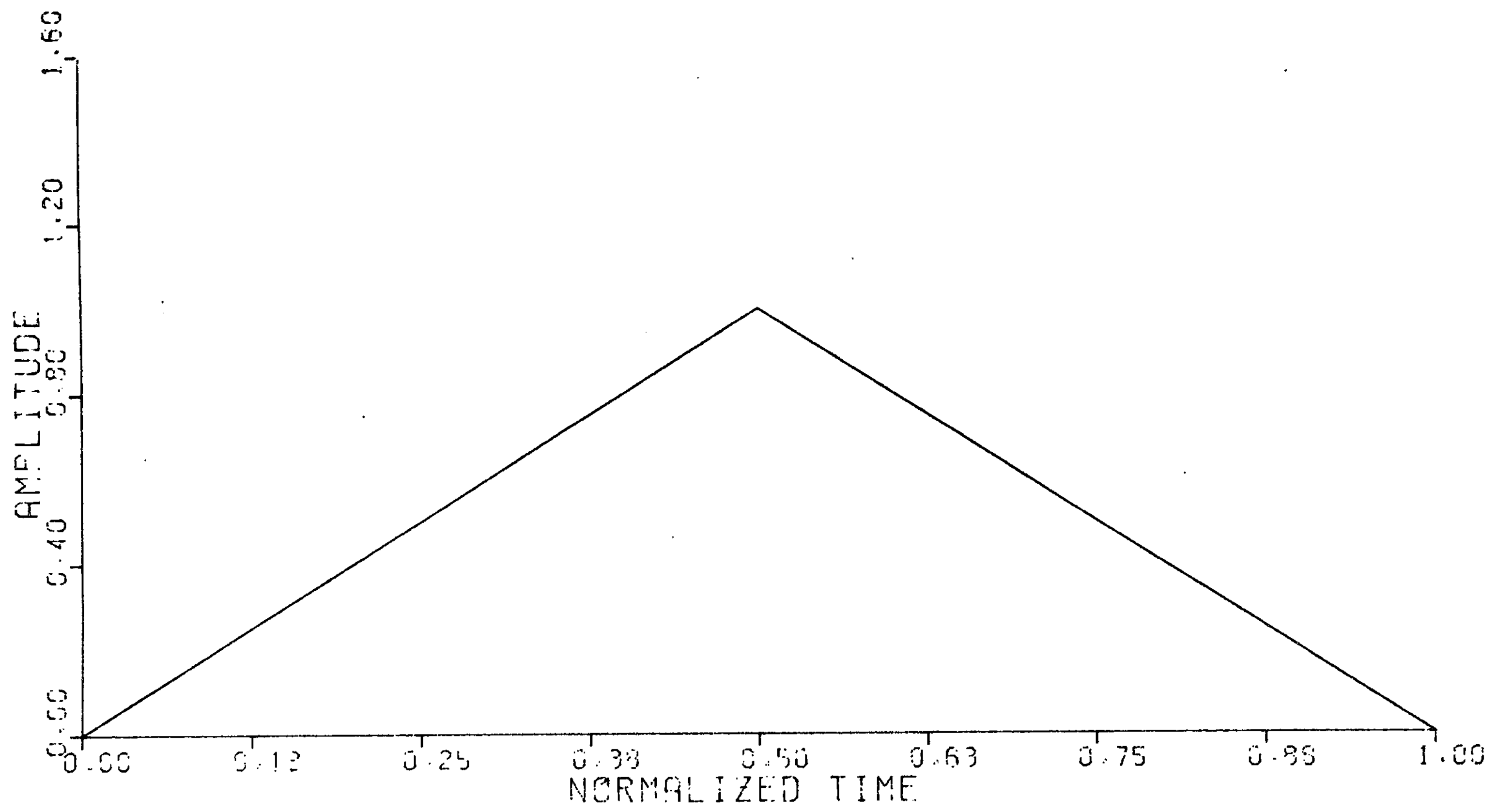


Figure 4-12 Triangular Function

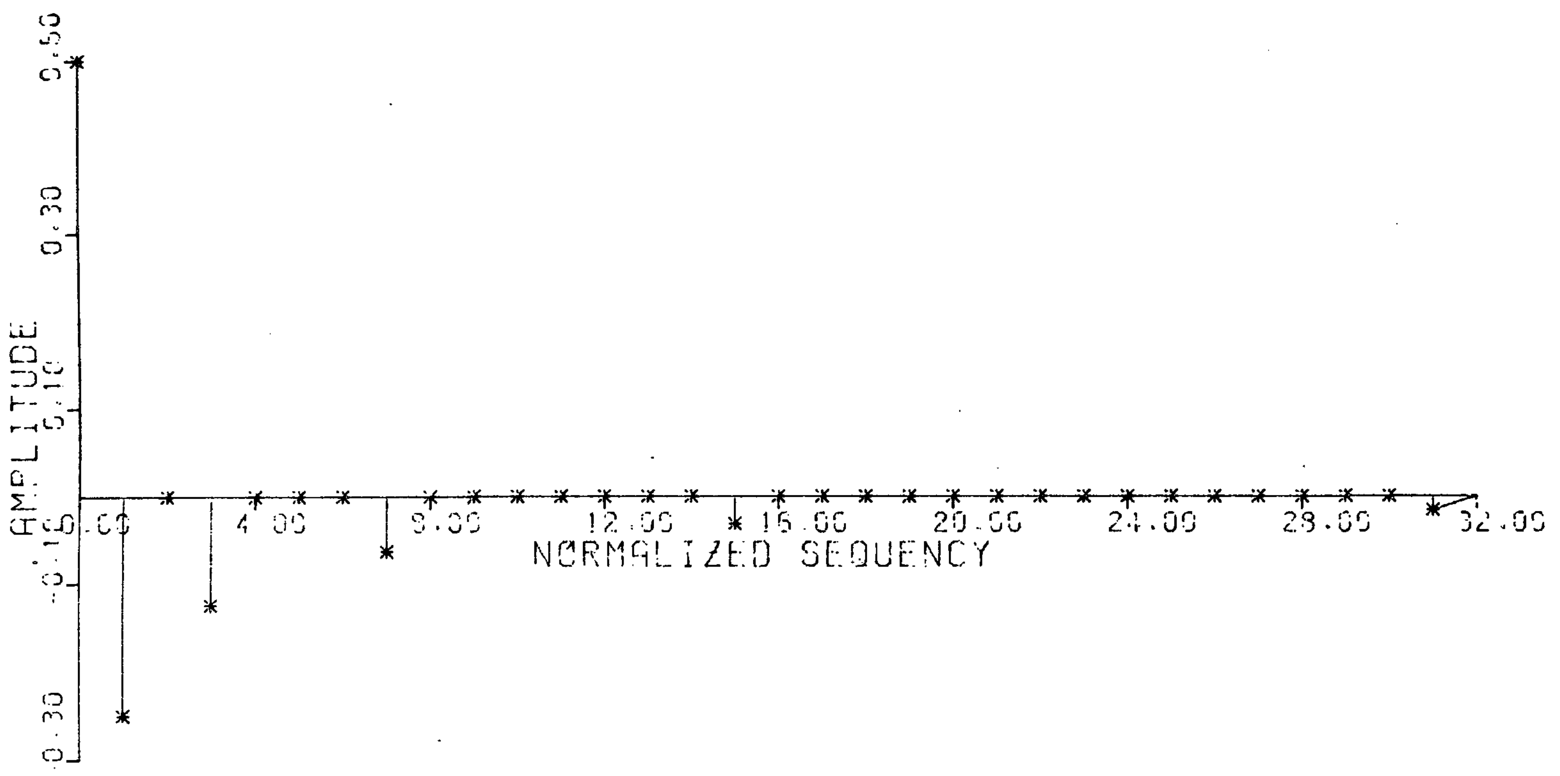


Figure 4-13 Cal Domain Description of Triangular Funcion

By using the above procedures a method has been demonstrated for synthesizing waveforms that is relatively simple. Hutchins and Insam[5] have employed these techniques for realizing an electronic music synthesizer device and have suggested an electronic organ based on this method. This synthesis method was possible because a generator was available that produced a set of synchronized Walsh functions simultaneously.

4.2 Dyadic Invariant Linear Systems

A linear system is said to be time invariant when a time translation of the input function results in the identical time translation of the output function (see Figure 4-14). A linear system is said to be dyadic invariant when a dyadic time translation of the input function results in the identical dyadic time translation of output function (see Figure 4-15).

Johnson and Pichler have shown that the input-output relationship for dyadic invariant linear systems is described by the dyadic convolution integral discussed in Chapter 1[6]. Therefore, if h(t) is the impulse response of such a system, the input-output relationship is given by:

$$y(t) = \int_{-\infty}^{\infty} x(t)\ h(t \oplus \tau) d\tau \qquad (4\text{-}1)$$

where x(t) is the input to the system and y(t) is the output. The response of a dyadic invariant system would be of limited interest if the dyadic convolution had to be performed for every input function. This is similar to classical linear system theory where the convolution

[5]See References 48 and 49.

[6]See References 20, 21, 23, 24, 50, 69, 70 and 71.

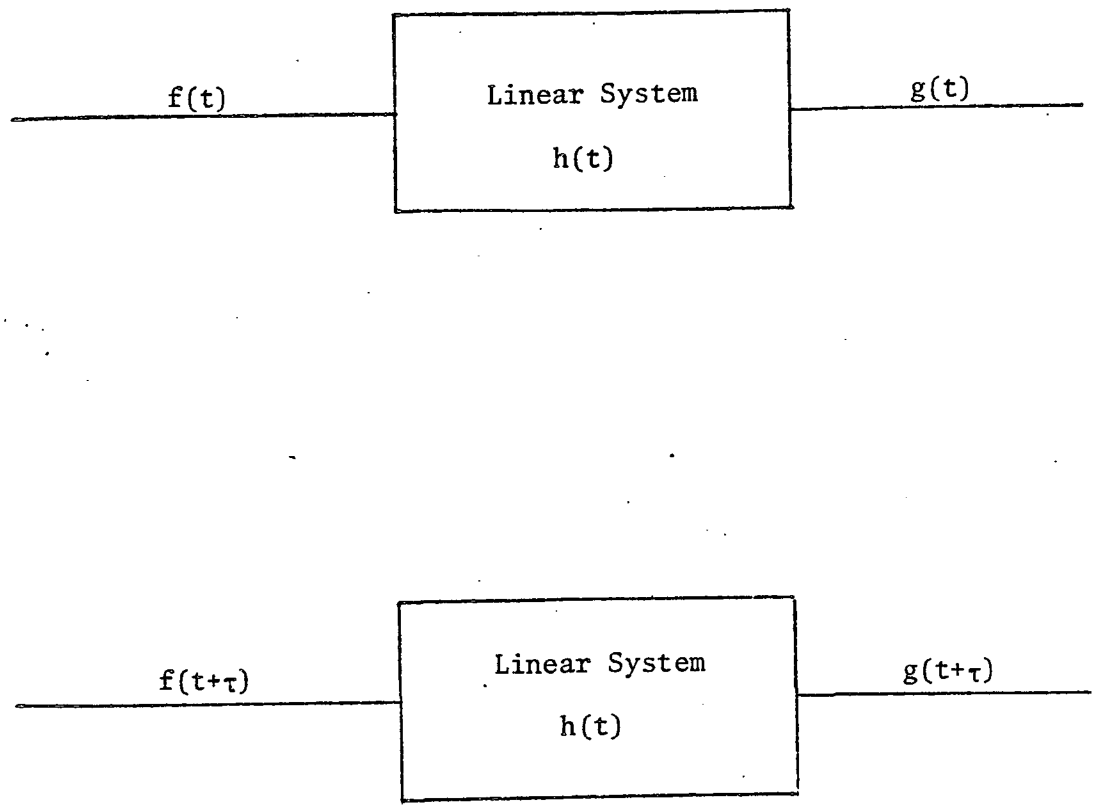
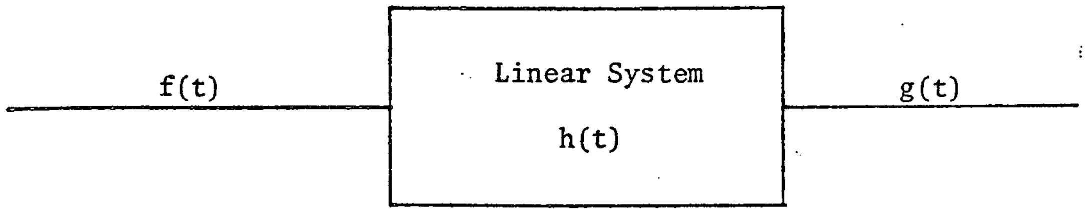


Figure 4-14 A Time-Invariant Linear System



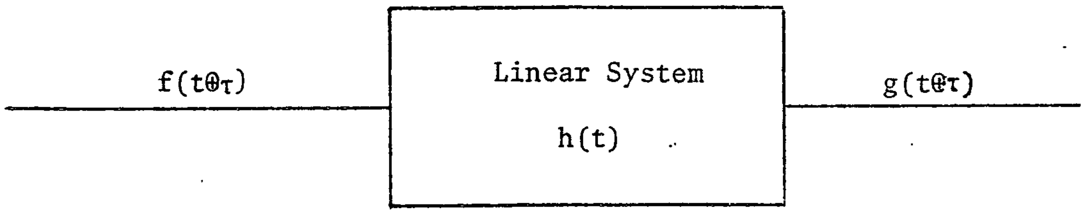


Figure 4-15 A Dyadic-Invariant Linear System

integral (purely time-domain analysis) is preferably avoided in many cases. Indeed, just as transforming to the frequency domain allows a simplification of classical linear system theory, so also does transforming to the sequency domain afford a simpler analysis of dyadic invariant linear systems. Applying the dyadic convolution theorem of equation 1-22 to equation 4-1 and letting $Y(\sigma)$, $H(\sigma)$ and $X(\sigma)$ denote the Walsh-Fourier transforms of $y(t)$, $h(t)$ and $x(t)$ respectively[7].

$$Y(\sigma) = H(\sigma)\ X(\sigma) \tag{4-2}$$

where $H(\sigma)$ is known as the Walsh or sequency transfer function of the dyadic invariant system.

It is not obvious that any classical linear systems possess this dyadic-invariant property. The dyadic-invariant concept as applied to linear systems has been used to develop sequency filters (lowpass, highpass, bandpass). Johnson has presented a rigorous discussion of dyadic-invariant linear systems and has developed the Walsh transfer

[7]The notation $X(\sigma)$ will be used to denote the even and odd Walsh-Fourier transforms of $x(t)$, i.e.

$$X(\sigma) = \begin{cases} X_c(\mu) \\ X_s(\mu) \end{cases}$$

the product of two Walsh-Fourier transforms using this notation will be interpreted to mean

$$H(\sigma)\ X(\sigma) = \begin{cases} H_c(\mu)\ X_c(\mu) \\ H_s(\mu)\ X_s(\mu) \end{cases}$$

the σ notation will be used for convenience.

function concept[8,9]. Johnson also presented a fascinating study of Walsh function analysis of sampled-data systems. A simple sample-data system will be analyzed in the next section based on Johnson's work.

4.3 The Zero-Order Hold as an Ideal Sequency Lowpass Filter

The simplest and most often used data reconstruction filter of sampled-data systems is the zero-order hold (ZOH)[10]. The ZOH is a lowpass filter having a frequency characteristic defined by a $\frac{\sin x}{x}$ relationship. The impulse response of a ZOH is shown in Figure 4-16. The impulse response is usually described by a unit pulse; however, a pulse of amplitude sixteen is used for convenience in the analysis procedure to be presented later.

Johnson has shown that the ZOH is an ideal sequency low pass filter, i.e. a sequency filter whose sequency transfer function is described by Figure 4-17[11,12]. Using this fact, the input-output

[8]Johnson used the Paley-ordered Walsh functions for his work, however Pichler has used sequency ordering to develop the same concept. See References 50 and 69.

[9]At the 1973 NATO conference on signal processing, Pichler stated that he knew of "no naturally occuring system that possessed the dyadic-invariant property." See Reference 70, p. 41.

[10]For a discussion of the zero-order hold and other data reconstruction devices, see Chapter 2 of Reference 54.

[11]Johnson also showed that the ZOH is an example of a dyadic-invariant linear system. See Referencc 50, pp. 87-89.

[12]For a more detailed discussion on sequency filters see References 43, 44, 45, 46, 58, 64, 74 and 80.

