TRANSFORM DOMAIN ADAPTIVE FILTERING USING WALSH TRANSFORMS

by

Derviş ÜSKÜDAR

B.S. in E.E., Boğaziçi University, 1982

Submitted to the Institude for Graduate Studies in Science and Engineering in partial fulfilment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Boğaziçi University

1986

TRANSFORM DOMAIN ADAPTIVE FILTERING USING

WALSH TRANSFORMS

APPROVED BY

Doç. Dr. Avni MORGÜL
(Thesis Supervisor) .....

Doç. Dr. Ömer CERİD .....

Doç. Dr. Bülent SANKUR .....

Dr. Emin ANARIN .....

DATE OF APPROVAL : March 24, 1986

To Drs.

Avni Morgül, Ömer Cerid, and Semih Bingöl whom I had the good fortune to work with and learn from

ABSTRACT

The purpose of this study is to investigate the behaviour of the transform domain adaptive filter in the Walsh domain. As it is well known, because of the binary nature of the basis functions, the computationally intensive frequency domain adaptive filter algorithm lends itself to easier implementation if Walsh transform is employed.

This thesis starts with a brief review of Walsh functions, Walsh transforms and adaptive filtering. The Walsh domain adaptive filter has been simulated on the digital computer. Simulation results for various input signals embedded in noise are given. It is shown that better convergence can be achieved when the adaptive filter is operated in the Walsh domain, rather than time domain.

The effect of initial weight on convergence has been investigated. It has been observed that small values of the initial weight are necessary for better convergence performance. Although the theoretical basis of the effect of small initial weight on convergence characteristics has not been established yet, simulation results indicate improved performance with a small initial weight.

Since the Walsh transform requires only additions and subtractions, the digital implementation of Walsh domain adaptive filter is easily achievable, and can be thought as the scope of a further study.

ÖZETCE

İletişim sistemlerinde haber işaretine gürültü karışması kaçınılmaz bir olaydır. İletişim sonucunda elde edilen işaretten gürültünün ayırd edilmesi oldukça zor bir iş olarak karşımıza çıkar. Haber işaretinden gürültüyü ayırma işlemi dikkatlice yapılmazsa, sonuç olarak daha da bozulmuş bir işaretle karşılaşabiliriz. Buna rağmen, eğer haberin gürültüden arındırılması işlemi uyarlanır (adaptive) bir sistem vasıtasıyla yapılırsa gürültünün haber işareti üzerindeki etkisi azaltılabilir.

Uyarlanır bir sistemde, süzme işlemi gelen işarete bağlı olarak parametreleri kendi kendine ayarlanan süzgeç devreleri ile yapılır. İşaretteki düzelmenin daha kısa bir sürede sağlanabilmesi için uyarlanır süzme işlemi zaman uzayı yerine frekans uzayında yapılabilir.

Walsh dönüşümü katsayıları yalnızca gerçel sayılardan oluşmaktadır. Fourier dönüşümü yada diğer başka dönüşümlerle karşılaştırıldığında Walsh dönüşümünün daha az hesaplama işlemi gerektirdiği görülmüştür. Bu çalışmada Walsh uzayı uyarlanır süzgeçlerin benzetimi yapılmış ve benzetim sonuçları incelenmiştir.

Walsh fonksiyonları belirli bir kurala göre dizilerek birbirini takip eden ve sadece +1 ve -1 değerlerini alabilen işaretlerden meydana gelmektedirler. Bir Walsh fonksiyonunun tanımlanabilmesi için indeks numarası ve zaman aralığı olmak üzere iki değişkene ihtiyaç vardır. Walsh fonksiyonlları **Wal**(n,t) şeklinde yazılmak suretiyle ifade

edilmekte olup n indeks numarasını, t de zaman aralığını temsil etmektedir. Walsh fonksiyonlarının ilk tanımlanmasında sıralanış numaralarını belirleyen etken olarak işaret değiştirme sayısı göz önüne alınmıştır. Daha sonraları muhtelif özellikleri göz önüne alınarak Paley ve Hadamard sıralamaları yapılmıştır.

Hadamard sıralamasındaki Walsh fonksiyonlarının örneklenmiş hali Hadamard matrisleriyle de tanımlanmaktadır. Bu sebeple Walsh dönüşümüne aynı zamanda Hadamard dönüşümü de denilmektedir. Bir fonksiyonun Walsh dönüşümü alınırken ikinin tam katı herhangi bir sayıda örneklenmiş halinin Hadamard matrisiyle çarpımı örnek miktarına bölünür. Ters dönüşüm alınırkende aynı işlem yapılır ancak örnekleme miktarına bölme işlemi yapılmaz. Hadamard matrisinin elemanları sadece +1 ve -1 lerden oluştuklarından gerçekte çarpım yerine toplam işlemi yapılmaktadır. Bu özelliği itbarıyla işlem kolaylığı sağlaması bir yana sayısal olarak gerçekleştirilmeye de uygundur.

Uyarlanır süzme işleminde uyarlama algoritması olarak en küçük kareler (least-mean square) algoritması seçilmiştir. Bu işlemde rastgele bir başlangıç ağırlığı seçilir ve giriş işaretinin bu ağırlıkla çarpımı çıkış işareti olarak tanımlanır. Çıkış işaretinin istenen işarete olan uygunluğu araştırılır ve o andaki uygunsuzluğu giderecek yönde bir düzeltme ağırlık üzerinde yapılır. Değeri yeniden belirlenmiş olan ağırlık bir sonraki işlemde giriş işaretiyle çarpılır ve en uygun çözümü bulmaya çalışarak bu işlemler devam eder gider.

Benzetim başlangıcında programlama dili olarak FORTRAN seçilmiş

ancak grafik neticeler elde edebilmek amacıyla daha sonra BASIC dilinde program yeniden düzenlenmiştir. Çeşitli işaretler için benzetim sonuçları gözlenmiş ve bunlardan bir kısmı sonuçlar kısmında sunulmuştur.

Beyaz gürültü (white noise) tarafından bozulmuş olan bir sinüs işaretinin iyileştirilmesi uyarlama adım genişliği 0.005 alındığında ortalama 60 adım sonucunda başarılmış ve gürültünün genliğinde yaklaşık 30 dB lik bir azalma sağlanmıştır. Karesel bir işaretin üzerindeki gürültünün temizlenmesinde ise daha iyi sonuçlar elde edilmiştir. Uyarlama adım genişliği 0.0025 olarak seçildiğinde yaklaşık 40 adım sonunda işaretin iyileşmesi sağlanmış ve genelde 40 dB nin üzerinde bir gürültü bastırımı sağlanmıştır. Uyarlamanın sağlanmasından sonra hata genliğindeki değişmenin 8 dB den daha az sınırlar içinde olduğu gözlenmiştir.

Başlangıç ağırlığının yakınsama üzerindeki etkisi araştırılmış, küçük başlangıç ağırlığı seçildiğinde daha iyi yakınsama etkinliği elde edildiği görülmüştür.

TABLE OF CONTENTS

Page

ABSTRACT iv

ÖZETCE v

TABLE OF CONTENTS viii

LIST OF FIGURES x

LIST OF SYMBOLS xii

I. INTRODUCTION 1

II. WALSH FUNCTIONS AND WALSH TRANSFORMS 6

2.1 INTRODUCTION TO WALSH FUNCTIONS 6

2.2 RADEMACHER FUNCTIONS 8

2.3 WALSH FUNCTIONS 9

2.4 FUNCTION ORDERING 13

2.4.1 WALSH ORDERING 13

2.4.2 PALEY ORDERING 15

2.4.3 HADAMARD ORDERING 16

2.5 BIT REVERSAL OPERATION 18

2.6 GENERATION OF WALSH FUNCTIONS 20

2.6.1 WALSH FUNCTION GENERATION FROM RADEMACHER FUNCTIONS ... 22

2.6.2 WALSH FUNCTION GENERATION FROM HADAMARD MATRICES 25

2.7 WALSH TRANSFORM 27

2.7.1 MATRIX DEFINITION OF WALSH TRANSFORM 29

2.7.2 FAST WALSH TRANSFORM 32

III. ADAPTIVE FILTERS 39

3.1 ADAPTIVE SIGNAL PROCESSING 39

3.2 ADAPTIVE FILTERING 40

3.3 ADAPTIVE FILTER OPERATION 43

Page

3.4 THE LMS ADAPTATION ALGORITHM 46

3.5 TRANSFORM DOMAIN LMS ALGORITH 48

IV. SIMULATION OF WALSH DOMAIN ADAPTIVE FILTER 51

V. RESULTS AND CONCLUSIONS 54

5.1 ADAPTIVE FILTERING OF A SINE WAVE 54

5.2 ADAPTIVE FILTERING OF A RECTANGULAR WAVE 57

5.3 ADAPTIVE FILTERING OF A SQUARE WAVE 60

5.4 EFFECTS OF INITIAL WEIGHT AND ADAPTIVE STEP SIZE ON CONVERGENCE CHARACTERISTICS 63

5.5 CONCLUSIONS 65

5.6 SUGGESTIONS FOR FURTHER STUDY 67

APPENDIX 68

REFERENCES 83

LIST OF FIGURES

		Page
FIGURE 2.1	A set of Rademacher functions	8
FIGURE 2.2	A set of Walsh functions	11
FIGURE 2.3	A set of **Cal** and **Sal** functions	14
FIGURE 2.4	A set of Walsh functions in Paley order	17
FIGURE 2.5	A set of Walsh functions in Hadamard order	19
FIGURE 2.6	Flow diagram for bit reversal operation	21
FIGURE 2.7	Derivation of **Wal**(6,t) from the Rademacher functions	24
FIGURE 2.8	A scheme for generating **Wal**(n,t) from Binary equivalent of its index number, n	24
FIGURE 2.9	Flow diagram for hadamard matrix evaluation	28
FIGURE 2.10	Flow diagram for Walsh transform	31
FIGURE 2.11	Fast Walsh transform signal flow graph	35
FIGURE 2.12	Flow diagram for fast Walsh transform	38
FIGURE 3.1	Open-loop processing system	39
FIGURE 3.2	Open-loop adaptive processing system	39
FIGURE 3.3	Closed-loop adaptive processing system	40
FIGURE 3.4	Adaptive noise cancelling system	41
FIGURE 3.5	Adaptive linear combinatorial system	43
FIGURE 3.6	Transform domain adaptive filter	48
FIGURE 4.1	Flow diagram for Walsh domain adaptive filter	53
FIGURE 5.1	Transform domain adaptive filtering of a sine wave	55
FIGURE 5.2	Walsh spectrum of the waveform shown in Fig.5.1(a)	56

		Page
FIGURE 5.3	Transform domain adaptive filtering of a rectangular wave	58
FIGURE 5.4	Walsh spectrum of the waveform shown in Fig.5.3(a)	59
FIGURE 5.5	Transform domain adaptive filtering of a square wave	61
FIGURE 5.6	Walsh spectrum of the waveform shown in Fig.5.5(a)	62
FIGURE 5.7	Convergence characteristics of the mean-square error as the initial weight varies from 0 to 0.5	64
FIGURE 5.8	Convergence characteristics of the mean-square error as the adaptive step size varies from 0.001 to 0.005	64

LIST OF SYMBOLS

a	Walsh coefficient
b	binary bit
C	constant
d	(1) desired response
	(2) desired response vector
$\hat{d}$	estimate of d
D	transformed desired response vector
e	error signal
$\hat{e}$	estimate of e
E	transformed error vector
E[ ]	expectation operator
f	analog signal
F	transformed signal
g	Gray code bit
H	Hadamard matrix
H	transformed weight vector
i	general index
j	general index
k	general index
n	(1) general index
	(2) noise signal
Pal	Paley function
R	Rademacher function
s	analog signal

t	(1) continuous time
	(2) time index
w	weight value
W	weight vector
$\mathbf{Wal}$	Walsh function
x	input signal vector
X	transformed input signal vector
y	(1) output signal
	(2) output signal vector
Y	transformed output signal vector
λ	eigenvalue
μ	adaptive step size
ϕ	correlation function
∇	gradient operator
$\hat{\nabla}$	estimate of ∇

I. INTRODUCTION

The problem of detecting a signal and its paramaters finds applications in signal processing systems such as radar, sonar, biomedical, digital commununications, etc. In many cases, the presence of noise which is inevitable brings out serious problems. Subtracting noise from a received signal would seem to be a dangerous procedure. It could result in an increase in output noise power if it is done improperly. However, when subtraction of noise from a received signal is controlled by an adaptive algorithm which does not require statistical information about the received signal, noise reduction can be performed with little risk of distorting the signal or increasing the output noise level.

In an adaptive system, enhancement of the output signal can be performed after many operations. In order to obtain rapid convergence, adaptive filtering can be done in a transformed domain rather then time domain. The Walsh transform of a signal consists of only real numbers. This brings less computational requirements in comparison with other transform techniques, such as Fourier transform.

The main objective of this study is to set up a computer model of Walsh domain adaptive filter and investigate the performance of it for the signal enhancement problem.

Walsh functions are complete and orthogonal functions and they are characterised by assuming only two states +1 and -1, thus matching the

behaviour of digital logic. When the sine functions are used to analyze time variable circuits, they often lead to unnecessary complications. It was found that Walsh functions were excellently suited for the analysis of sampled signals and the design of equipment for such signals.

In chapter II we mainly deal with the Walsh functions and Walsh transforms. It has been shown that the Walsh functions were originally suggested, in a different manner, by Hadamard at first. Although the Walsh functions are ordered according to the number of zero crossings for the first time, they can be ordered in some different ways. The orderings and the relationship between them are investigated. When the Walsh functions are ordered in Hadamard order, Hadamard matrices represent the sampled version of the Walsh functions.

We can obtain the Walsh transform of a function by multipliying the sampled version of the function with the Hadamard matrix of appropriate order followed a division of sample number. The sample number should be taken as an integer power of two.

The basic concept of adaptive filtering and transform domain adaptive filtering are introduced in chapter III. The subject of adaptive processors has been a research topic since the 1960s and these have subsequently been applied in many practical systems mainly as adaptive filters. Conventional signal processing systems for the extraction of noise from an incoming signal such as a matched filter operate in an open loop fashion. That is, the same processing function is carried out in the present time interval regardless of whether that

function produced the correct result in the preceeding time interval. In other words, conventional signal processing techniques make the basic assumption that the signal degradation is a known and time invariant quantity. Adaptive processors, on the other hand, operate with a closed loop arrangement. The incoming signal is filtered or weighted in a programmable filter to yield an output which is then compared against a desired, conditioning or training, signal to yield an error signal. This error is then used to update the processor weighting paramaters using an algorithm such that the error is progressively minimized, i.e., the processor output more closely approximates to the training signal.

Adaptive filters are concerned with the use of a programmable filter whose frequency response or transfer function is altered, or adapted, to pass without degradation the desired components of the signal and to attenuate the undesired or interfering signals, or to reduce any distortion on the input signal. In an adaptive system an absolute minimum of a priori information is necessary about the incoming signal. The adaptive filter operates by estimating the statistics of the incoming signal and adjusting its own response in such a way as to minimize some cost function. This cost function may be derived in a number of ways depending on the intended application, but normally it is derived by the use of a secondary signal source or conditioning input. This secondary signal input may be defined as the desired output of the filter. In this case, the task of the adaptive algorithm is to adjust the weights in the programmable filter device in such a way as to minimize the difference between the filter output and the secondary input.

The common realisation of an adaptive filter is a transversal filter where the weights are updated iteratively using an adaptive algorithm. In the transformed domain application, the adaptation is done in a transformed domain rather than in the time domain. In the transformed domain adaptation, the incoming data is processed in blocks and adaptation is done once for each block. By this way, the input data block is converted from serial into parallel form. Thus, the transform domain filter can be thought of as 1 tap transversal filters in parallel.

The computer simulation of the Walsh domain adaptive filter is introduced in chapter IV. Programming language was FORTRAN at the beginning of the study. In order to evaluate graphical results it was rewritten in BASIC. The computer programs are applicable with RADIO SHACK TRS-80 MODEL 16 digital computer.

The results are investigated through an extensive simulation study with many types of input signals. The convergence behaviour of the Walsh domain adaptive filter studied with sine and square waves are given in chapter V. The enhancement of a sine wave corrupted by white noise is performed after about 60 iterations with an adaptive step size of 0.005. The reduction in the noise amplitude is generally greater than 30 dB. Better convergence performances are obtained for the square wave application. The enhancement of a square wave corruted by white noise is performed after about 40 iterations with an adaptive step size of 0.0025. After the adaptation is completed the variation in the amplitude of the mean-square error is less than 8 dB. The value of the

adaptive step size is changed automatically according to the power of the input signal for the mentioned applications. It has been observed that small values of the initial weight are necessary for better convergence performance.

II. WALSH FUNCTIONS AND WALSH TRANSFORMS

2.1 INTRODUCTION TO WALSH FUNCTIONS

The basis for the development of electrical engineering in many areas is a system of sine and cosine functions. Whenever the term frequency is used, reference is made implicitly to these functions. This is due to the desirable properties of frequency domain representation of a large class of functions encountered in the theoretical and practical aspects of engineering design.

In recent years more general classes of complete systems of orthogonal functions have been used for theoretical investigations as well as equipment design. Furthermore, semiconductor devices have made it possible to use linear time variable circuits instead of linear time-invariant ones. While sine and cosine functions have indisputable advantages for linear time-invariant circuits, they often lead to unnecessary complications if they are used to analyze time-variable circuits. With the application of digital techniques and semiconductor technology to the area of electrical systems, the Walsh functions have come into use since they are characterised by assuming only two states, thus matching the behaviour of digital logic.

Historically, the Walsh functions were defined in 1923 by the American mathematician J. L. Walsh. These functions formed a complete orthonormal set taking only two values +1 and -1. Almost at the same time (in 1922), but independent of Walsh, the German mathematician, H. Rademacher, presented another set of two level orthogonal functions,

which were found later to form an incomplete but true subset to the Walsh functions.