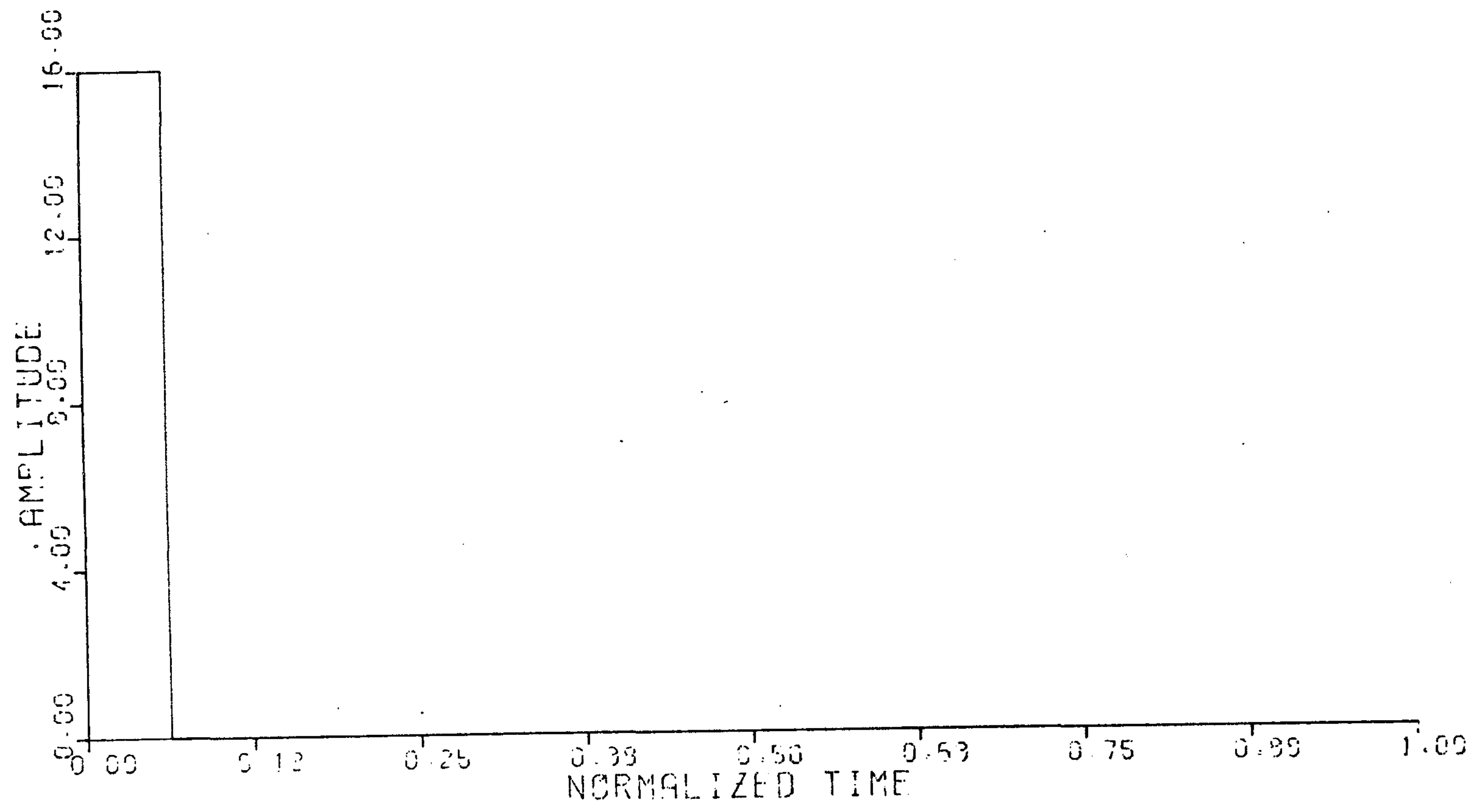


Figure 4-16 Impulse Response of a Zero-Order Hold

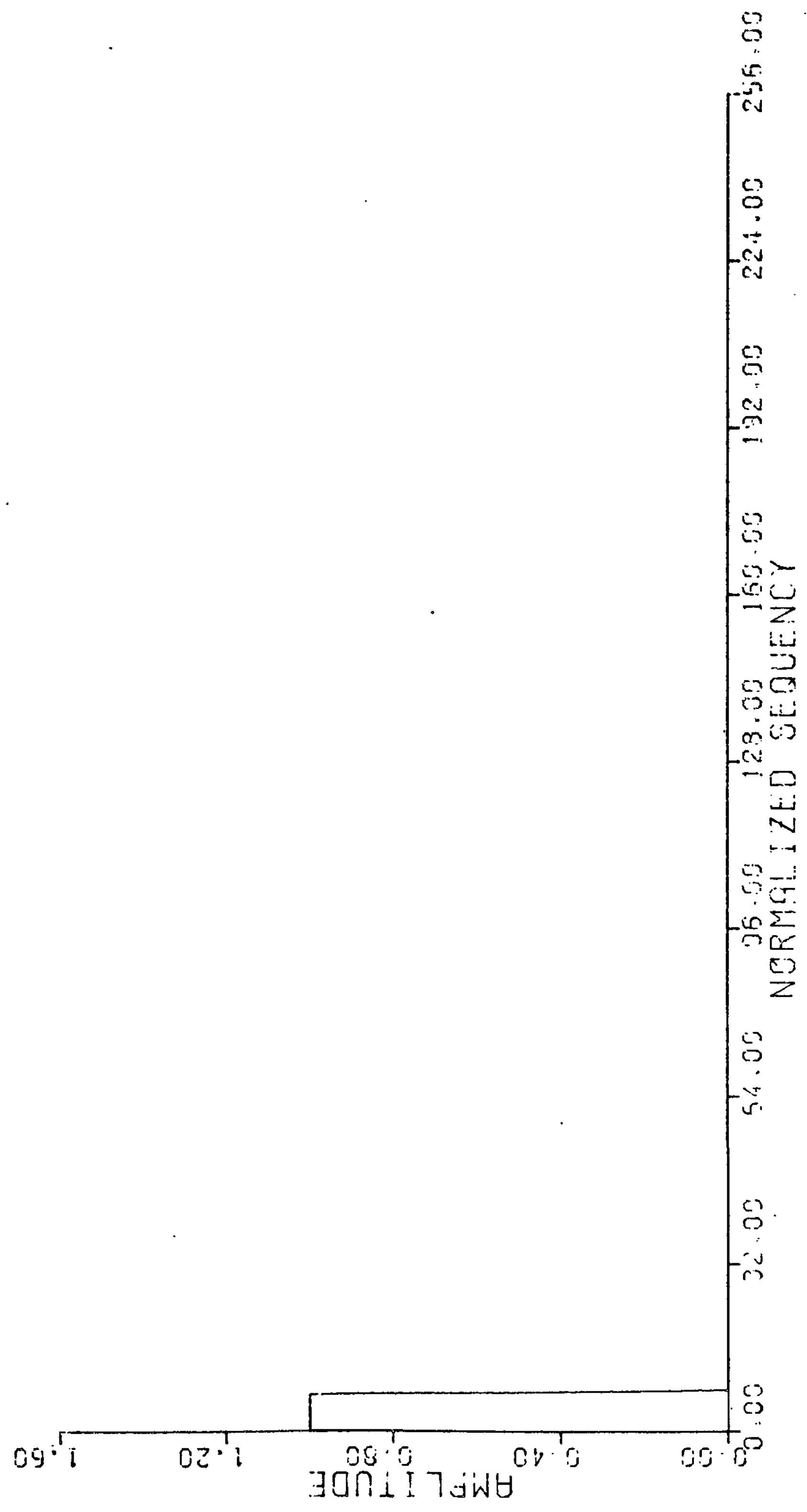


Figure 4-17 An Ideal Sequency Lowpass Filter

relationship for the linear system shown in Figure 4-18 will be analyzed using the Discrete Fourier Transform and the Discrete Walsh Transform. The sampler shown in Figure 4-18 is an ideal (impulse) sampler and the impulse response of the ZOH is assumed to be described by Figure 4-16.[13] The sampling period will be assumed to be 1/16 of a normalized second. This leads to a cutoff sequency of 8 for the ZOH (see Figure 4-17). The sampling theorem for sequency analysis is almost identical to that of the sampling theorem for frequency analysis, i.e. if a signal is sequency-limited to a sequency of B zps that signal must be sampled at a rate of at least 2B samples per second.[14]

The analysis of the system of Figure 4-18 was performed using the Discrete Walsh Transform and the Discrete Fourier Transform discussed in Chapter 3. The subroutines FFT1 and FWT4 were used to perform the necessary calculations. From the material presented in Chapter 3, one notes that the DWT and DFT can be used for handling only discrete data. However, the input and output functions of the system [f(t) and h(t)] are in general continuous. Besides this problem, the sampler is assumed to be an ideal sampler whose output f*(t) is a series of Dirac delta functions. One Dirac delta function occurs every 1/16 of a normalized second. This presents the dilemma of simulating continuous time functions and an ideal sampler using a digital computer.

---

[13]All the describing variables (time, frequency and sequency) will be assumed to be normalized.

[14]At present there is a debate in the literature whether or not the sampling theorem has a dyadic nature to it, therefore not allowing a general sampling principle to be stated. See References 43, 50, 51 and 62.

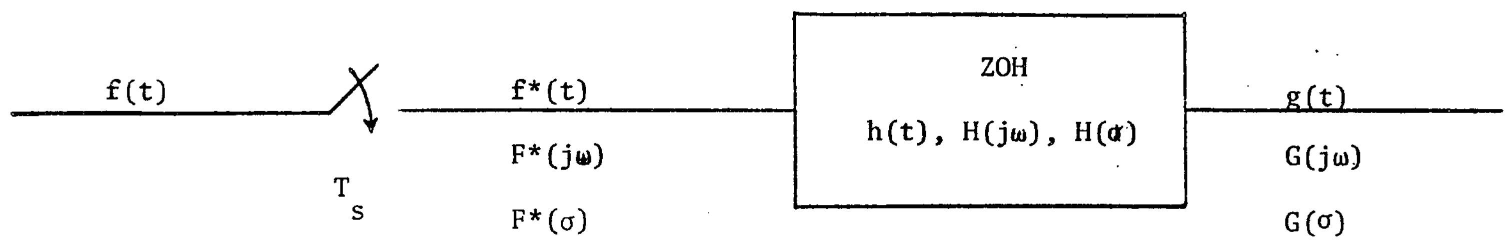


Figure 4-18 Sampler and Zero-Order Hold System

The method used to simulate the continuous time function was that of dividing the unit interval up to 512 points. It was assumed that by describing the input and output function by 512 points along the unit interval that a reasonably good approximation to a continuous function could be made. The number of points chosen was such that it was much greater that the desired sampling time of the system, i.e. the input-output functions are described every 1/512 of a normalized second whereas the sampler is running every 1/16 of a normalized second.

The final problem is that of simulating the ideal sampler. From the discussion presented in Chapter 3, the discrete points used in the DWT or DFT can be assumed to be Dirac delta functions[15], therefore using this concept, the sampler can be simulated directly by allowing the output of the sampler be equal to a modified version of the 512 point discrete input functions discussed above. Since the input functions and output functions are described for 512 points along the unit interval, the output of the sampler must also be described for 512 points, i.e. the output of the sampler must be defined to be some value at every 1/512 of a normalized second. But the sampler is running every 1/16 second, hence the question arrises what is the output of an ideal sampler between sampling instants? The answer is that output of the sampler is defined to be identically zero between sampling instants[16]. Hence, the sampler was simulated by allowing its output to be f(t) every 1/16 of a normalized second and being zero elsewhere.

[15]The actual Dirac delta functions appear in the derivation of the DWT or DFT from the continuous Fourier transform.

[16]See Chapters 1 and 2 of Reference 54.

Using this scheme the array of numbers used to describe the output of the sampler for the needed 512 points was one of having one data point equal to the value of f(t) at the sampling instance, followed by 32 zeroes, then another data point followed by 32 zeroes, etc. This allowed 16 samples of the input function every unit interval. The 32 zeroes come about because of the necessity of defining the output of the sampler every 1/512 of a normalized second.

Once the sampled signal f*(t) was constructed using the above scheme, the analysis proceeded by first obtaining the DWT or DFT of f*(t), then the transformed signal was multiplied by its respective frequency or sequency transfer function of the ZOH. Finally, the inverse transform was obtained and g(t) was plotted. A flow chart for the FORTRAN mainline program used to implement this analysis procedure is shown in Figures 4-19 and 4-20. The sequency and frequency transfer functions of the ZOH were obtained by computing the DWT and DFT of the impulse response shown in Figure 4-16. The sequency and frequency transfer functions are shown in Figures 4-17 and 4-21 respectively[17]. From Figure 4-17 the sequency transfer function is unity for the first eight sequencies and zero thereafter. This fact was used in that it was not necessary to multiply the transformed sampled time function by the sequency transfer function in the analysis procedure. The only thing required was to set all the sequency coefficients above eight to zero and then immediately do the inverse transformation. This saved

[17]Figure 4-21 is the magnitude of the transfer function. Figure 4-21 shows the folding properties of the output of DFT. Using the 512-point transform only 256 unique frequency coefficients were resolved.

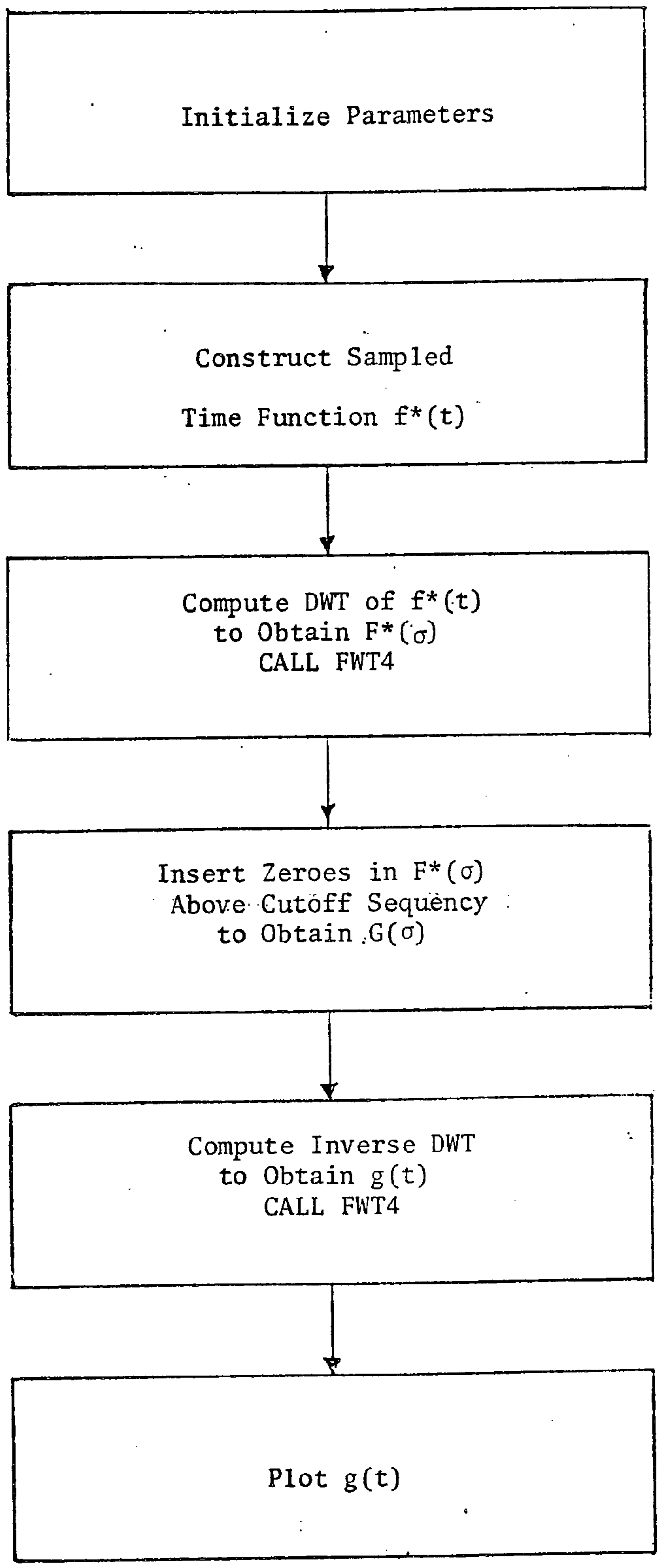

Initialize Parameters
Construct Sampled Time Function f*(t)
Compute DWT of f*(t) to Obtain F*(σ) CALL FWT4
Insert Zeroes in F*(σ) Above Cutoff Sequency to Obtain G(σ)
Compute Inverse DWT to Obtain g(t) CALL FWT4
Plot g(t)

Figure 4-19 Flow Chart for Analysis of ZOH Using Sequency Transfer Function

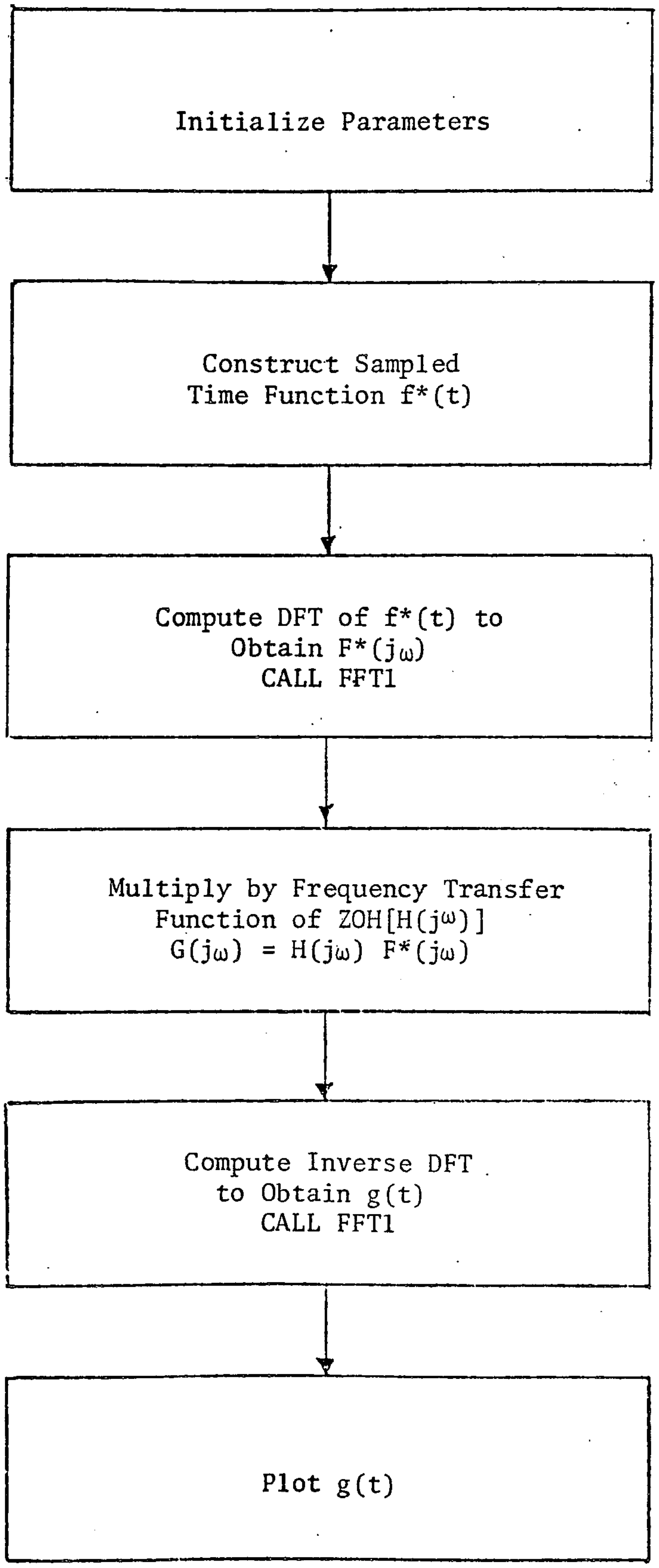

Initialize Parameters
Construct Sampled Time Function f*(t)
Compute DFT of f*(t) to Obtain F*(jω) CALL FFT1
Multiply by Frequency Transfer Function of ZOH[H(jω)] G(jω) = H(jω) F*(jω)
Compute Inverse DFT to Obtain g(t) CALL FFT1
Plot g(t)

Figure 4-20 Flow Chart for Analysis of ZOH Using Frequency Transfer Function

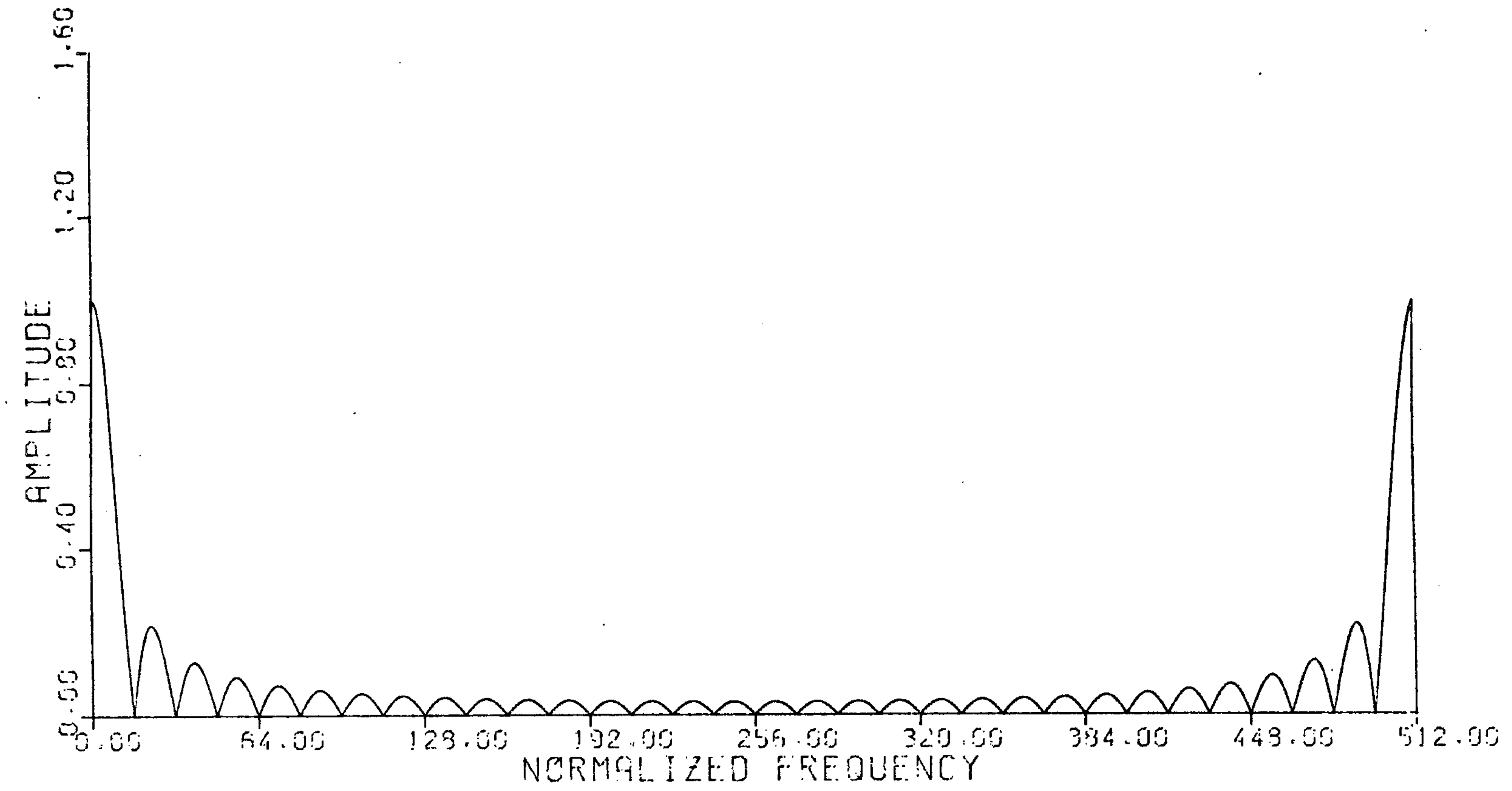


Figure 4-21 Frequency Transfer Function of ZOH Described in Figure 4-16

time in analyzing the system using the sequency transfer function and that is the reason why the amplitude of the impulse response was chosen to being 16 and not unity. This also demonstrates one of the advantages of using the sequency transfer concept for analyzing this system. By analyzing the system using discrete transformation techniques one is assuming that the discrete convolution theorem (dyadic and arithmetic) are good approximations in this case of the integral convolution theorem. Another way to think of this system is that of a digital sequency filter; hence using the discrete convolution theorem to analyze it.

The system of Figure 4-18 was analyzed using the sequency and frequency techniques discussed above for five different input functions. The input waveforms were the five waveforms synthesized using the Walsh Function generator.(see Figures 4-3, 4-5, 4-7, 4-10 and 4-12). The results are shown in Figures 4-22 through 4-31[18]. The results obtained using the sequency transfer function technique are <u>identical</u> to that obtained using standard frequency techniques. The outputs of the ZOH were exactly what one would expect using such a device.

The impulse response for the system was chosen such that the sequency filter would cut off at a sequency that was compatible with the sequency-limited signals synthesized in section 4.1. By comparing Figures 4-22 through 4-31 to Figure 4-2 it is obvious that the waveshapes are "similar" but not identical. This is due to the fact that synthesized waveforms did not come about from some sampling process, hence these

[18]The outputs are attenuated from what one would expect given this particular system. This is caused by the way the sampler was simulated and is due to something known as the "stretch" phenomena. See References 2 and 26

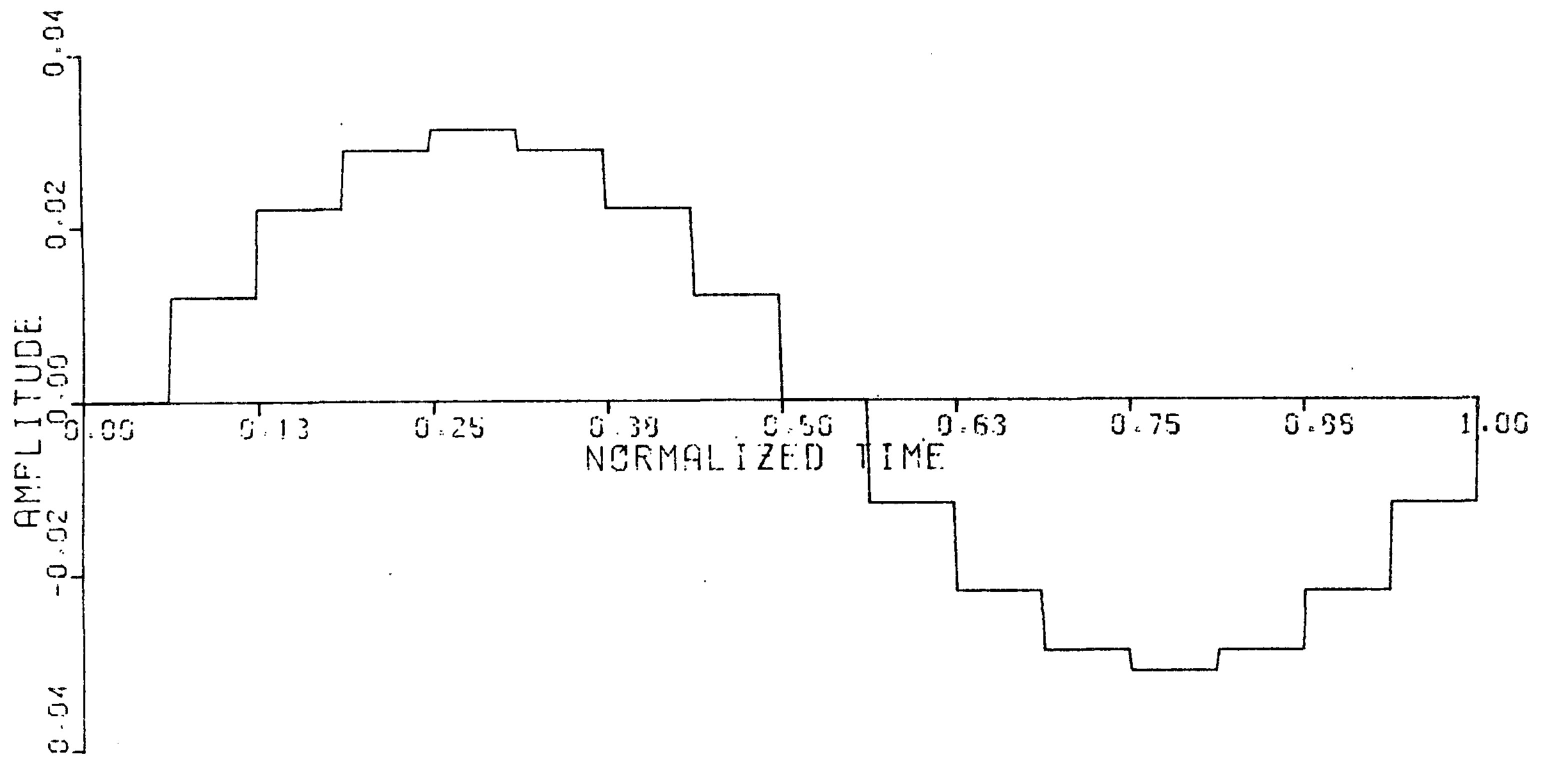


Figure 4-22 Output of ZOH Using Sequency Transfer Function (Sine-Wave Input)

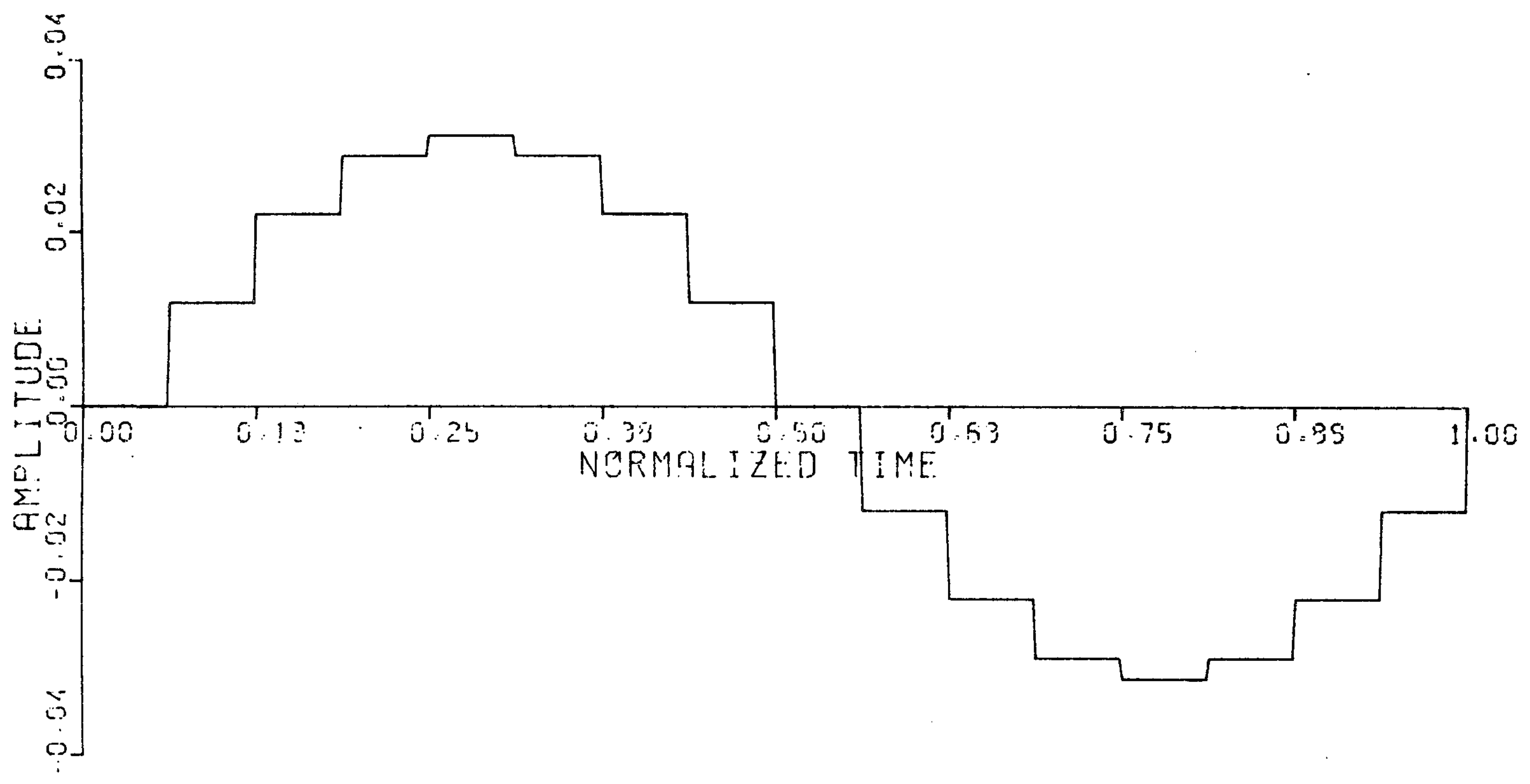


Figure 4-23 Output of ZOH Using Frequency Transfer Function (Sine-Wave Input)