In his original paper [1], Walsh gave a recursive definition of the Walsh funtions that orders the functions according to the average number of zero crossings in the time interval. In 1931, an entirely different definition of the Walsh functions was described by R.E.A.C.Paley [2,3]. His definition is based on finite products of Rademacher functions and the order obtained was quite different from that of the Walsh.

A much earlier approach to the Walsh function's definition is through the application of certain orthogonal matrices, containing only the entries +1 and -1, and known as Hadamard matrices [4]. The Walsh functions obtained through Hadamard matrices represent another order which will be explained later.

A new theory is not generally accepted unless its advantages can be demonstrated convincingly. In engineering, a convincing demonstration means a working equipment. Walsh functions were virtually completed by the 1930's. However, publications referred to engineering and other applications did not appear until semiconductors and the digital computers had come into use. The theory of Walsh functions related to practical engineering problems is still being developed. The recent areas of application are; electromagnetic radiation, radar systems, multiplexing, data processing, voice communications, pattern recognition, random access communications, TV picture processing, and seismic event detection [5,6,7,8].

2.2 RADEMACHER FUNCTIONS

Rademacher functions are an incomplete set of orthogonal functions. They were developed in 1922 by the German mathematician H. Rademacher.

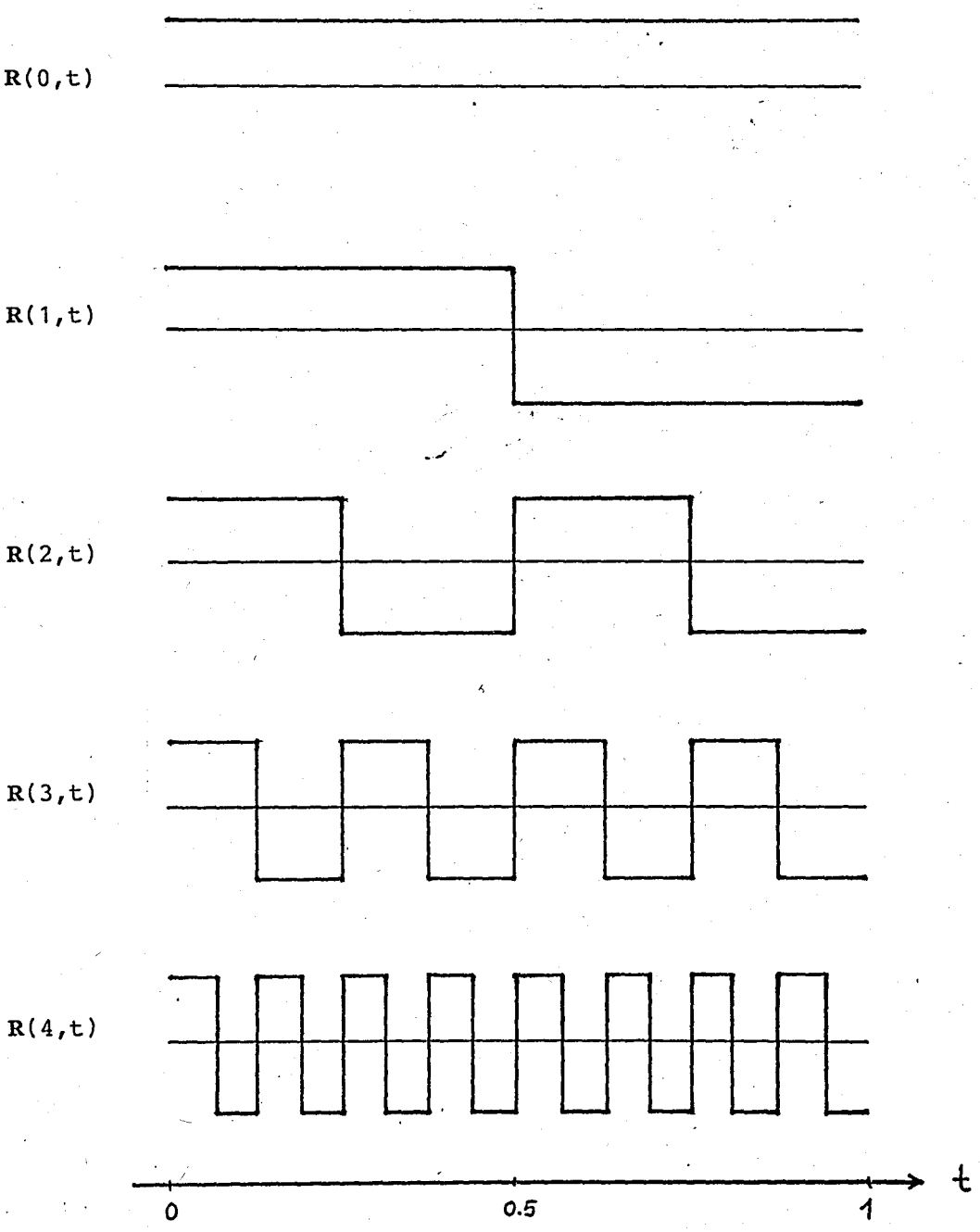


Figure 2.1 A set of Rademacher functions

They represent a series of rectangular pulses or square waves having unit mark-space ratio. The first five of them are shown in Fig.2.1. The Rademacher function of index n is denoted by R(n,t). Thus, they have two arguments n and t such that R(n,t) has 2^{n-1} periods of square-wave over a normalised time base $0 \leqslant t \leqslant 1$. The first function, R(0,t), is equal to 1 for the entire interval. The next and subsequent functions are square waves and their amplitudes are limited between +1 and -1 [8].

They can be derived from sinusoidal functions, as

$$R(n,t) = \text{sign}(\sin(2^n \pi t)) \qquad (2.1)$$

The generation of Rademacher functions may be obtained from a sinusoidal waveform of appropriate frequency by amplification followed by hard limiting. They are important principally since the Walsh functions can be derived from them.

2.3 WALSH FUNCTIONS

The term frequency is defined as the parameter f that distinguishes the functions $\cos 2\pi ft$ and $\sin 2\pi ft$. Generally, practical interpretation of f is the number of cycles per second. The general definition of frequency can be given as one-half the average number of zero-crossing per second. Harmuth introduced the term sequency to describe the generalised frequency. He applied the term sequency to describe functions whose zero-crossings are not uniformly spaced over an

interval, and which are not necessarily periodic [9]. The definition of sequency coincides with the definition of frequency when it is applied to sinusoidal functions. Then, the sequency of a periodic function equals one-half the number of sign changes per period and the sequency of an aperiodic function equals one half the number of sign changes per unit time.

The incomplete set of Rademacher functions was completed by Walsh in 1923, to form the complete set of rectangular functions known as Walsh functions. Walsh functions form an ordered set of rectangular waveforms taking only two amplitude values +1 and -1. Unlike Rademacher functions, the Walsh rectangular waveforms do not have unit mark-space ratio. They are defined over a limited time, T, known as the time base. Like sine-cosine functions, two arguments are required for complete definition [10]. These are, a time period, t, (usually normalised to the time base as t / T) and an ordering number, n, related to sequency. The function is written as

$$\mathbf{Wal}(n,t) \tag{2.2}$$

The Walsh functions can be defined by a difference equation rather than a differential equation. In terms of Rademacher functions they can be defined as

$$\mathbf{Wal}(n,t) = \prod_{i=1}^{m+1} g_{i-1}\mathbf{R}(i,t) \tag{2.3}$$

where n is expressed as a binary number

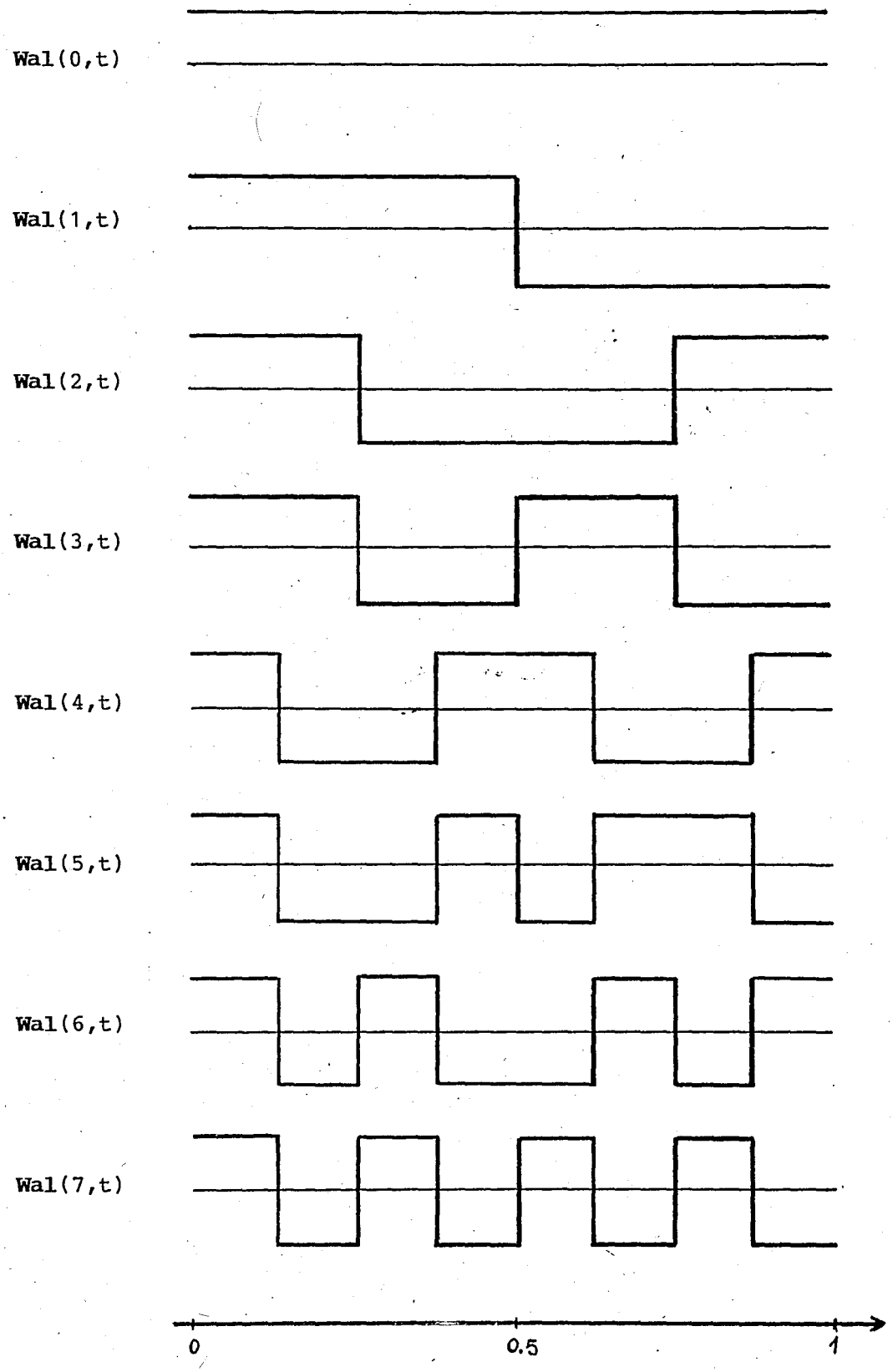


FIGURE 2.2 A set of Walsh functions

$$n = b_m 2^m + b_{m-1} 2^{m-1} + ... + b_0 2^0 \qquad (2.4)$$

and g is the Gray code equivalent of the binary decomposition of the number n. The first 8 ones of Walsh functions arranged in increasing value of the number of zero crossings are shown in Fig.2.2. As it is shown in Fig.2.2 Walsh functions are symmetrical about their mid or zero point.

The product of any two Walsh functions yields a third Walsh function [11].

$$\mathbf{Wal}(i,t) \cdot \mathbf{Wal}(j,t) = \mathbf{Wal}(k,t) \qquad (2.5)$$

where

$$k = i \oplus j \qquad (2.6)$$

The sign $\oplus$ represents modulo-2 addition, and modulo-2 addition is a Binary addition with no carry, as

$$0 + 0 = 0 \quad 0 + 1 = 1 \quad 1 + 0 = 1 \quad 1 + 1 = 0$$

Since the modulo-2 addition of a number with itself is equal to zero, the product of a Walsh function with itself yields **Wal**(0,t) written as

$$\mathbf{Wal}(i,t) \cdot \mathbf{Wal}(i,t) = \mathbf{Wal}(0,t) \tag{2.7}$$

Since, Walsh functions are complete and orthogonal

$$\int_T \mathbf{Wal}(i,t) \cdot \mathbf{Wal}(j,t) \quad dt = \begin{cases} C \text{ if } i=j \\ 0 \text{ if } i \neq j \end{cases} \tag{2.8}$$

Since C=1, for the case where i=j, Walsh functions are said to be an orthonormal set [11].

2.4 FUNCTION ORDERING

The set of Walsh functions is generally classified into three groups [12]. These groups differ from one another in such a way that their appearances having the same index number are different. The three types of orderings are; Walsh ordering, Paley ordering, and Hadamard ordering.

2.4.1 WALSH ORDERING

This is the ordering which was originally defined by Walsh. In this order, the functions are arranged in increasing number of zero crossings (Fig.2.2). It has the advantage of having resemblance of the sine and cosine functions when we define **Cal** and **Sal** functions [13], as

$$\mathbf{Cal}(n,t) = \mathbf{Wal}(2n,t) \tag{2.9a}$$

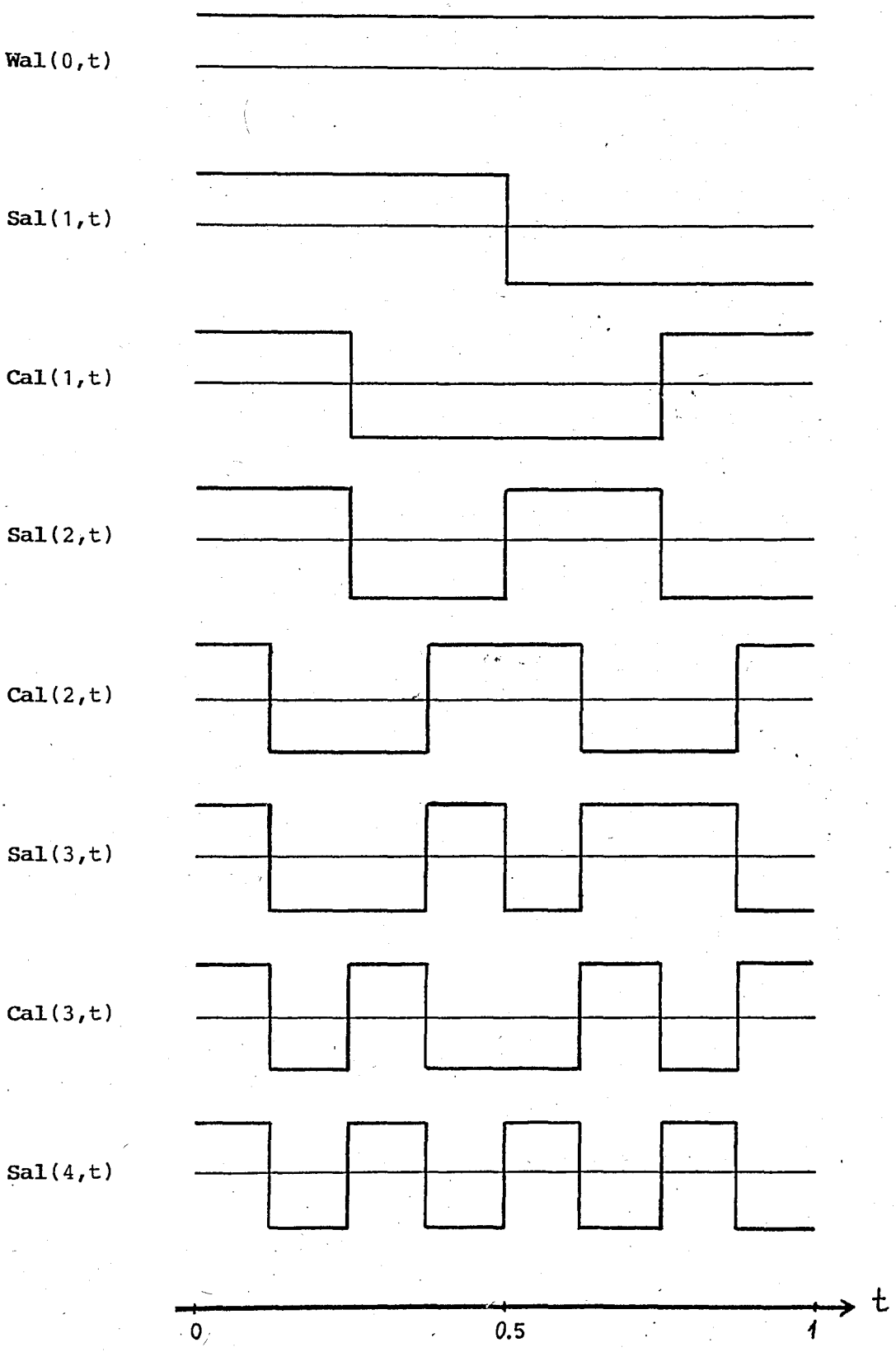


FIGURE 2.3 A set of Cal and Sal functions

$$\mathbf{Sal}(n,t) = \mathbf{Wal}(2n-1,t) \qquad (2.9b)$$

Using the two equations above , the first 8 ones of the Walsh functions are shown in Fig.2.3. It is appearent from Fig.2.2 that the sequency of a Walsh function is greater than the sequency of the preceeding Walsh function and has exactly one more zero crossing in the same time interval.