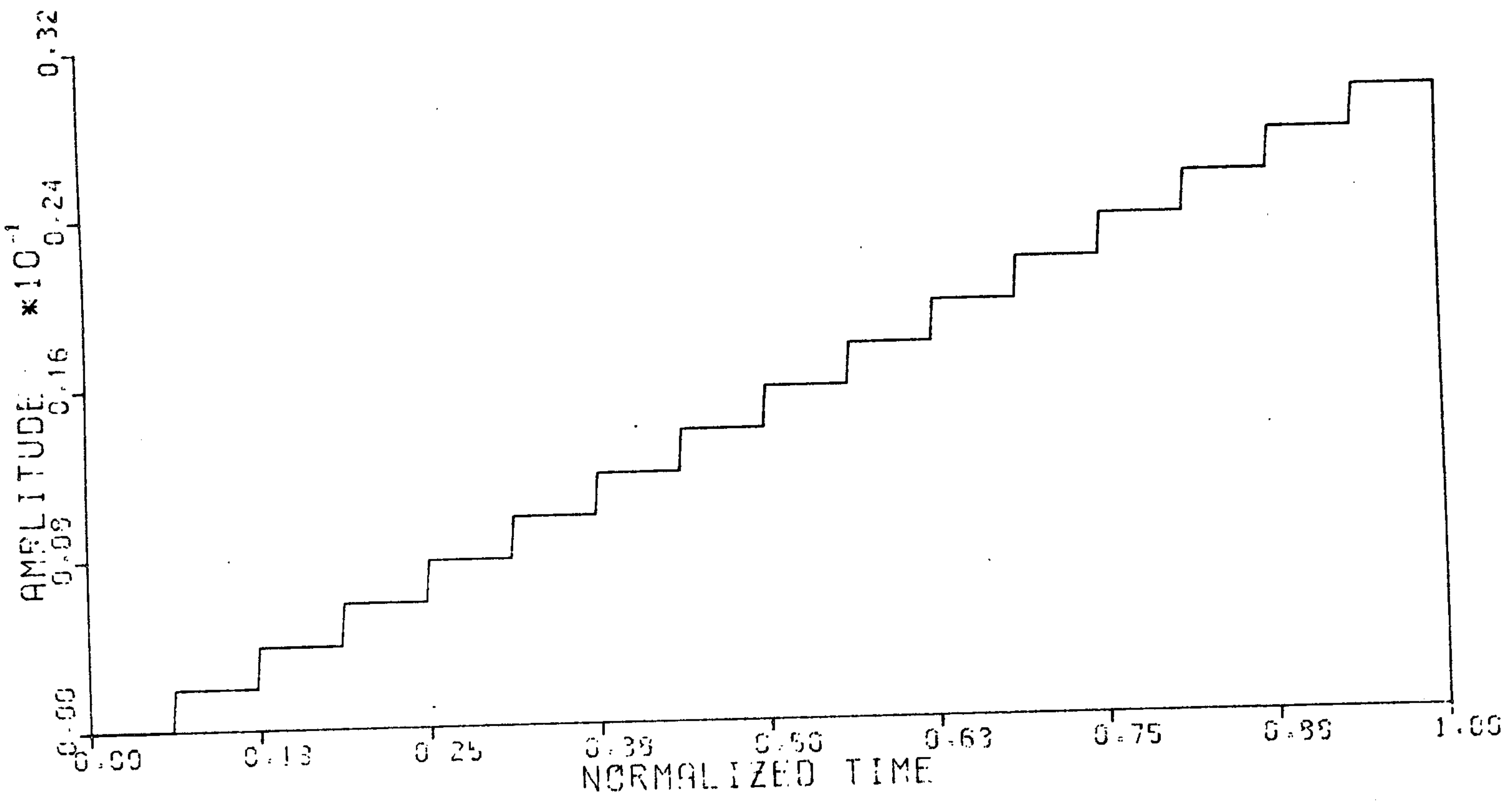


Figure 4-24 Output of ZOH Using Sequency Transfer Function (Ramp Input)

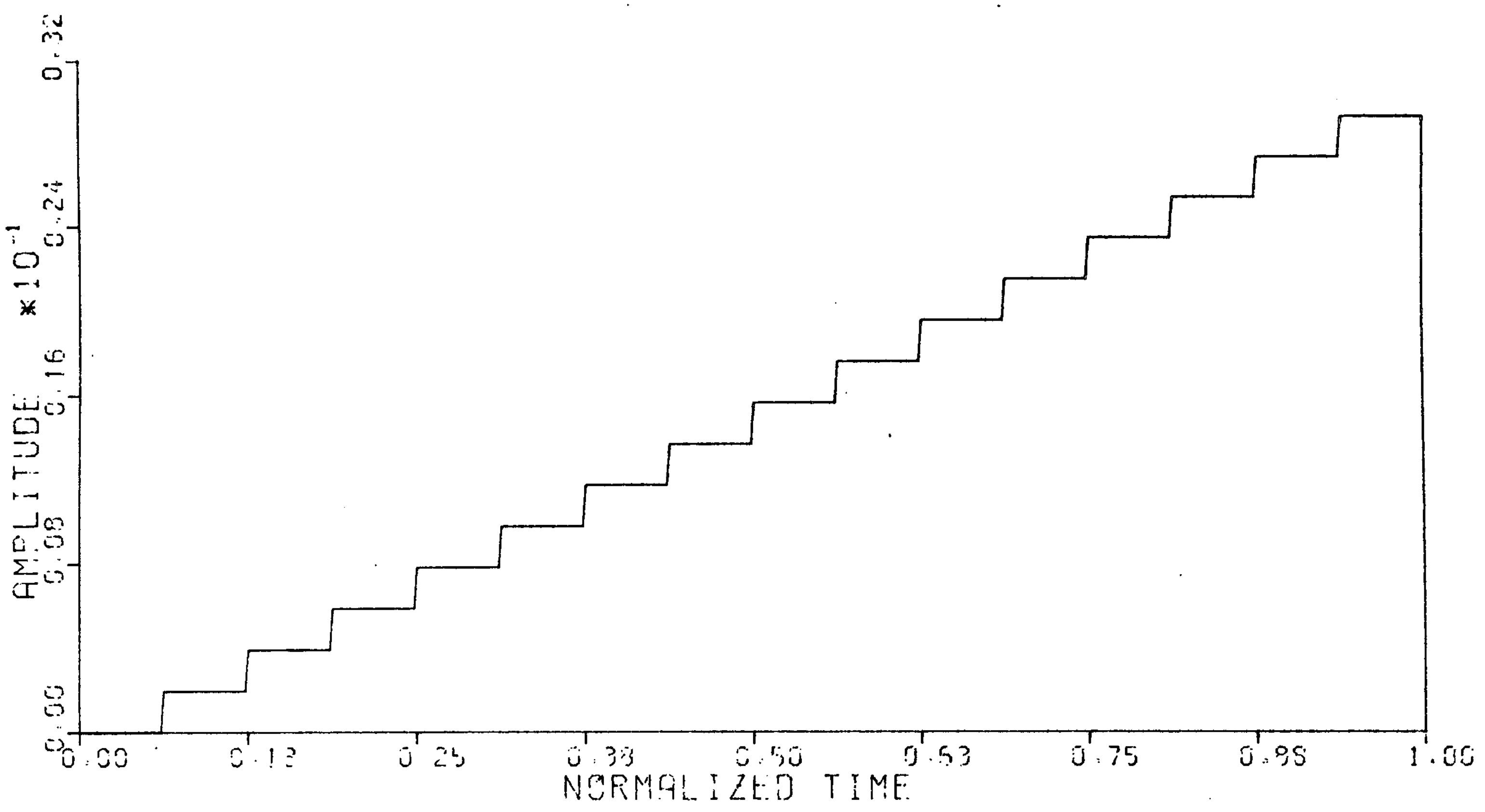


Figure 4-25 Output of ZOH Using Frequency Transfer Function (Ramp Input)

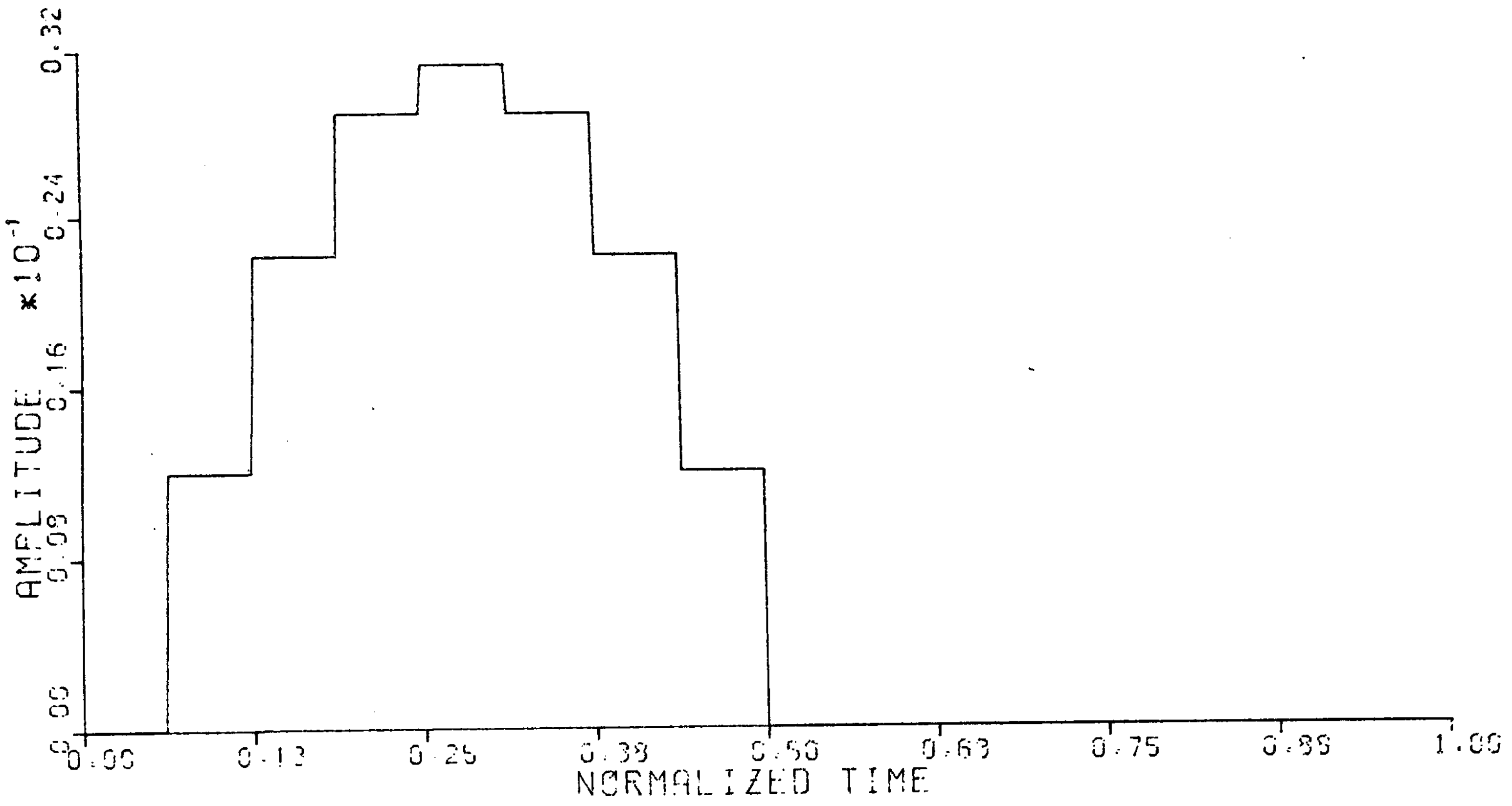


Figure 4-26 Output of ZOH Using Sequency Transfer Function (Half-Wave Rectified Sine-Wave Input)

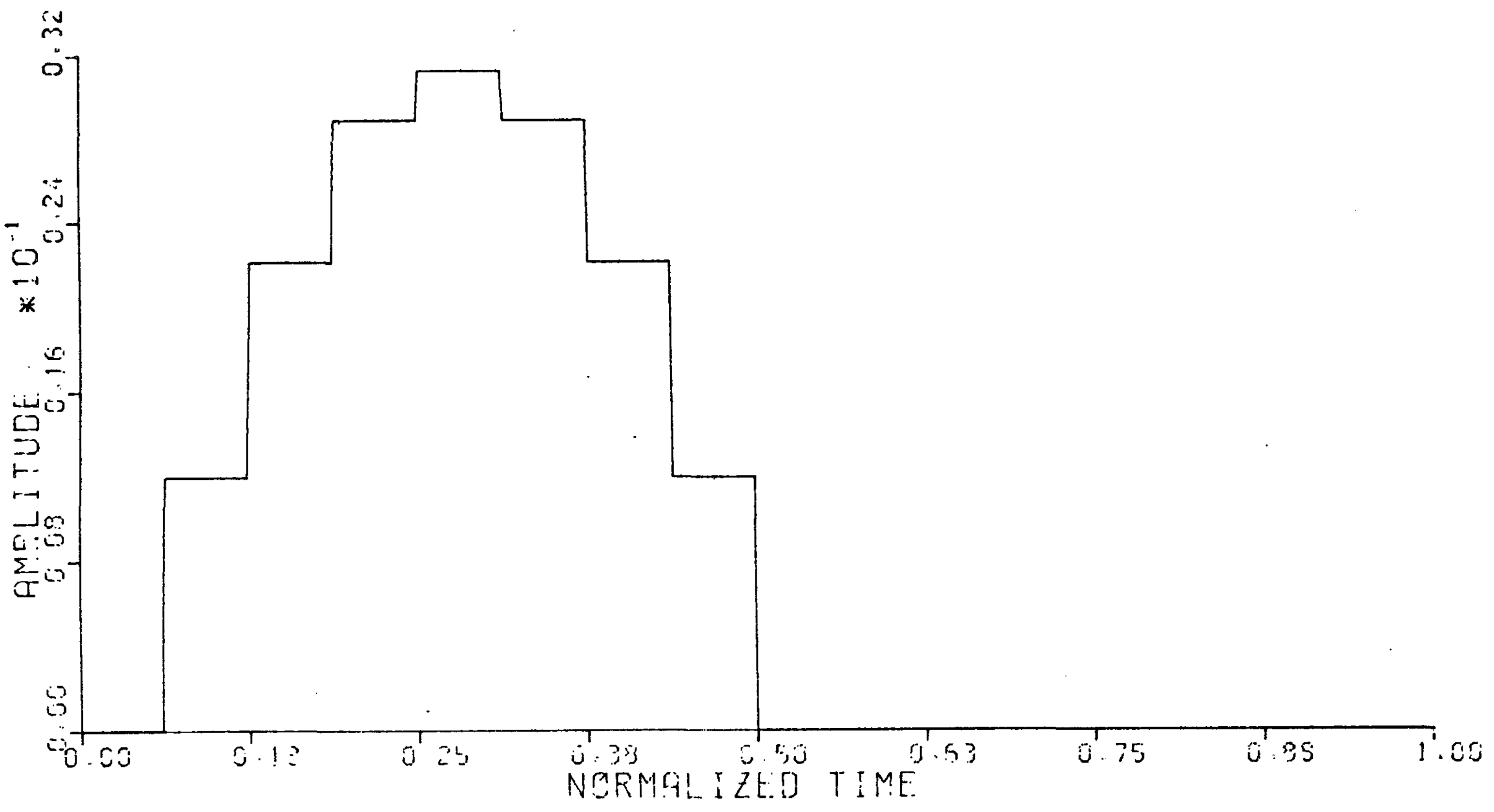


Figure 4-27 Output of ZOH Using Frequency Transfer Function (Half-Wave Rectified Sine-Wave Input)

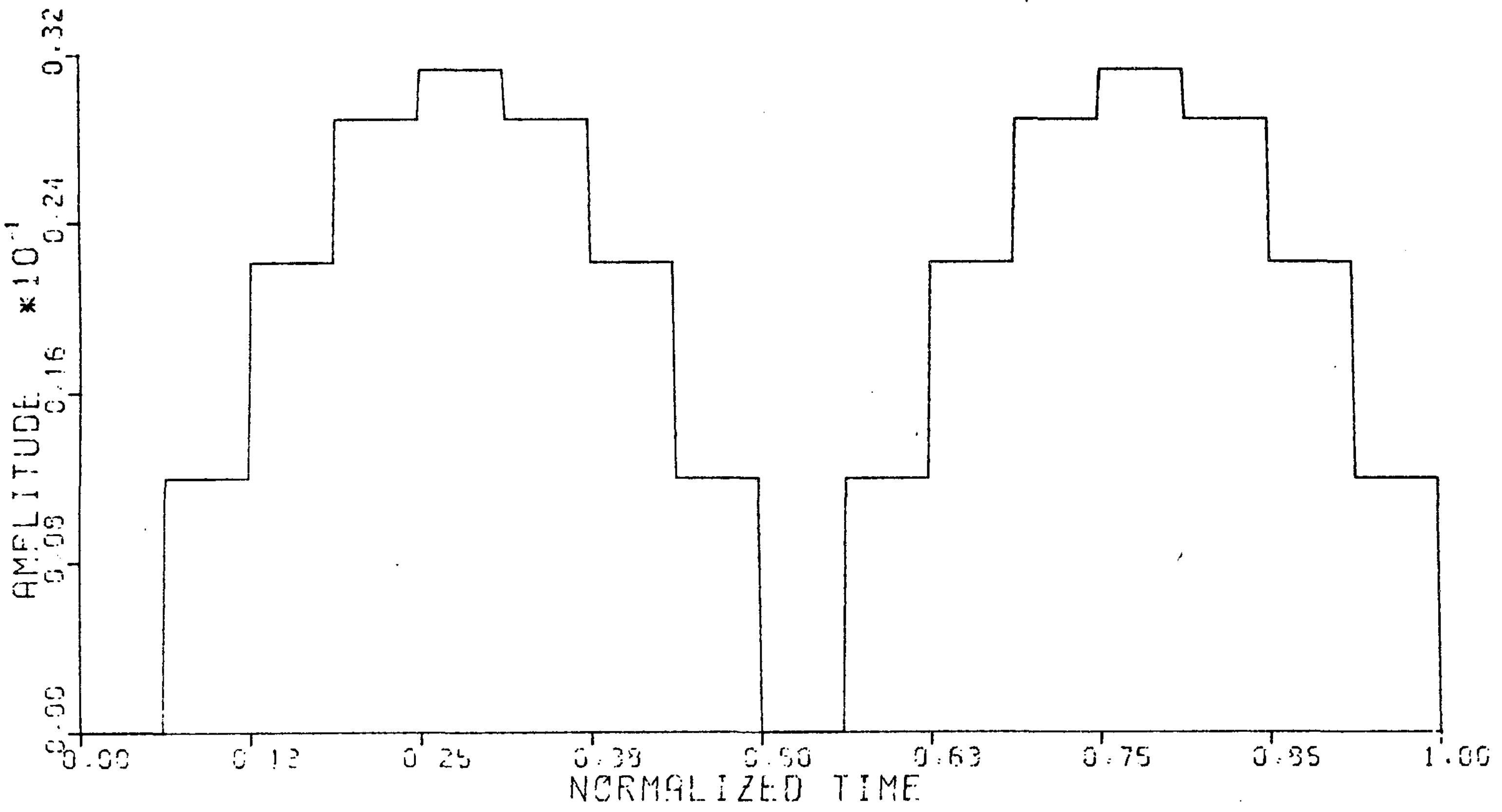


Figure 4-28 Output of ZOH Using Sequency Transfer Function (Full-Wave Rectified Sine Wave Input

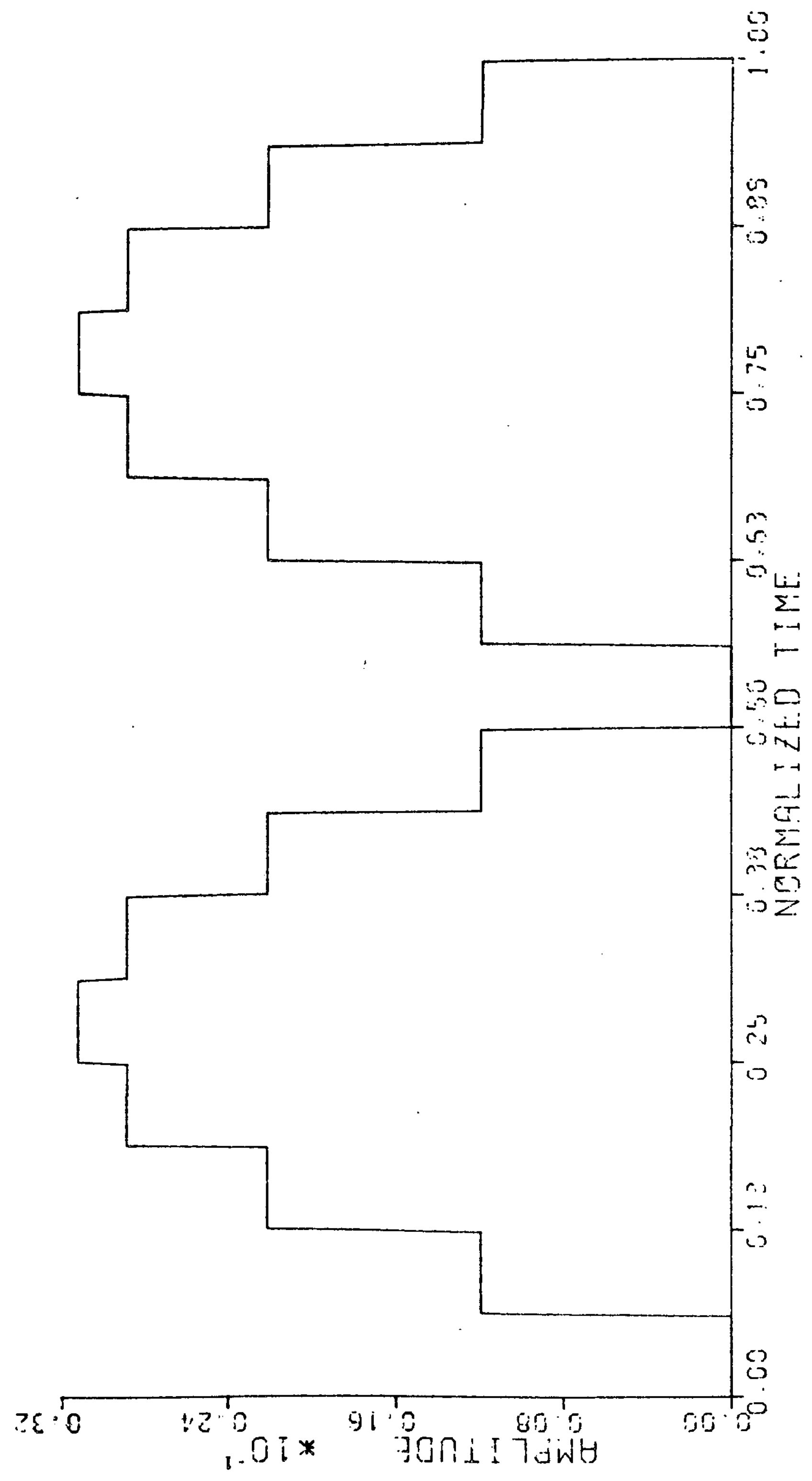


Figure 4-29 Output of ZOH Using Frequency Transfer Function (Full-Wave Rectified Sine-Wave Input)

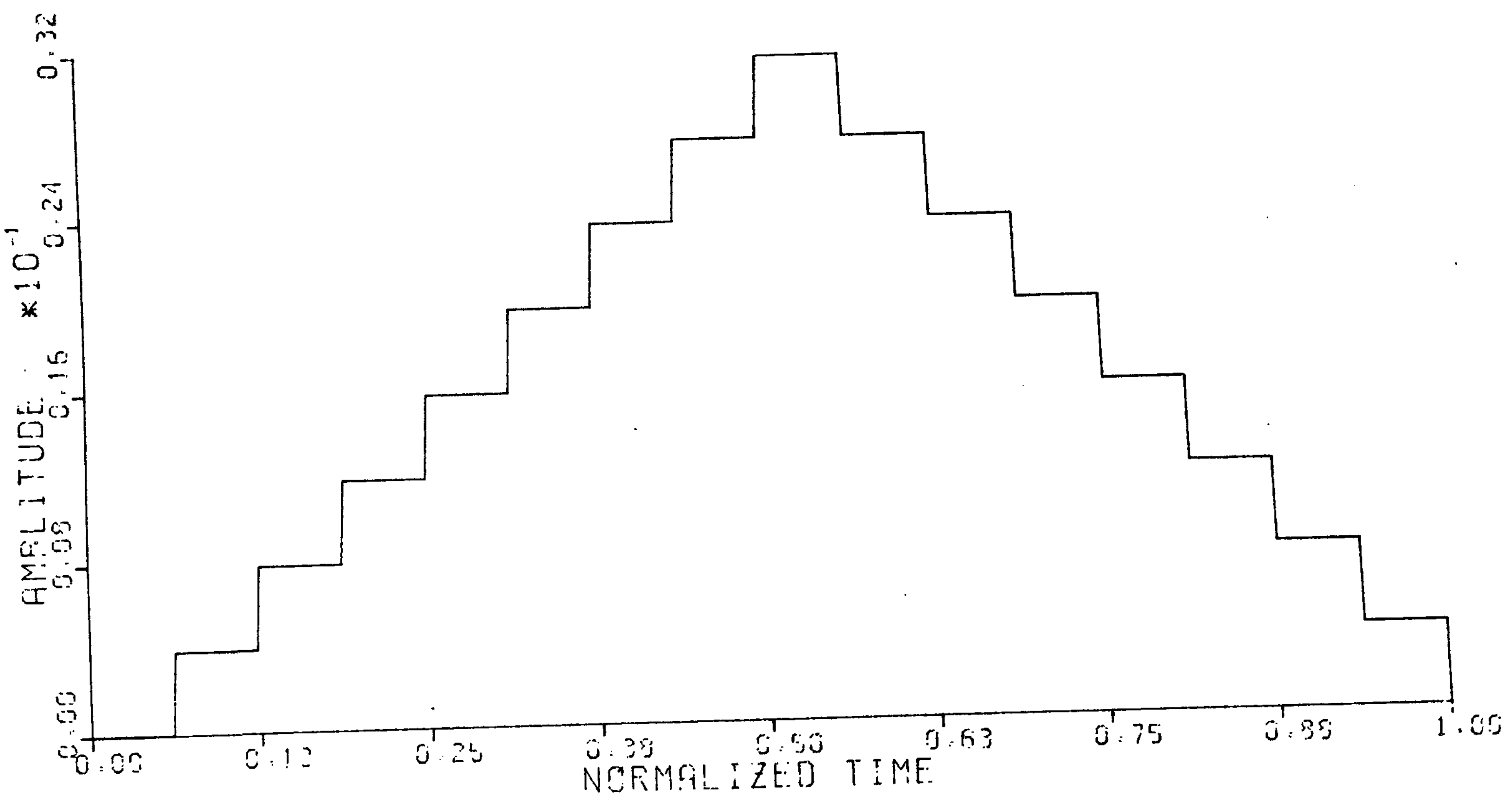


Figure 4-30 Output of ZOH Using Sequency Transfer Function (Triangular Function Input)

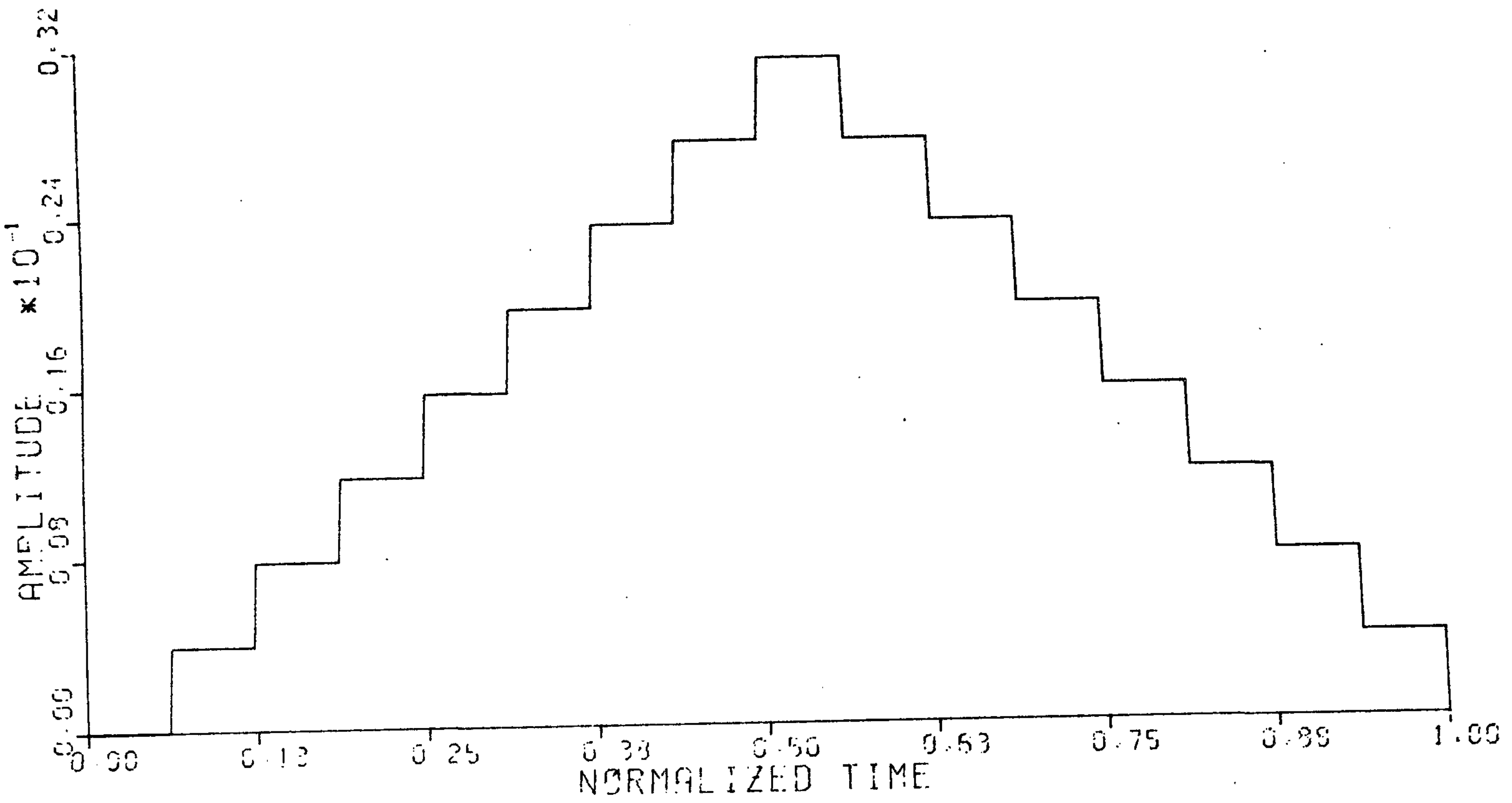


Figure 4-31 Output of ZOH Using Frequency Transfer Function (Triangular Function Input)

waveshapes should not be considered the output of a hold device.

In conclusion, a sequency transfer function has been used to describe the input-output relationship of a simple dyadic-invariant system. The system used was that of a familiar time-invariant linear system. In general, to use the sequency transfer function concept one must first show that the system is dyadic-invariant, i.e. that input-output relationship is described by the dyadic convolution integral. Demonstrating dyadic-invariance is rather complicated to do. This fact has led to problems in trying to develop a linear system theory using Walsh functions. An advantage of using Walsh functions to analyze the system of Figure 4-18 is that it was not necessary to do any complex multiplication operations to evaluate the convolution product. In fact the operation required was just that of setting the sequency coefficients above the cutoff value to be zero and immediately performing the inverse transformation. Another advantage is the time saved due to the relative speed of the FWT as opposed to the FFT.[19]

[19] The dyadic-invariant system concept seems to contribute more in the area of discrete-data or sample-data systems. For more details on Walsh function system analysis, see References 1, 18, 21, 23, 37, 50 and 81.

Chapter 5

CONCLUDING REMARKS

5.0 Introduction

The purpose of this chapter is to discuss briefly some disadvantages to using Walsh functions. The disadvantages will mainly lie in the area of truncation errors and cyclic shift problems.

The results of this thesis will be quickly reviewed and topics for further investigation will be presented in the second half of this chapter.