2.4.2 PALEY ORDERING

We can evaluate this ordering when we obtain the Walsh functions from Rademacher functions. It is stated that the Walsh transforms possess a better convergence when they are arranged in Paley order [12]. The relationship between Walsh and Paley orderings can be formalised as

$$\mathbf{Pal}[g(n),t] = \mathbf{Wal}(n,t) \qquad (2.10)$$

where g(n) is a function based on the Gray code. If we consider the Binary equivalent of order number, n, as

$$n = (b_m, b_{m-1}, \ldots\ldots b_1, b_0)_2 \qquad (2.11)$$

then g(n) can be evaluated as

$$g_i = b_i \oplus b_{i+1} \qquad (2.12)$$

Thus, for n=6 we obtain g=5 and we can write

$$Pal(5,t) = Wal(6,t)$$

The relationship between **Wal** and **Pal** ordering is also formalised as

$$Wal(b(n),t) = Pal(n,t) \qquad (2.13)$$

where b(n) is evaluated from Gray code to Binary conversion of the order number, n. The first 8 ones of Walsh functions in Paley order are shown in Fig.2.4.

2.4.3. HADAMARD ORDERING

The set of Hadamard ordered Walsh functions is obtained when we evaluate the Walsh functions from the Hadamard matrices [14]. The relationship between Walsh ordering and Hadamard ordering of the Walsh functions can be formalized as

$$Wal_h(g(<(n)>,t) = Wal(n,t) \qquad (2.14)$$

where $g<(n)>$ stands for the bit reversed value of the Binary to Gray code conversion of the order number n. Thus, in a sequence consisting of 8 elements, for n=1 we get b(n)=001 and g(n)=001. After bit reversal operation, we get $g<(n)>$=100, and this is equal to 4 in decimal system. Then, we can write as

$$Wal_h(4,t) = Wal(1,t)$$

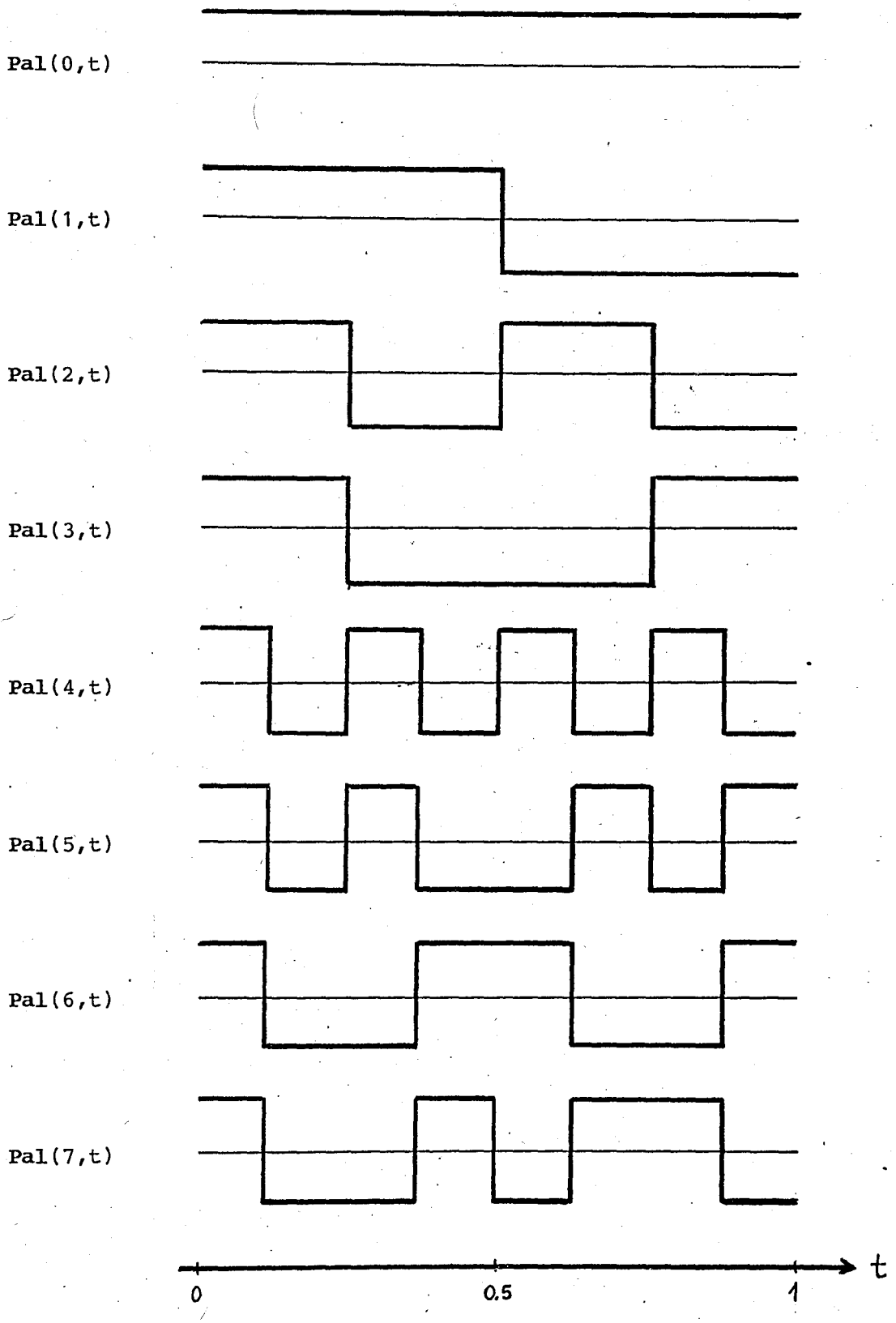


FIGURE 2.4 A set of Walsh functions in Paley order

The relationship between Walsh ordering and Hadamard ordering of a Walsh function can be formalised as

$$\mathbf{Wal}(b(\langle n\rangle),t) = \mathbf{Wal}_h(n,t) \qquad (2.15)$$

where $\langle n\rangle$ is obtained from the bit reversal of n and, $b(\langle n\rangle)$ is the Gray code to Binary conversion of $\langle n\rangle$. The first 8 ones of the Walsh fuctions in Hadamard order are shown in Fig.2.5.

One can easily deduce that the relationship between Paley and Hadamard orderings is simply a bit reversal operation,as

$$\mathbf{Pal}(\langle n\rangle,t) = \mathbf{Wal}_h(n,t) \qquad (2.16)$$

$$\mathbf{Wal}_h(\langle n\rangle,t) = \mathbf{Pal}(n,t) \qquad (2.17)$$

2.5 BIT REVERSAL OPERATION

The bit reversed sequence of a given series can be evaluated finding the mirror appearance of its sequence number represented in the Binary system. This can be summarized as,

Decimal	Binary	Binary bit-reversed	Decimal bit-reversed
0	00	00	0
1	01	10	2
2	10	01	1
3	11	11	3

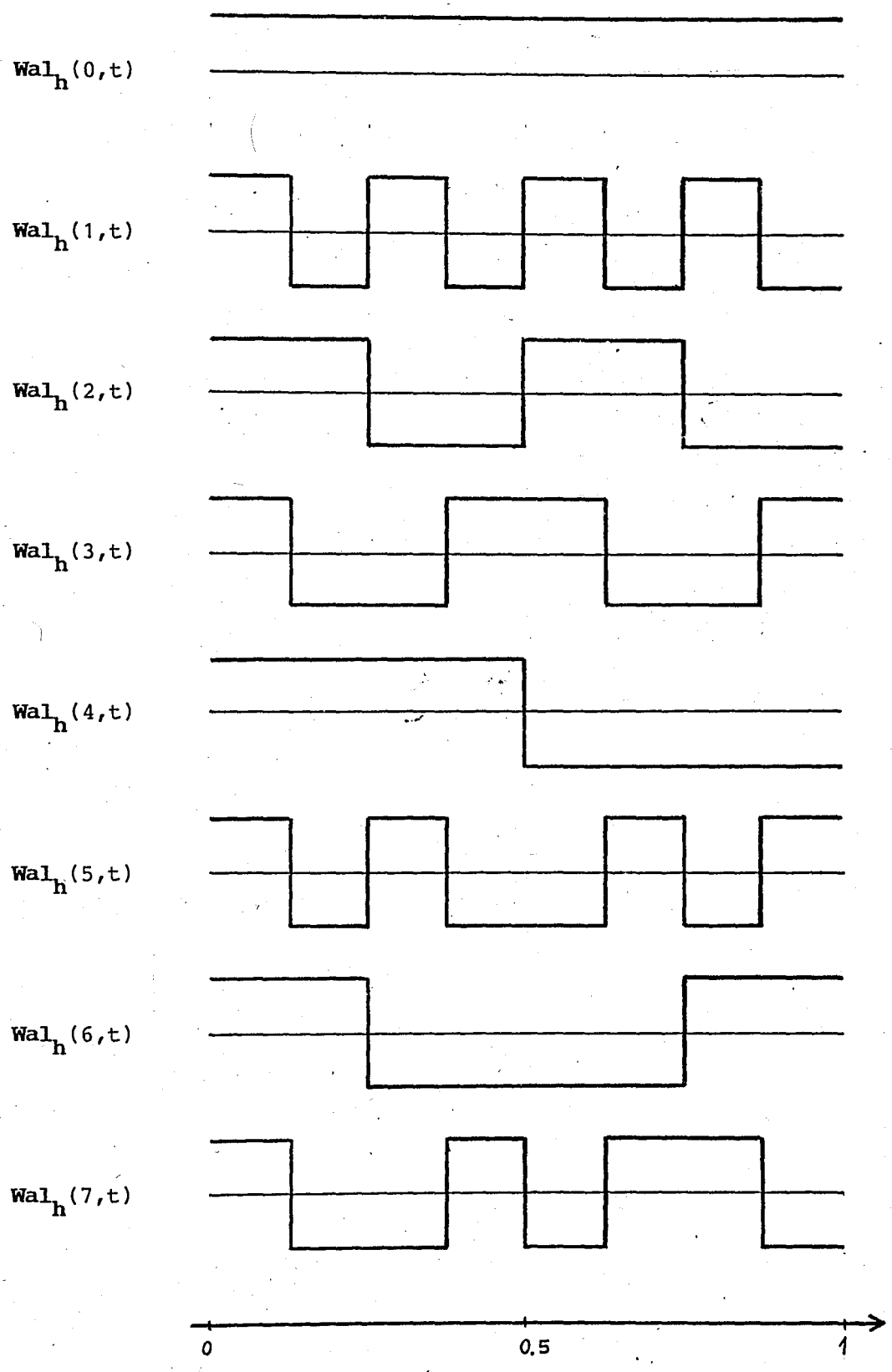


FIGURE 2.5 A set of Walsh functions in Hadamard order

or

Decimal	Binary	Binary bit-reversed	Decimal bit-reversed
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