5.1 Disadvantages of Using Walsh Functions[1]

Walsh functions do have disadvantages such that in some applications sine-cosine Fourier analysis techniques are vastly superior. One of the disadvantages to using Walsh-Fourier techniques is that an arithmetic time shift theorem does not exist for the Walsh-Fourier transform[2, 3]. If a given time function is shifted by a finite amount, the magnitude of its sine-cosine Fourier Transform is not altered by the time shift. The difference between the shifted and original signal is a phase constant related to the time shift. This is not so for the Walsh-Fourier transform. This difficulty may be attributed to the fact that after time

[1] This entire section is based on an excellent paper by Blachman describing some of the problems encountered when using Walsh functions. See Reference 13.

[2] For the Discrete Fourier Transform, the arithmetic time shift theorem is sometimes referred to as the "modulo N shift property".

[3] For a discussion of the shift theorem see Reference 17, p. 104-107.

shifting, a Walsh function generally becomes the sum of an infinite number of Walsh functions while a sinusoid simply turns into the sum of a sine and cosine of the same frequency. Thus a change in the time scale or a shift of the time origin usually will grossly alter a Walsh spectrum but has only a minor effect (a phase shift) on the standard Fourier spectrum.

There is a dyadic time shift theorem[4] for the Walsh-Fourier transform; however, it like all of the dyadic properties of Walsh functions is somewhat alien to the average engineer. This shift property can be related to the dyadic convolution property discussed in Chapter 1, 3 and 4. This lack of a shift property implies that to use the Walsh-Fourier Transform, or the Discrete Walsh Transform, one must have very good synchronization such that the time scale or time origin is not changed in any way. This synchronization in some applications can not be adequately maintained; hence Walsh techniques can not be used.

There are two sources of error encountered when using the discrete transforms discussed in Chapter 3, truncation error and roundoff error. Truncation error is due to the fact that only a finite number of terms can actually be used for a particular time function. Roundoff error is due to using only a finite number of digits in representing the coefficients of the transform. Blachman has shown, taking into account both truncation and roundoff errors, that a given sequency-limited time function in the absence of exact synchronization requires $\pi^2/6$ times as many Walsh coefficients to standard Fourier coefficients for the same rms error.

[4]See Reference 50, p. 24.

A sequency-limited time function has a staircase-like appearance (see Figure 4-2). The Walsh analysis above was not synchronized with the "jumps" of the time function and hence $\frac{\pi^2}{6}$ more coefficients were needed. Blachman has presented similar results when assuming that the time function is continuous.

In general, it usually requires more Walsh coefficients to represent a time function than standard Fourier coefficients given the same rms error. This fact was noted in section 4.1 when discussing why the sine wave appeared to have even Walsh coefficients (see Table 4-1). The larger error property has the effect of nullifying part of the advantage gained by the relative speed of the Discrete Walsh Transform, i.e. the Walsh coefficents can be calculated faster but have a larger rms error associated to them. Hence, there is a trade-off between relative speed and error when using the Discrete Walsh Transform as opposed to the Discrete Fourier Transform.

Blachman has shown that these errors can be attributed to the fact that the sine-cosine Fourier series tends to approximate, in some cases very accurately, the optimum Fourier series known as the Karhunen-Loene expansion[5]. The Walsh-Fourier series does not approximate the Karhunen-Loene expansion as accurately and in fact the Walsh coefficients tend to be asymptomatically correlated which accounts for the greater truncation errors.

[5]The Karhunen-Loene expansion is optimal in the sense that the Fourier coefficients for this expansion are uncorrelated with each other. See Reference 13, p. 351.

The above discussion is not meant to imply that Walsh functions should be discarded as having no applications in engineering work. On the contrary, Walsh functions have been used in many applications with excellent success[6]. The above discussion was presented to indicate that Walsh functions do have some disadvantages and in general they are somewhat sub-optimal to the standard sine-cosine set[7].

5.2 Summary of Results

This thesis has demonstrated the usefulness of Walsh functions in the area of spectral analysis and synthesis. A Walsh function generator design has been presented and waveform synthesis has been accomplished using this generator.

A standard Fast Fourier Transform subroutine has been modified to compute various versions of the Fast Walsh Transform. The input-output relationship of a simple sample-data system has been analyzed and identical results obtained using either the frequency or sequency transfer function of the system.

5.3 Topics for Further Research

The topics for further research in the Walsh function area or related areas will be listed below in two groups. The first group will be Walsh function topics and the second group will be related topics in the signal processing area.

---

[6]A sequency-division multiplex system has been in operation in the West German telephone system for about four years. For other applications see References 33, 42, 43, 44, 45, 46, 56, 58, 84, and 88.

[7]Essentially Chapters 2 through 4 present the advantages to Walsh functions.

Walsh function area:

1. Development of a two-dimensional Fast Walsh Transform subroutine and investigation of topics related to picture processing using this subroutine. The two dimensional dyadic convolution property should be investigated.
2. Further investigation of dyadic-invariant linear systems to ascertain if the sequency transfer function concept can be used to describe more complex practical engineering systems.
3. The application of continuous and digital sequency filters to the communications area.
4. Investigation of electromagnetic Walsh waves and their applications to optics and communications.
5. Development of a set of Walsh function experiments for undergraduate students to demonstrate "alternate" methods of Fourier analysis.

Signal processing area:

1. Investigation of other sets of orthogonal functions and their convolution properties, particularly the set of Haar functions and the set of Slant functions. Discrete Fourier Transforms can be computed using these functions by suitable variations of a Cooley-Tukey algorithm. Two-dimensional versions of these Fourier transforms have been used for picture processing.
2. Further investigation of the Generalized Discrete Fourier Transform and its properties.