As it is shown above ,the rearrangement is made by taking into account the largest sequence number. In this study, the number of data in a sequence is always taken as an integer power of two. The rearrangement of a series into bit reversed order can be made through a suitable routing in the case of hardware derivation or a bit reversal software routine. Program-1 is written to rearrange a given series into bit reversed order. The flow diagram of this computer program is shown in Fig.2.6.

2.6 GENERATION OF WALSH FUNCTIONS

The Walsh functions can be evaluated in several different ways, each of which has its own particular advantages. In this study, we are going to evaluate the Walsh functions from the product of the Rademacher functions for the continuous case, and from the Hadamard matrices for the discrete case.

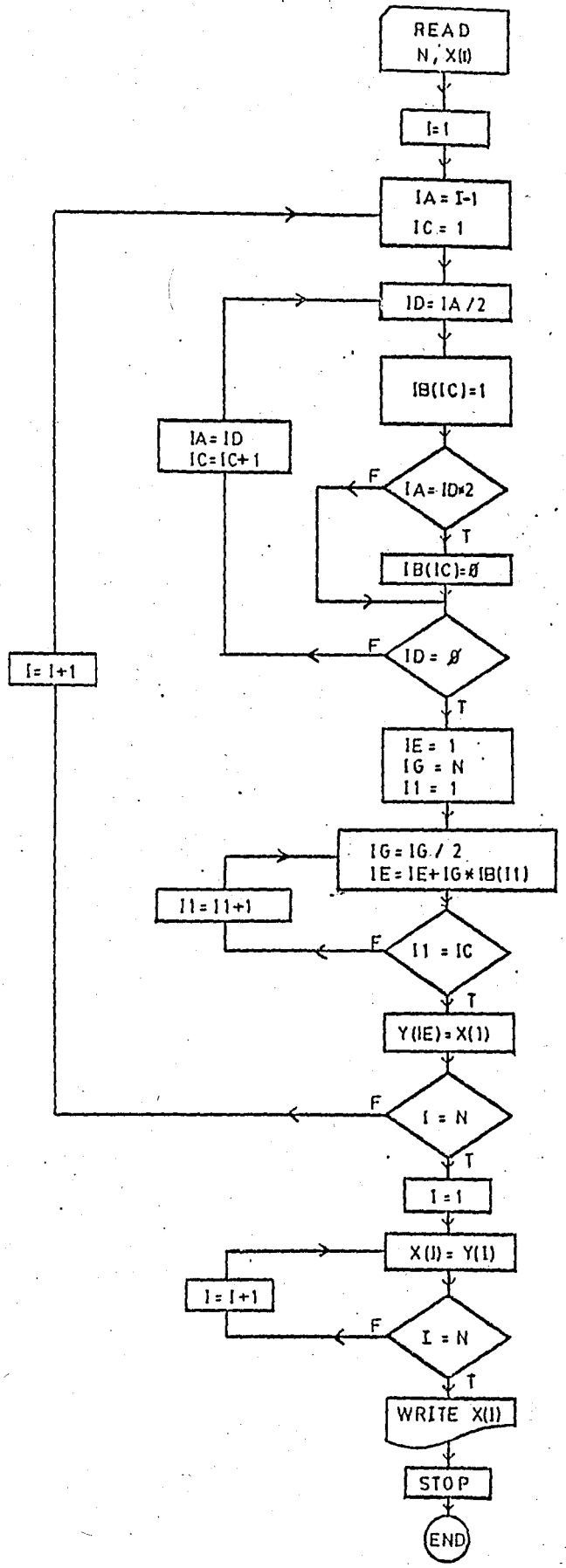


FIGURE 2.6 Flow diagram for bit reversal operation

2.6.1 WALSH FUNCTION GENERATION FROM THE RADEMACHER FUNCTIONS

The Rademacher functions form an incomplete series having odd symmetry [8]. In particular, a complete set of Walsh functions in Paley order can be obtained from the selected Rademacher function products [15]. The product series for the Rademacher functions is expressed as

$$\mathbf{Pal}(n,t) = \prod_{i=1}^{m+1} b_{i-1} \cdot R(i,t) \qquad (2.18)$$

We have seen that the relationship between Walsh and Paley orderings is simply a Binary to Gray code conversion of the sequence number. In order to find the value of bit position i of the sequence number n to be expressed in the Gray code we need to add bit i to bit i+1 of the original binary number. Thus, from Eq.(2.11), by expressing n as a string of binary bits and expressing this in the Gray code, we can write

$$n = (g_m \; g_{m-1} \; g_{m-2} \; \ldots\ldots\ldots g_1 \; g_0 \;)_2 \qquad (2.19)$$

where

$$g_i = b_i \oplus b_{i+1}$$

Then, we can write

$$\mathbf{Wal}(n,t) = \prod_{i=1}^{m+1} (b_i \oplus b_{i-1}) \cdot R(i,t) \qquad (2.20)$$

If we use 0 instead of -1 in the representation of the Rademacher

functions, this product becomes a modulo-2 summation written as

$$\mathbf{Wal}(n,t) = \oplus \sum_{i=1}^{m+1} (b_i \oplus b_{i-1}) \cdot R(i,t) \qquad (2.21)$$

Hence, to find **Wal**(6,t), we first express it in the Binary code as b(n)=110. Then, by rearranging this in the Gray code we get g(n)=101. As it is seen from the Gray code equivalent, the second bit position is equal to zero. This means that (from Eq.2.20)

$$\mathbf{Wal}(6,t) = R(3,t) \; . \; R(1,t)$$

or (from Eq.2.21)

$$\mathbf{Wal}(6,t) = R(3,t) \oplus R(2,t)$$

A graphical illustration of this function's derivation is shown in Fig.2.7.

The hardware scheme to generate any Walsh function with an index in the range $0 < n \leqslant 31$ is shown in Fig.2.8. A Binary counter is used to generate Rademacher functions. The Binary equivalent $b_4b_3b_2b_1b_0$ of the desired index, n, is loaded into the input register. The Binary equivalent of the index number n is converted to the Gray code equivalent $g_4g_3g_2g_1g_0$ by the modulo-2 adders. The output of the modulo-2 adders controls the transfer of the Rademacher functions through the AND gates. The output of the AND gates controls the final modulo-2 adder.

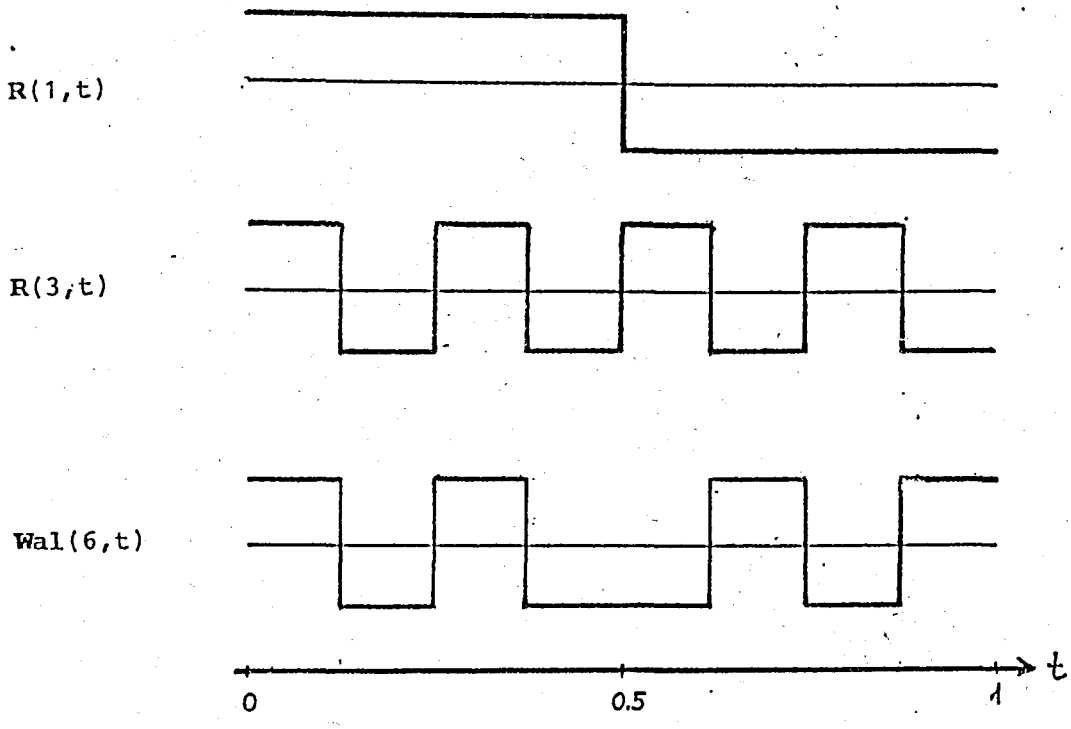


FIGURE 2.7 Derivation of Wal(6,t) from the Rademacher functions

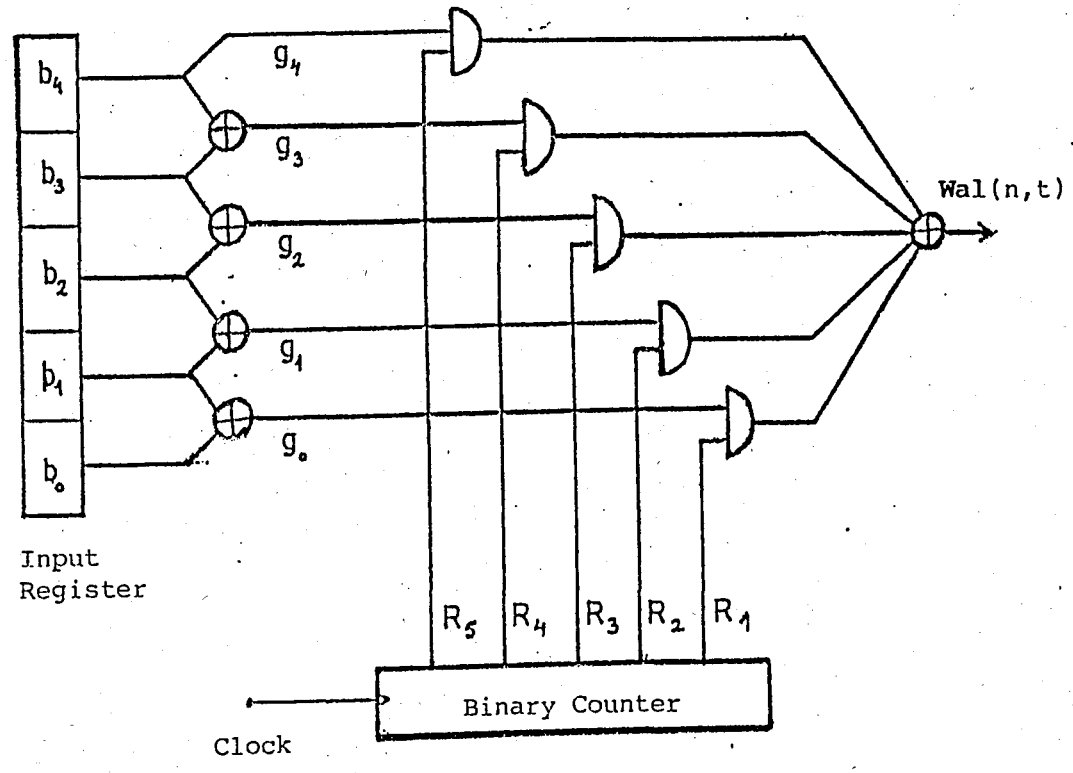


FIGURE 2.8 A scheme for generating **Wal**(n,t) from Binary equivalent of its index number, n

The output of the final modulo-2 adder is the desired Walsh function of index n.

2.6.2 WALSH FUNCTION GENERATION FROM HADAMARD MATRICES

The Hadamard matrix is a square array whose array coefficients consist of only +1's and -1's, and where its rows (and columns) are orthogonal to one another. In a symmetrical Hadamard matrix, it is possible to interchange rows and columns or to change the sign of each element in a row without affecting the orthogonality properties. This makes it possible to obtain a symmetrical Hadamard matrix whose first row and first column contain only positive 1's. The matrix obtained in this way is known as the "normal form" for the Hadamard matrix [14]. The lowest order Hadamard matrix is of the order 2 written as

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (2.22)$$

Higher order matrices, which are restricted to having integer powers of two, can be obtained from the recursive relationship given by

$$H_N = H_{N/2} \boxtimes H_2 \qquad (2.23)$$

where the sign $\boxtimes$ denotes the direct or Kronecker product and N is an integer power of two. The Kronecker product can be defined as the replacement of each element in the matrix to be evaluated by the matrix H_2.

Thus, to evaluate the H_4 matrix, we can write as

$$H_4 = H_2 \otimes H_2 \tag{2.24}$$

Then, we replace each of the 1's and -1's of the matrix in H_2 by the complete matrix of H_2, or by its inverse. Thus, we can evaluate H_4 as

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{2.25}$$

Furthermore, if we now replace each element in the H_4 matrix by an H_2 matrix, we obtain an H_8 matrix.

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{2.26}$$

Each row of a Hadamard matrix of order N represents a sampled Walsh function in the Hadamard order. Program-2 is written to evaluate the

Hadamard matrices of any order restricted having the integer powers of 2. The flow diagram of this computer program is shown in Fig.2.9.

2.7 WALSH TRANSFORM

It can be stated that every function f(t), which is integrable,is capable of being represented by a Walsh series defined over the open interval (0,1) [11], as

$$f(t) = a_0 + a_1 \mathbf{Wal}(1,t) + a_2 \mathbf{Wal}(2,t) + \ldots\ldots \qquad (2.27)$$

where the coefficients are given by

$$a_n = \int_0^1 f(t).\mathbf{Wal}(n,t)\,dt \qquad (2.28)$$

Therefore, we can define a transform pair as

$$F(n) = \int_0^1 f(t).\mathbf{Wal}(n,t)\,dt \qquad (2.29)$$

and

$$f(t) = \sum_{n=0}^{\infty} F(n).\mathbf{Wal}(n,t) \qquad (2.30)$$

This definition is applied to a continuous function limited in time over the interval $0 \leqslant t \leqslant 1$. For numerical use, it is convenient to consider a discrete series of N terms which are set up by sampling the continuous functions at N equally spaced points over the open interval (0,1). In order that the properties of the continuous and discrete