APPENDIX

```
PAGE   1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0018        0018        0000

V2 M11   ACTUAL 16K  CONFIG 16K

// FOR
*LIST ALL
*ONE WORD INTEGERS
      SUBROUTINE FWT1 (MR,N,M)
C     THIS SUBRONTINE COMPUTES A FAST HADAMARD TRANSFORM.THE ALGORITHM
C     IS BASED ON THE FACTORING OF THE HADAMARD MATRIX USING BINARY
C     INDEXING AND KRONECKER PRODUCT OPERATION. THIS PROGRAM IS TAKEN
C     FROM ''COMPUTATION OF THE FAST HADAMARD TRANSFORM'' BY Y.Y. SHUM
C     AND A.R.ELLIOTT&PROCEEDINGS OF SYMPOSIUM ON APPLICATIONS OF WALSH
C     FUNCTIONS 1972,NATIONAL TECHNICAL INFORMATION SERVICE,AD-744-650,
C     SPRINGFIELD,VA.,PP.177-180.
C
C
C     MR-THIS VECTOR CONTAINS THE SAMPLE VALUES TO BE TRANSFORMED.
C     N-THIS IS THE DIMENSION OF THE VECTOR MR.
C     M- THIS IS THE POWER OF 2 THAT N IS EQUAL,I.E. N=2**M.
C
C     SUBROUNTINE FWT1 COMPUTES ONLY THE FORWARD HADAMARD TRANSFORM
C     OF THE INPUT DATA. TO COMPUTE THE INVERSE TRANSFORM SEE SUBROUTINE
C     FWT2. THE OUTPUT OF THIS SUBROUTINE IS ALREADY NORMALIZED SO IT
C     IS NOT NECESSARY TO DIVIDE THE OUTPUT COEFFICIENTS BY N.
C
C     NOTE&THE TRANSFORM COEFFICIENTS ARE RETURNED FROM FWT1 IN THE
C     VECTOR MR.    THE INPUT DATA IS DESTROYED.
C
C
      REAL MR(1)
C
      L=N
      K=1
      DO 3 NM=1,M
      I=0
      L=L/2
      DO 2 NL=1,L
      DO 1 NK=1,K
      I=I+1
      J=I+K
      MR(I)=(MR(I)+MR(J))/2
    1 MR(J)=MR(I)-MR(J)
    2 I=J
    3 K=K*2
      RETURN
      END
 VARIABLE ALLOCATIONS
      L(I )=0002          K(I )=0003          NM(I )=0004          I(I )=0005
      J(I )=0008

 STATEMENT ALLOCATIONS
  1     =0065  2     =0078  3     =0085
```

```
PAGE   1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0018        0018        0000

V2 M11   ACTUAL 16K  CONFIG 16K

// FOR
*LIST ALL
*ONE WORD INTEGERS
      SUBROUTINE FWT2 (MR,N,M,IFSET)
C     THIS SUBRONTINE COMPUTES A FAST HADAMARD TRANSFORM.THE ALGORITHM
C     IS BASED ON THE FACTORING OF THE HADAMARD MATRIX USING BINARY
C     INDEXING AND KRONECKER PRODUCT OPERATION. THIS PROGRAM IS TAKEN
C     FROM ''COMPUTATION OF THE FAST HADAMARD TRANSFORM'' BY Y.Y. SHUM
C     AND A.R.ELLIOTT&PROCEEDINGS OF SYMPOSIUM ON APPLICATIONS OF WALSH
C     FUNCTIONS 1972.NATIONAL TECHNICAL INFORMATION SERVICE,AD-744-650,
C     SPRINGFIELD,VA.,PP.177-180.
C
C
C     MR-THIS VECTOR CONTAINS THE SAMPLE VALUES TO BE TRANSFORMED.
C     N-THIS IS THE DIMENSION OF THE VECTOR MR.
C     M- THIS IS THE POWER OF 2 THAT N IS EQUAL,I.E. N=2**M.
C     IFSET-THIS FLAG DETERMINES WHETHER A FORWARD OR INVERSE TRANSFORM
C            IS TO BE COMPUTED. IFSET=-1  AN INVERSE TRANSFORM IS COM-
C            PUTED. IFSET=1  A FORWARD TRANSFORM IS COMPUTED.
C
C
C     THE OUTPUT OF THIS SUBROUTINE IS ALREADY NORMALIZED SO IT IS NOT
C     NECESSARY TO NORMALIZE BEFORE OR AFTER THE SUBROUTINE IS CALLED.
C
C     NOTE&THE TRANSFORM COEFFICIENTS ARE RETURNED FROM FWT2 IN THE
C     VECTOR MR.    THE INPUT DATA IS DESTROYED.
C
C
C
      REAL MR(1)
      L=N
      K=1
      DO 5 NM=1,M
      I=0
      L=L/2
      DO 4 NL=1,L
      DO 3 NK=1,K
      I=I+1
      J=I+K
      IF(IFSET) 1,2,2
    1 MR(I)=MR(I)+MR(J)
      MR(J)=MR(I)-MR(J)-MR(J)
      GO TO 3
    2 MR(I)=(MR(I)+MR(J))/2
      MR(J)=MR(I)-MR(J)
    3 CONTINUE
    4 I=J
    5 K=K*2
```

```
PAGE   2

      RETURN                                                                      FWT2 460
      END                                                                         FWT2 470
VARIABLE ALLOCATIONS
     L(I )=0002           K(I )=0003          NM(I )=0004           I(I )=0005
     J(I )=0008

STATEMENT ALLOCATIONS
 1    =0055  2    =0079  3    =00A0  4    =00A9  5    =0086

FEATURES SUPPORTED
 ONE WORD INTEGERS

CALLED SUBPROGRAMS
 FADDX    FSUBX    FLDX     FSTO     FSTOX    FDVR     FLOAT    SUBSC    SUBIN

INTEGER CONSTANTS
     1=000C      0=000D       2=000E

CORE REQUIREMENTS FOR FWT2
 COMMON      0  VARIABLES     12  PROGRAM    188

RELATIVE ENTRY POINT ADDRESS IS 000F (HEX)

END OF COMPILATION
```

```
PAGE   1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0018        0018         0000

V2 M11   ACTUAL 16K  CONFIG 16K

// FOR
*LIST ALL
*ONE WORD INTEGERS
      SUBROUTINE FWT3 (XREAL,N,NU)
C     THIS SUBROUTINE COMPUTES A FAST WALSH TRANSFORM.THE OUTPUT CO-
C     EFFICIENTS ARE PALEY OR DYADIC ORDERED.THIS ALGORITHM IS BASED ON
C     A COOLEY-TUKEY TYPE ALGORITHM.FWT3 IS BASICALLY A STANDARD FAST
C     FOURIER TRANSFORM SUBROUTINE THAT HAS BEEN MODIFIED TO DO A WALSH
C     TRANSFORM.THE MAJOR MODIFICATIONS ARE THE SINE FUNCTIONS ARE SET
C     EQUAL TO ONE AND THE INPUT DATA IS BIT REVERSED.THE FFT SUBROUTINE
C     WAS TAKEN FROM ''THE FAST FOURIER TRANSFORM'' BY E.ORAN BRIGHAM&
C     PRENTICE-HALL&ENGLEWOOD CLIFF.N.J.&1974.PP.160-164.THE MODIFICAT-
C     IONS WERE SUGGESTED BY J.L.SHANKS.''COMPUTATION OF THE FAST WALSH-
C     FOURIER TRANSFORM''&I.E.E.E. TRANS. ON COMPUTERS,VOL.EC-18,PP.457-
C     459,MAY1969.
C
C
C     XREAL-THIS VECTOR CONTAINS THE DATA TO BE TRANSFORMED.
C     N-THE NUMBER OF POINTS TO BE TRANFORMED.
C     NU-THE POWER OF 2 THAT N IS EQUAL I.E.,N=2**NU.
C
C     THIS SUBROUTINE REQUIRES THE INTEGER FUNCTION IBITR TO DO THE BIT
C     REVERSE OPERATIONS.
C     FWT3 COMPUTES THE FORWARD AND INVERSE TRANSFORMS.THE CALL TO THE
C     SUBROUTINE IS IDENTICAL.   THE OUTPUT OR INPUT DATA FOR FWT3 MUST
C     BE SCALED RELATIVE TO THE VALUE OF N AS WITH MOST FFT SUBROUTINES.
C
C     NOTE&THE TRANSFORM COEFFICIENTS ARE RETURNED FROM FWT3 IN THE
C     VECTOR XREAL. THE INPUT DATA IS DESTROYED.
C
C
      DIMENSION XREAL(1)
C     BIT REVERSE INPUT
      DO 103  K=1,N
      I=IBITR(K-1,NU)+1
      IF(I-K) 103,104,104
  104 TREAL=XREAL(K)
      XREAL(K)=XREAL(I)
      XREAL(I)=TREAL
  103 CONTINUE
C     NOW COMPUTE THE TRANSFORM
C
C     INITIALIZE THE PARAMETERS
      N2=N/2
      K=0
      DO 100  L=1,NU
  102 DO 101  I=1,N2
C     COMPUTE THE ARRAY INDEX FOR THE DUAL NODE PAIR
```

```
PAGE    2
                                                                              FWT3 460
      K1=K+1                                                                  FWT3 470
      K1N2=K1+N2                                                              FWT3 480
C     COMPUTE THE DUAL NODE PAIR                                              FWT3 490
      TREAL=XREAL(K1N2)                                                       FWT3 500
      XREAL(K1N2)=XREAL(K1)-TREAL                                             FWT3 510
      XREAL(K1)=XREAL(K1)+ TREAL                                              FWT3 520
  101 K=K+1                                                                   FWT3 530
      K=K+N2                                                                  FWT3 540
      IF(K-N) 102,99,99                                                       FWT3 550
   99 K=0                                                                     FWT3 560
  100 N2=N2/2                                                                 FWT3 570
      RETURN                                                                  FWT3 580
      END
VARIABLE ALLOCATIONS
 TREAL(R )=0000          K(I )=0004          I(I )=0005          N2(I )=0006
  K1N2(I )=0009

STATEMENT ALLOCATIONS
 104  =0038  103  =0055  102  =006D  101  =0098  99  =0086  100  =00BA

FEATURES SUPPORTED
 ONE WORD INTEGERS

CALLED SUBPROGRAMS
 IBITR   FADD   FSUB   FLD   FLDX   FSTO   FSTOX   SUBSC   SUBIN

INTEGER CONSTANTS
     1=000C      2=000D      0=000E

CORE REQUIREMENTS FOR FWT3
 COMMON      0  VARIABLES     12  PROGRAM    192

RELATIVE ENTRY POINT ADDRESS IS 000F (HEX)

END OF COMPILATION
```

```
PAGE   1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0018        0018        0000

V2 M11   ACTUAL 16K  CONFIG 16K

// FOR
*LIST ALL
*ONE WORD INTEGERS
      SUBROUTINE FWT4 (XREAL,N,NU)
C     THIS SUBROUTINE COMPUTES A FAST WALSH TRANSFORM.THE OUTPUT CO-
C     EFFICIENTS ARE SEQUENCY ORDERED.THIS ALGORITHM BASED ON A COOLEY-
C     TUKEY TYPE ALGORITHM.FWT4 IS BASICALLY A STANDARD FAST FOURIER
C     SUBROUTINE THAT HAS BEEN MODIFIED TO DO A WALSH TRANSFORM. THE
C     MAJOR MODIFICATIONS ARE THE SINE FUNCTIONS ARE SET EQUAL TO ONE
C     AND THE INPUT DATA IS BIT REVERSED. THERE IS A SLIGHT MODICATION
C     TO THE WAY THE ''BUTTERFLYS'' ARE COMPUTED. THE FFT SUBROUTINE WAS
C     TAKEN FROM ''THE FAST FOURIER TRANSFORM'' BY E. ORAN BRIGHAM&
C     PRENTICE-HALL,INC.&   1974.PP.160-164.THE MODIFICATIONS WERE
C     SUGGESTED BY J.L.SHANKS,''COMPUTATION OF THE FAST WALSH-FOURIER
C     TRANSFORM''&I.E.E.E. TRANS.ON COMPUTERS,VOL.EC-18,PP.457-459,
C     MAY1969&AND BY  J.W.MANZ,''A SEQUENCY-ORDERED FAST WALSH TRANSFORM
C     ''&I.E.E.E. TRANS. ON AUDIO AND ELECTROACOUSTICS,VOL.AU-20,NO.3,
C     AUG.1972,PP.204-205.
C
C
C     XREAL-THIS VECTOR CONTAINS THE DATA TO BE TRANSFORMED.
C     N-THE NUMBER OF POINTS TO BE TRANFORMED.
C     NU-THE POWER OF 2 THAT N IS EQUAL I.E.,N=2**NU.
C
C     THIS SUBROUTINE REQUIRES THE INTEGER FUNCTION IBITR TO DO THE BIT
C     REVERSE OPERATIONS.
C     FWT4 COMPUTES THE FORWARD AND INVERSE TRANSFORMS,THE CALL TO THE
C     SUBROUTINE IS IDENTICAL.   THE OUTPUT OR INPUT DATA FOR FWT4 MUST
C     BE SCALED RELATIVE TO THE VALUE OF N AS WITH MOST FFT SUBROUTINES.
C
C     NOTE&THE TRANSFORM COEFFICIENTS ARE RETURNED FROM FWT3 IN THE
C     VECTOR XREAL. THE INPUT DATA IS DESTROYED.
C
C
      DIMENSION XREAL(1)
C     BIT REVERSE INPUT
      DO 103  K=1,N
      I=IBITR(K-1,NU)+1
      IF(I-K) 103,104,104
 104  TREAL=XREAL(K)
      XREAL(K)=XREAL(I)
      XREAL(I)=TREAL
  103 CONTINUE
C     NOW COMPUTE THE TRANSFORM
C
C     INITIALIZE THE PARAMETERS
      N2=N/2
      K=0
```

```
PAGE   2
      NPASS=0
      DO 100  L=1,NU
  102 DO 101  I=1,N2
C     COMPUTE THE ARRAY INDEX FOR THE DUAL NODE PAIR
      K1=K+1
      K1N2=K1+N2
C     COMPUTE THE DUAL NODE PAIR
      TREAL=XREAL(K1N2)
      IF(NPASS) 109,110,110
  109 TREAL=-TREAL
  110 XREAL(K1N2)=XREAL(K1)-TREAL
      XREAL(K1)=XREAL(K1)+ TREAL
  101 K=K+1
      K=K+N2
      IF(NPASS) 120,120,121
  120 NPASS=1
      GO TO 130
  121 NPASS=-1
  130 IF(K-N) 102,99,99
   99 K=0
      NPASS=1
  100 N2=N2/2
      RETURN
      END
VARIABLE ALLOCATIONS
 TREAL(R )=0000          K(I )=0004           I(I )=0005          N2(I )=0006
    K1(I )=0009       K1N2(I )=000A

STATEMENT ALLOCATIONS
 104  =0030  103  =0057  102  =0073  109  =0090  110  =0095  101  =00AF  120  =00
 100  =00E1

FEATURES SUPPORTED
 ONE WORD INTEGERS

CALLED SUBPROGRAMS
 IBITR   FADD    FSUB    FLD     FLDX    FSTO    FSTOX   SUBSC   SNR     SUBIN

INTEGER CONSTANTS
     1=000E      2=000F      0=0010

CORE REQUIREMENTS FOR FWT4
 COMMON      0  VARIABLES     14  PROGRAM    230

RELATIVE ENTRY POINT ADDRESS IS 0011 (HEX)

END OF COMPILATION
```

```
PAGE   1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0018        0018         0000

V2 M11   ACTUAL 16K   CONFIG 16K

// FOR
*LIST ALL
*ONE WORD INTEGERS
      FUNCTION IBITR(J,NU)
C     THIS INTEGER FUNCTION RETURNS THE BIT REVERSED VALUE FOR THE IN-
C     TEGER J. THE SUBPROGRAM WAS TAKEN FROM ''THE FAST FOURIRE TRANS-
C     FORM'' BY E.ORAN BRIGHAM&PRENTICE-HALL,INC.&ENGLEWOOD CLIFFS,N.J.
C     1974,PP.160-164.
C
C     J-THE VALUE OF THE INTEGER TO BE BIT REVERSED.
C     NU-THE NUMBER OF BITS TO BE CONSIDERED I.E.,THE NUMBER OF BITS
C        THAT DEFINE THE MAXIUM VALUE OF J.
C
C
      J1=J
      IBITR=0
      DO 200  I=1,NU
      J2=J1/2
      IBITR=IBITR*2+(J1-2*J2)
  200 J1=J2
      RETURN
      END
VARIABLE ALLOCATIONS
 IBITR(I )=0002         J1(I )=0003           I(I )=0004          J2(I )=0005

STATEMENT ALLOCATIONS
 200  =0037

FEATURES SUPPORTED
 ONE WORD INTEGERS

CALLED SUBPROGRAMS
 SUBIN

INTEGER CONSTANTS
     0=0006      1=0007      2=0008

CORE REQUIREMENTS FOR IBITR
 COMMON       0  VARIABLES       6   PROGRAM      66

RELATIVE ENTRY POINT ADDRESS IS 0009 (HEX)

END OF COMPILATION
```

```
PAGE   1

// JOB

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0018        0018        0000

V2 M11   ACTUAL 16K  CONFIG 16K

// FOR
*LIST ALL
*ONE WORD INTEGERS
      SUBROUTINE FFT1 (XREAL,XIMAG,N,NU,IFSET)
C
C     THIS SUBROUTINE COMPUTES A FAST FOURIER TRANSFORM USING THE
C     COOLEY-TUKEY ALGORITHM.THIS ALGORITM IS BASED ON THE INPUT BEING
C     IN STANDARD FORM AND THE OUTPUT BEING IN BIT REVERSED ORDERED,
C     THE ARGUMENT OF THE SINE FUNCTIONS ARE BIT REVERSED. THIS
C     SUBROUTINE WAS TAKEN FROM ''THE FAST FOURIER TRANSFORM'' BY E.
C     ORAN BRIGHAM&PRENTICE-HALL&ENGLEWOOD CLIFFS,N.J.&1974,PP.160-164.
C
C
C     XREAL-THIS VECTOR CONTAINS THE REAL PART OF THE DATA TO BE TRANS-
C           FORMED.
C     XIMAG- THIS VECTOR CONTAINS THE IMAGINARY PART OF THE DATA TO BE
C           TRANSFORM.
C     N-THE NUMBER OF POINTS TO BE TRANSFORMED.
C     NU-THE POWER OF 2 THAT N IS EQUAL I.E.,N=2**NU.
C     IFSET-IF IFSET=1 A FORWARD TRANSFORM WILL BE COMPUTED. IF IFSET=
C           -1 A INVERSE TRANSFORM WILL BE COMPUTED.
C
C     THIS SUBROUTINE REQUIRES THE INTEGER FUNCTION IBITR TO DO THE BIT
C     REVERSE OPERATION.
C
C     NOTE&THE TRANSFORM COEFFICIENTS ARE RETURNED FROM FFT1 IN THE
C     VECTORS XREAL AND XIMAG. THE INPUT DATA IS DESTROYED.
C
C
      DIMENSION XREAL(1),XIMAG(1)
C     INITIALIZE THE PARAMETERS
      N2=N/2
      NU1=NU-1
      K=0
C     COMPUTE THE TRANSFORM
      DO 100  L=1,NU
  102 DO 101  I=1,N2
C     COMPUTE THE VALUE OF THE SINE AGRUMENT USING BIT REVERSE
      P=IBITR(K/2**NU1,NU)
      ARG=6.283185*P/FLOAT(N)
      C=COS(ARG)
      S=SIN(ARG*IFSET)
C
C     COMPUTE THE ARRAY INDEX FOR THE DUAL NODE PAIR
      K1=K+1
      K1N2=K1+N2
C     COMPUTE THE DUAL NODE PAIR
      TREAL=XREAL(K1N2)*C+XIMAG(K1N2)*S
```

```
-PAGE    2
      TIMAG=XIMAG(K1N2)*C-XREAL(K1N2)*S
      XREAL(K1N2)=XREAL(K1)-TREAL
      XIMAG(K1N2)=XIMAG(K1)-TIMAG
      XREAL(K1)=XREAL(K1)+TREAL
      XIMAG(K1)=XIMAG(K1)+TIMAG
C
  101 K=K+1
      K=K+N2
      IF(K-N) 102,99,99
   99 K=0
      NU1=NU1-1
  100 N2=N2/2
C     BIT REVERSE THE OUTPUT
      DO 103  K=1,N
      I=IBITR(K-1,NU)+1
      IF(I-K) 103,104,104
  104 TREAL=XREAL(K)
      TIMAG=XIMAG(K)
      XREAL(K)=XREAL(I)
      XIMAG(K)=XIMAG(I)
      XREAL(I)=TREAL
      XIMAG(I)=TIMAG
  103 CONTINUE
      RETURN
      END
 VARIABLE ALLOCATIONS
      P(R )=0000        ARG(R )=0002          C(R )=0004          S(R )=0006        T
     N2(I )=000E        NU1(I )=000F          K(I )=0010          L(I )=0011
   K1N2(I )=0014

 STATEMENT ALLOCATIONS
  102  =0057  101  =00E1   99   =00FC  100  =0106  104  =012E  103  =015C

 FEATURES SUPPORTED
  ONE WORD INTEGERS

 CALLED SUBPROGRAMS
  IBITR   FCOS    FSIN    FADD    FSUB    FMPY    FLD     FLDX    FSTO    FSTOX   F
  SUBIN

 REAL CONSTANTS
  .628318E 01=0018

 INTEGER CONSTANTS
      2=001A      1=001B      0=001C

 CORE REQUIREMENTS FOR FFT1
  COMMON       0  VARIABLES       24  PROGRAM      336

 RELATIVE ENTRY POINT ADDRESS IS 001D (HEX)

 END OF COMPILATION
```

```
PAGE   1

// JOB

LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         0018         0018         0000

V2 M11    ACTUAL 16K   CONFIG 16K

// FOR
*LIST ALL
*ONE WORD INTEGERS
      SUBROUTINE SORT1(X,NPONT,MT)
C     THIS SUBROUTINE SHUFFLES THE OUTPUT COEFFICIENTS FROM A FAST
C     HADAMARD TRANSFORM PROGRAM SO THAT THEY ARE IN SEQUENCY ORDER.
C     THIS ALGORITM WAS SUGGESTED BY B.K.BHAGAVAN AND R.J.POLGE IN
C     ''SEQUENCING THE HADAMARD TRANSFORM''&I.E.E.E. TRANS. ON AUDIO
C     AND ELECTROACOUSTICS,OCT1973,PP.472-473.
C
C
C     X-THIS VECTOR CONTAINS THE HADAMARD COEFFICIENTS TO BE SEQUENCY
C       ORDERED.
C     NPONT-THE NUMBER OF POINTS TO BE ORDERED.
C     MT-THE POWER OF 2 THAT NPONT IS EQUALI.E.,NPONT=2**MT.
C
C
      DIMENSION X(1)
C     INITIALIZE AND SET UP THE LOOP FOR STEP 1
      IT=MT-1
      KPONT=2**IT
      KPASS=1
      INSET=0
   50 DO 100 I=1,IT
      JT=2**(I-1)
      JTT=IT-I+1
      KT=2**JTT
      DO 100 J=1,JT
      DO 100 K=2,KT,2
      IN1=K+(J-1)*(2**(JTT+1))+INSET
      IN2=IN1-1+KT
      Y=X(IN1)
      X(IN1)=X(IN2)
      X(IN2)=Y
  100 CONTINUE
C
C     STEP 2 OF THE ALGORITHM
C
      DO 200 I=2,KPONT,2
      K=KPONT+I+INSET
      Y=X(K)
      X(K)=X(K-1)
  200 X(K-1)=Y
C
C     STEP 3 OF THE ALGORITHM
C
      IF(KPASS) 300,350,350
  300 INSET=INSET+MPONT
```

```
PAGE   2

      IF(NPONT-INSET) 350,350,50                                             SORT 460
  350 INSET=0                                                                SORT 470
      IT=IT-1                                                                SORT 480
      IF(IT) 400,400,360                                                     SORT 490
  360 KPONT=2**IT                                                            SORT 500
      MPONT=2*KPONT                                                          SORT 510
      KPASS=-1                                                               SORT 520
      GO TO 50                                                               SORT 530
  400 RETURN                                                                 SORT 540
      END                                                                    SORT 550
VARIABLE ALLOCATIONS
     Y(R )=0000         IT(I )=0006       KPONT(I )=0007      KPASS(I )=0008
    JT(I )=0008        JTT(I )=000C          KT(I )=000D          J(I )=000E
   IN2(I )=0011      MPONT(I )=0012

STATEMENT ALLOCATIONS
 50    =003D  100   =009E  200   =00DA  300   =00EE  350   =00FA  360   =0108  400   =01

FEATURES SUPPORTED
 ONE WORD INTEGERS

CALLED SUBPROGRAMS
 FLD      FLDX     FSTO     FSTOX    FIXI     SUBSC    SUBIN

INTEGER CONSTANTS
     1=0016       2=0017       0=0018

CORE REQUIREMENTS FOR SORT1
 COMMON       0  VARIABLES      22  PROGRAM     264

RELATIVE ENTRY POINT ADDRESS IS 0019 (HEX)

END OF COMPILATION
```

```
PAGE   1

// JOB

LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         0018         0018         0000

V2 M11   ACTUAL 16K   CONFIG 16K

// FOR
*LIST ALL
*ONE WORD INTEGERS
      SUBROUTINE WALSH (INUM,NWAL,IERR,NPONT)
C     THIS SUBROUTINE COMPUTES A 'NXN' WALSH MATRIX,WHERE N IS A INTEGRA
C     POWER OF 2.                                        29OCT.1974
C     THIS ALGORITHM WAS SUGGESTED BY B.A.HUTCHINS,''EXPERIMENTAL
C     ELECTRONIC MUSIC DEVICES EMPLOYING WALSH FUNCTIONS''&JOURNAL OF
C     THE AUDIO ENGINEERING SOCIETY&VOL.21,NO.8,OCT.1973,PP.640-645.
C
C
C     INUM-AN INTEGER SUCH THAT NPONT=2**INUM,WHERE NPONT IS THE SIZE
C          OF THE DESIRED WALSH MATRIX.
C     NWAL-AN ARRAY CONTAINING THE WALSH MATRIX OF DIMENSION 'NPONT X
C          NPONT'.
C     IERR-ERROR CODE,IF IERR=0--NO ERROR
C                     IF IERR=1--ERROR MADE IN SETTING UP THE RADEMACHER
C                                FUNCTIONS.
C
C
C
      DIMENSION NWAL(NPONT,NPONT)
      IERR=0
C
C     GENERATE THE WALSH FUNCTION  WAL(0,T)
C
      DO  20 I=1,NPONT
   20 NWAL(1,I)=1
C
C     GENERATE THE RADEMACHER FUNCTIONS
C
      DO 200 K=1,INUM
      INDEX=2**K
      DO 200 L=1,NPONT
      P=(((2.**K)*L)-1.)/FLOAT(NPONT)
C     HIGHEST INTEGER OF P
      N1=P
      N1=N1+1
      P=FLOAT(N1)/2.
      N1=P
      P=P-FLOAT(N1)
      IF( P) 210,175,185
  175 NWAL(INDEX,L)=-1
      GO TO 200
  185 NWAL(INDEX,L)=1
  200 CONTINUE
      GO TO 215
```

```
PAGE   2
  210 IERR=1
      GO TO 400
C
C     GENERATE THE OTHER WALSH FUNCTIONS FROM THE ABOVE RAD. FUNCTIONS.
C
  215 IN=INUM-1
      DO 300 KK=1,IN
      NS=2**(KK+1)-2**KK-1
      INDEX=2**(KK+1)
      DO 300 NA=1,NS
      INN=INDEX-NA
      NAKK=NA+1
      DO 300 L=1,NPONT
      NCEX=NWAL(INDEX,L)*NWAL(NAKK,L)
      IF(NCEX) 240,250,250
  240 NWAL(INN,L)=-1
      GO TO 300
  250 NWAL(INN,L)=1
  300 CONTINUE
  400 RETURN
      END
VARIABLE ALLOCATIONS
     P(R )=0000          I(I )=0006            K(I )=0007        INDEX(I )=0008
    IN(I )=000B         KK(I )=000C           NS(I )=000D           NA(I )=000E
```

REFERENCES

1. Ahmed, N. and Rao, K. R., "Spectral Analysis of Linear Digital Systems Using BIFORE", Electronic Letters, Vol. 6, No. 2, Jan. 1970, pp. 43-44.

2. Ahmed, N. and Rao, K. R., "Discrete Fourier and Hadamard Transforms", Electronic Letters, Vol. 6, No. 7, April 1970, pp. 221-224.

3. Ahmed, N. and Rao, K. R., "Walsh Functions and Hadamard Transforms", Proceedings of Walsh Function Symposium, NTIS, 1972, AD-744-650, pp. 8-13.

4. Ahmed, N., Rao, K. R. and Schultz, R. B., "The Generalized Transform", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, pp. 60-67.

5. Ahmed, N. and Natarajan, T., "On Logical and Arithmetic Autocorrelation Functions", IEEE Trans. on Electromagnetic Compatibility, Vol. EMC-16, No. 3, August 1974, pp. 177-183.

6. Ahmed, N., Rao, K. R. and Schultz, R. B., "A Generalized Discrete Transform, Proceedings of the IEEE, Sept. 1971, pp. 1360-1362.

7. Ahmed, N., Rao, K. R. and Abdussattar, A. L., "BIFORE or Hadamard Transform", IEEE Trans. Audio Electroacoust., Vol. AU-19, No. 3, Sept. 1971, pp. 225-234.

8. Ahmed, N., et al, "On Notation and Definition of Terms Related to a Class of Complete Orthogonal Functions", IEEE Trans. Electromag. Compat., Vol. EML-15, No. 2, May 1973, pp. 75-80.

9. Bell, D. A., "Walsh Functions and Hadamard Matrixes", Electronic Letters, Vol. 2, No. 9, Jan. 1966, pp. 340-341.

10. Berauer, G., "Fast 'In Place' Computation of the Discrete Walsh Transform in Sequency Order", Proceedings of Walsh Function Symposium, NTIS, 1972, AD-744-650, pp. 272-275.

11. Bergland, G. D., "A Guided Tour of the Fast Fourier Transform", IEEE Spectrum, Vol. 6, July 1969, pp. 41-52.

12. Blachman, N. M., "Some Comments Concerning Walsh Functions", IEEE Trans. on Information Theory, Vol. IT-18, No. 3, May 1972, pp. 427-428.

13. Blachman, N. M., "Sinusoids versus Walsh Functions", Proceedings of the IEEE, Vol. 62, No. 3, March 1974, pp. 346-354.

14. Bhagauun, B. K. and Polge, R. J., "Sequencing the Hadamard Transform", IEEE Trans. on Audio and Electroacoustics, Oct. 1973, pp. 472-473.

15. Bobwetter, C., "The Mutual Spectral Representation of Trigonometric Functions and Walsh Functions", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, pp. 43-46.

16. Brigham, E. O., The Fast Fourier Transform, Prentice-Hall, 1974.

17. Bracewell, R., The Fourier Transform and Its Applications, McGraw-Hill, 1965.

18. Carl, J. W. and Kabrisky, M., "Playing the Identification Game with Walsh Functions", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, pp. 203-209.

19. Caspari, K., "Generalized Spectrum Analysis", Proceedings of Walsh Function Symposium, NTIS, 1970, AD-707-431, pp. 195-207.

20. Chen, W. H. and Prutt, W. K., "Color Image Coding with the Slant Transform", Proceedings of Walsh Function Symposium, NTIS, 1973, AD-763-000, pp. 155-161.

21. Cheng, D. K. and Liv, J. I., "Walsh-Transform Analysis of Discrete Dyadic-Invariant Systems", IEEE Trans. on Electromagnetic Compatibility, May 1974, pp. 136-139.

22. Cheng, D. K., Analysis of Linear Systems, Addison-Wesley, 1959.

23. Cheng, D. K. and Liv, J. J., "Walsh Transform Analysis of Discrete Linear Systems", Proceedings of Walsh Function Symposium, NTIS, 1973, AD-763-000, pp. 61-65.

24. Cheng, D. K. and Liv, J. J., "Time-Domain Analysis of Dyadic-Invariant Systems", Proceedings of the IEEE, July 1974, pp 1038-1040.

25. Crittenden, R. B., "Walsh-Fourier Transforms", Proceedings of Walsh Function Symposium, NTIS, 1970, AD-707-431, pp. 170-174.

26. Cooley, J. W., Lewis, P. A. W. and Welch, P. D. "The Finite Fourier Transform", IEEE Trans. on Audio Electroacoust., Vol. AU-17, June 1969, pp. 77-85.

27. Cooley, J. W., Lewis, P. A. W. and Welch, P. D., "Applications of the Fast Fourier Transform to Integrals, Fourier Series and Convolution Integrals", IEEE Trans. on Audio Electroacoust., Vol. AU-15, June 1967, pp. 79-84.

28. Cooley, J. W. and Tukey, J. W., "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computation, Vol. 19, No. 90, 1965, pp. 297-301.

29. Cooley, J. W., Lewis, P. A. W. and Welch, P. D., "The Fast Fourier Transform Algorithm: Programming Considerations in the Calculation of Sine, Cosine, and Laplace Transforms", Journal of Sound and Vibration, Vol. 12, July 1970, pp. 315-377.

30. Davis, H. F., Fourier Series and Orthogonal Functions, Allyn and Bacon, 1963.

31. Dinh, C., et al, "On Walsh Describing Functions", Proceedings of Walsh Function Symposium, NTIS, AD-744-650, 1972, pp. 362-368.

32. Ditkin, V. A. and Prudnikov, A. P., Integral Transforms and Operational Calculus, Pergamon Press, 1965.

33. Dunklee, A. L., "The Use of Walsh Transforms in Image Processing", Proceedings of Walsh Function Symposium, NTIS, 1973, AD-763-000, pp. 162-167.

34. Fine, N. J., "The Generalized Walsh Functions", Transactions of American Mathematical Society, Vol. 65, 1949, pp. 66-77.

35. Frank, T. H., "Implementation of Dyadic Correlation", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, pp. 111-117.

36. Gaubuts, D. A., and Kitai, R., "A Programmable Walsh Function Generator for Orthogonal Sequency Pairs", IEEE Trans. Electromag. Compat., May 1974, pp. 134-136.

37. Gethoffer, H., "Sequency Analysis Using Correlation and Convolution", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, pp. 118-123.

38. Glassman, J. A., "A Generalization of the Fast Fourier Transform", IEEE Trans. on Computers, Vol. C-19, No. 2, Feb. 1970, pp. 105-116.

39. Glisson, T. H., Black, C. I. and Sage, A. P., "The Digital Computation of Discrete Spectra Using the Fast Fourier Transform", IEEE Trans. on Audio and Electroacoustics, Vol. AU-18, No. 3, Sept. 1970, pp. 271-287.

40. Golomb, S. W. and Baumert, L. D., "The Search for Hadamard Matrices", The American Mathematical Monthly, Vol. 70, No. 1, Jan. 1963, pp. 12-17.

41. Hadamard, J., "Resolution d'une Question Relative aux Determinants", Bulletin des Science Mathematiques, Series 2, Vol. 17, 1893, Part 1, pp. 240-246.

42. Harmuth, H. F., "All You Always Wanted to Know About Electromagnetic Walsh Waves", Proceedings of Walsh Function Symposium, NTIS, AD-744-650, 1972, pp. 23-29.

43. Harmuth, H. F., Transmission of Information by Orthogonal Functions, 2nd Ed., Springer-Verlag, 1972.

44. Harmuth, H. F., "A Generalized Concept of Frequency and Some Applications", IEEE Trans. Info. Theory, Vol. IT-14, No. 3, May 1968, pp. 375-382.

45. Harmuth, H. F., "Applications of Walsh Functions in Communications", IEEE Spectrum, May 1969, pp. 82-91.

46. Harmuth, H. F., "Survey of Research and Development in the Field of Walsh Functions and Sequency Theory", Proceedings of Walsh Function Symposium, NTIS, 1973, AD-763-000, pp. 1-9.

47. Henderson, K. W., "Some Notes on the Walsh Functions", IEEE Trans. on Electronic Computers, Feb. 1964, pp. 50-52.

48. Hutchins, B. A., "Experimental Electronic Music Devices Employing Walsh Functions", Journal of the Audio Engineering Society, Vol. 21, No. 8, 1973, pp. 640-645.

49. Insam, E., "Walsh Functions in Waveform Synthesizers", Journal of the Audio Engineering Society, Vol. 22, No. 6, 1974, pp. 422-425.

50. Johnson, D. L., "Walsh Function Analysis of Dyadic Invariant Linear Systems", M. S. thesis, Syracuse University, Syracuse, New York, 1971.

51. Kak, S. C., "Sampling Theorem in Walsh-Fourier Analysis", Electronic Letters, Vol. 6, No. 14, July 1970, pp. 447-448.

52. Kernett, B. L. N., "A Note on the Finite Walsh Transform", IEEE Trans. on Info. Theory, July 1970, pp. 489-491.

53. Kreyszig, E., Advanced Engineering Mathematics, John Wiley, 1967.

54. Kuo, B. C., Analysis and Synthesis of Sampled-Data Control Systems, Prentice-Hall, 1963.

55. Lackey, R. G., and Meltzer, D., "A Simplified Definition of Walsh Functions", IEEE Trans. on Computers, Feb. 1971, pp. 211-213.

56. Lackey, R. B., "The Wonderful World of Walsh Functions", Proceedings of Walsh Function Symposium, NTIS, 1972, AD-744-650, pp. 2-7.

57. Lebert, F. J., "Walsh Function Generator for a Million Different Functions", Proceedings of Walsh Function Symposium, NTIS, AD-707-431, 1970, pp. 52-55.

58. Lee, J. D., "Review of Recent Work on Applications of Walsh Functions in Communications", Proceedings of Walsh Function Symposium, NTIS, 1970, AD-707-431, pp. 26-35.

59. Lee, T., "Hardware Approach to Walsh Function Sequency Filters", Proceedings of Walsh Function Symposium, NTIS, AD-707-431, 1970, pp. 7-11

60. Lopez de Zavalia, R. J., ët al, "Walsh Function Generator for Laboratory Use", Proceedings of Walsh Function Symposium, NTIS, AD-744-650, 1972, pp. 114-122.

61. Manz, J. W., "A Sequency-Ordered Fast Walsh Transform", IEEE Audio Electroacoust., Vol. AU-20, No. 3, August 1972, pp. 204-205.

62. Masqusi, M., "Walsh Functions and the Sampling Principle", Proceedings of Walsh Function Symposium, NTIS, 1972, AD-744-650, pp. 261-264.

63. Morris, R. L. and Miller, J. R., Designing with TTL Integrated Circuits, McGraw-Hill, 1971.

64. Murray, G. G., "Digital Walsh Filter Design", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, pp. 101-105.

65. Ohnsorg, F. R., "Spectral Modes of the Walsh-Hadamard Transform", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, p. 55-59.

66. Paley, R. E. A. C., "A Remarkable Series of Orthogonal Functions", Proceedings of the London Mathematical Society, Series 2, Vol. 34, Part 1, 1932, pp. 241-279.

67. Peterson, H. L., "Generation of Walsh Functions, Proceedings of Walsh Function Symposium, NTIS, 1970, AD-707-431, pp. 55-58.

68. Pichler, F., Das System der Sal- und Cal- Funktionen als Erweiterung des Systems der Walsh-Funktionen und die Theorie der Sal- und Cal- Fourier Transformation", Thesis, Dept. of Mathematics, Innsbruck University, Austria, 1967.

69. Pichler, F., "Walsh Functions and Linear System Theory", Proceedings of Walsh Function Symposium, NTIS, 1970, AD-707-431, pp. 175-182.

70. Pichler, F., "Walsh Functions--Introduction to the Theory", Proceedings of the NATO Advanced Study Institute on Signal Processing, Academic Press, 1973, pp. 23-41.

71. Pichler, F., "On State Space Description of Linear Dyadic Invariant Systems", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, pp. 166-170.

72. Pitassi, D. A., "Fast Convolution Using Walsh Transforms", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, pp. 130-133.

73. Powers, D. L., Boundary Value Problems, Academic Press, 1972.

74. Pratt, W. K., "Linear and Nonlinear Filtering in the Walsh Domain", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, pp. 38-42.

75. Rademacher, H., "Einige Satze vol Allgemeinen Orthogonalfuktionen", Math. Ann., Vol. 87, 1922, pp. 122-138.

76. Redinbo, G. R., "Transforms of Generalized Walsh Functions", Proceedings of the IEEE, Sept. 1971, pp. 1352-1353.

77. Robinson, G. S., "Logical Convolution and Discrete Walsh and Fourier Power Spectral", IEEE Trans. on Audio and Electroacoust.", Vol. AU-20, No. 4, Oct. 1972, pp. 271-280.

78. Robinson, G. S., "Fourier Transforms of Walsh Functions", IEEE Trans. on Electromagnetic Compatibility, Vol. EMC-16, No. 3, Aug. 1974, pp. 183-185.

79. Rosenbloom, J. H., "Physical Interpretation of the Dyadic Group", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, pp. 158-165.

80. Roth, D., "Special Filters Based on Walsh Functions", Proceedings of Walsh Function Symposium, NTIS, 1970, AD-707-431, pp. 12-16.

81. Sandy, G. F., "Some Walsh-Fourier Analysis Techniques", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, pp. 151-154.

82. Schreiber, H. H., "Bandwidth Requirements for Walsh Functions", Proceedings of Walsh Function Symposium, NTIS, 1970, AD-707-431, pp. 46-51.

83. Shanks, J. L., "Computation of the Fast Walsh-Fourier Transform", IEEE Trans. on Computers, May 1969, pp. 457-459.

84. Shum, F. Y. Y., Elliott A. R. and Brown, W. O., "Speech Processing with Walsh-Hadamard Transforms", IEEE Trans. on Audio and Electroacoust., Vol. AU-21, No. 3, June 1973, pp. 174-179.

85. Shum, Y. Y. and Elliott, A. R., "Computation of the Fast Hadamard Transform", Proceedings of Walsh Function Symposium, NTIS, 1972, AD-744-650, pp. 177-181.

86. Siemens, K. H. and Kitai, R., "Walsh Series to Fourier Series Conversion", Proceedings of Walsh Function Symposium, NTIS, 1972, AD-744-650, pp. 295-297.

87. Siemens, K. H. and Kitai, R., "A Nonrecursive Equation for the Fourier Transform of a Walsh Function", IEEE Trans. on Electromagnetic Compatibility, Vol. EMC-15, No. 2, May 1973, pp. 81-83.

88. Siemens, K. H., Digital Walsh-Fourier Analyser for Periodic Waveforms, M. E. thesis, McMaster University, Hamilton, Ontario, Canada, 1969.

89. Theilheimer, F., "A Matrix Version of the Fast Fourier Transform", IEEE Trans. on Audio and Electroacoust., Vol. AU-17, No. 2, June 1969, pp. 158-161.

90. Walsh, J. L., "A Closed Set of Orthogonal Functions", American Journal of Mathematics, Vol. 45, 1923, pp. 5-24.

91. Welch, L. R., "Walsh Functions and Hadamard Matrices", Proceedings of Walsh Function Symposium, NTIS, 1970, AD-707-431, pp. 163-165.

92. Wylie, C. R., Advanced Engineering Mathematics, McGraw-Hill, 1966.

93. Yuen, C. K., "Walsh Functions and Gray Code", Proceedings of Walsh Function Symposium, NTIS, 1971, AD-727-000, pp. 68-73.

94. Yuen, C. K., "Remarks on the Ordering of Walsh Functions", IEEE Trans. on Computers, Dec. 1972, p. 1452.