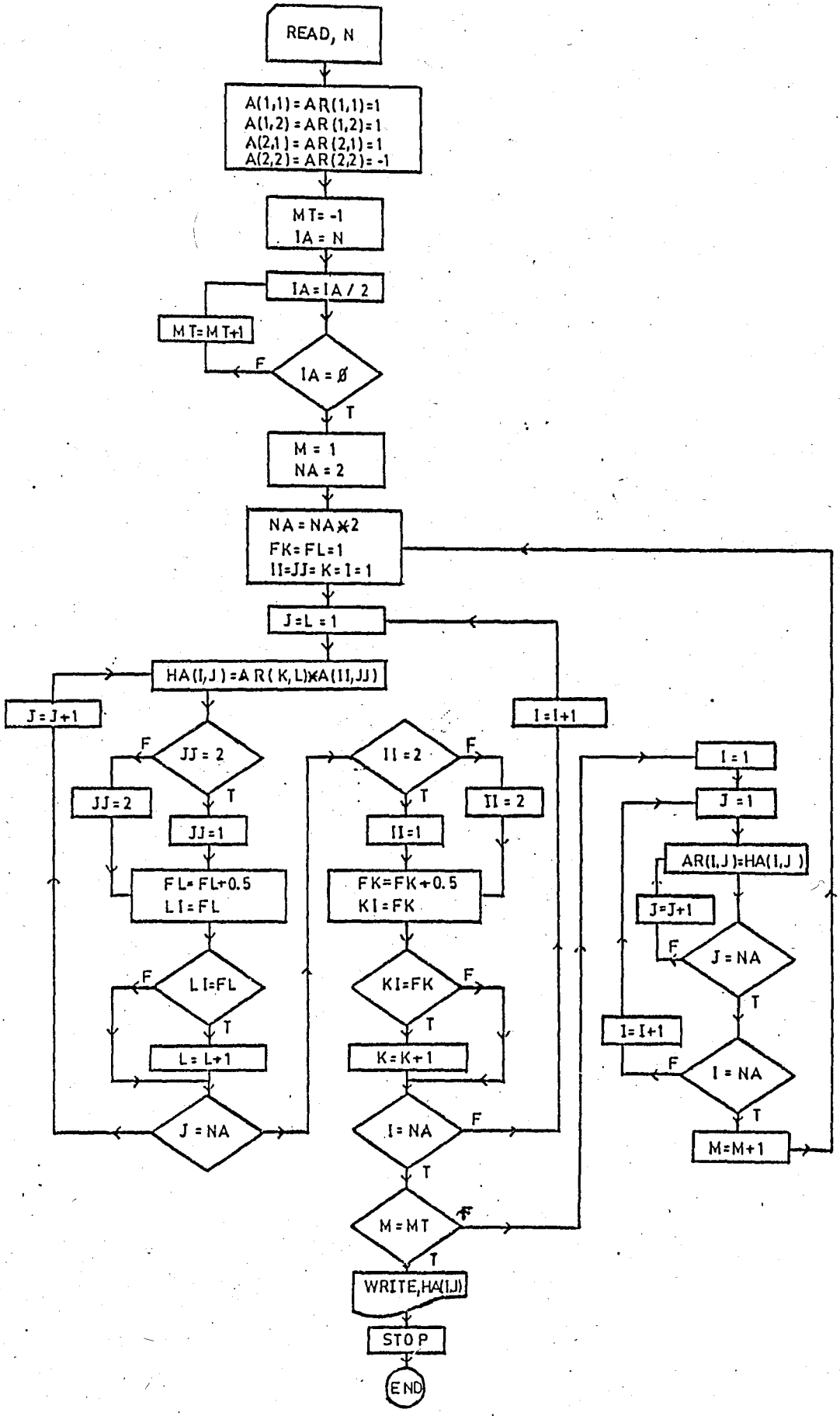


FIGURE 2.9 Flow diagram for hadamard matrix evaluation

systems should correspond, we must take N equal to an integer power of two, i.e. $N=2^n$. The integration shown in Eq.(2.29) may then be replaced by summation. Then, we can define the finite discrete Walsh transform pair as

$$X(n) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) \cdot \mathbf{Wal}(n,i) \qquad (2.31)$$

$$i = 0,1,2,3,\ldots\ldots,N-1$$

and

$$x(n) = \sum_{n=0}^{N-1} X(n) \cdot \mathbf{Wal}(n,i) \qquad (2.32)$$

$$i = 0,1,2,3,\ldots\ldots\ldots,N-1$$

As it is seen appearently, the Walsh transform involves only additions and subtractions. Therefore it has distinctive advantages in adaptability to digital implementation and also computer analysis.

2.7.1 MATRIX DEFINITION OF WALSH TRANSFORM

A Walsh matrix can be evaluated from appropriately sampling of continuous Walsh functions, or directly from the Hadamard matrices.Then, we can represent the Walsh matrix as **W**(n), where n is an integer number. This should have a certain order, N, that must satisfy

$$n = \log_2 N \qquad (2.33)$$

Let a continuous function be sampled over a time interval at N

equidistant points and the discrete function be represented by means of a vector x(n), where

$$x(n) = [\ x_0 \quad x_1 \quad \ldots\ldots\ldots\ x_{N-1}\ ]^T \tag{2.34}$$

Then, the Walsh transform of a sampled function can be defined [16], as

$$X(n) = \frac{1}{N} \cdot W(n) \cdot x(n) \tag{2.35}$$

where, the product X(n) is a vector consisting of the Walsh coefficients as

$$X(n) = [\ X_0 \quad X_1 \quad \ldots\ldots\ldots\ldots\ X_{N-1}\ ]^T \tag{2.36}$$

The inverse Walsh transform can be defined as

$$x(n) = W(n) \cdot X(n) \tag{2.37}$$

In fact, the matrix cited above is the Hadamard matrix in Walsh order. In computer applications, it is more convenient to use Walsh matrix in the Hadamard order rather than any other order. Program-3 is written to compute the Walsh transform of a given series. To compute the inverse transformation, TR should be set to -1. Fig. 2.10 shows the flow diagram of Program-3.

Example : Let N be equal to 8, and x(n) be

$$x(n) = [\ 0.7 \quad 1.0 \quad 0.7 \quad 0.0 \quad -0.7 \quad -1.0 \quad -0.7 \quad 0.0\ ]^T$$

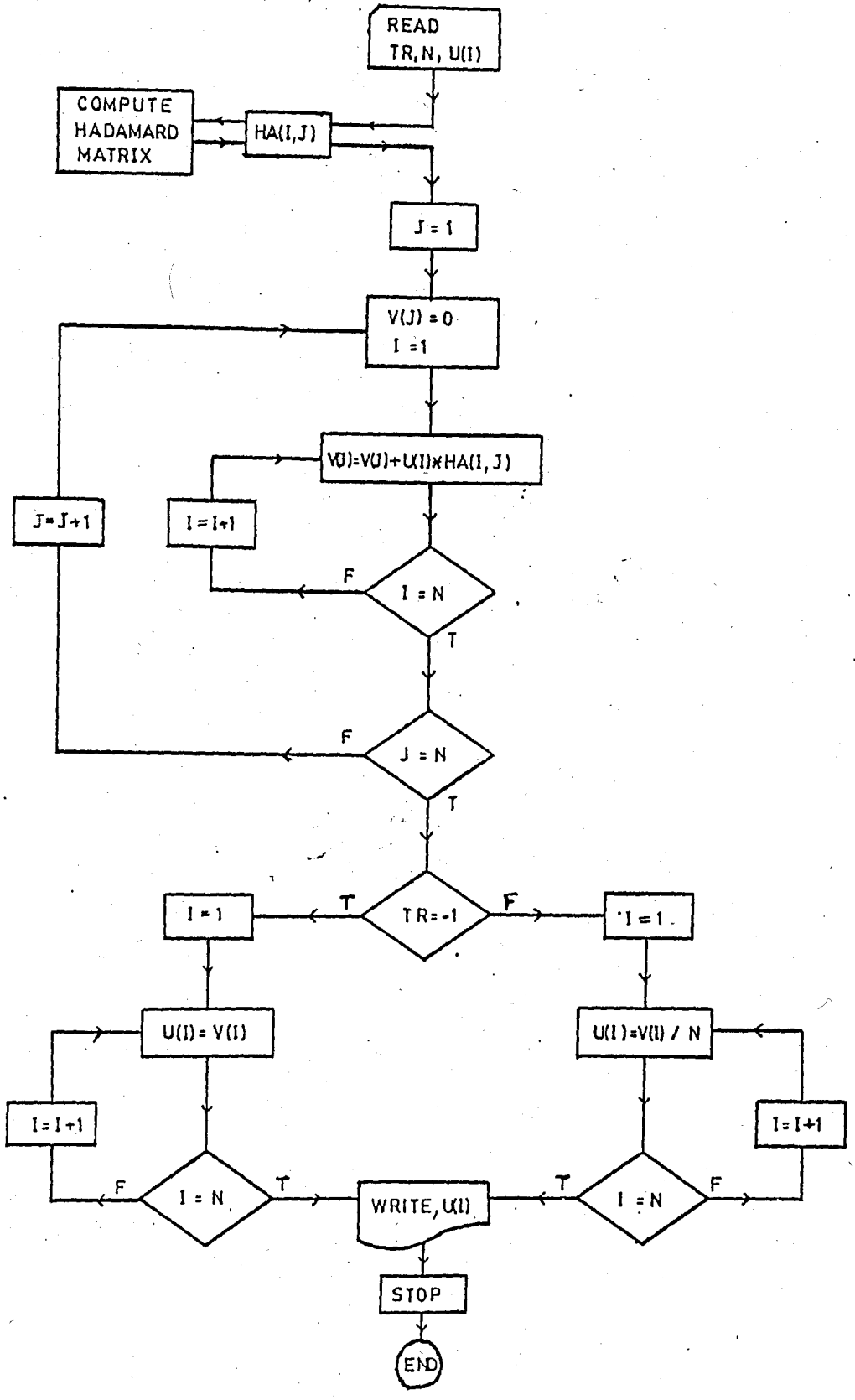


FIGURE 2.10 Flow diagram for Walsh transform

Then, the transform coefficients are found as

$$X(n)=\frac{1}{8}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 0.7 \\ 1.0 \\ 0.7 \\ 0.0 \\ -0.7 \\ -1.0 \\ -0.7 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.6 \\ 0.25 \\ 0.0 \\ 0.0 \\ -0.25 \\ 0.1 \\ 0.0 \end{bmatrix}$$

In order to verify that the transformation in Eq.(2.35) is unique, we substitude X(n), in Eq.(2.37) to obtain

$$x(n)=\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 0.0 \\ 0.6 \\ 0.25 \\ 0.0 \\ 0.0 \\ -0.25 \\ 0.1 \\ 0.0 \end{bmatrix} = \begin{bmatrix} 0.7 \\ 1.0 \\ 0.7 \\ 0.0 \\ -0.7 \\ -1.0 \\ -0.7 \\ 0.0 \end{bmatrix}$$

2.7.2 FAST WALSH TRANSFORM

Fast Walsh Transform can be derived using matrix factoring or matrix partitioning techniques [14]. Now, we are going to illustrate the

matrix partitioning technique for the case N=8. For N=8, Eq.(2.35) yields

$$X(3) = \frac{1}{8} \cdot W(3) \cdot x(3) \tag{2.38}$$

Using Eq.(2.23), W(3) is expressed in terms of W(2) to obtain

$$X(3) = \frac{1}{8} \begin{bmatrix} W(2) & W(2) \\ W(2) & -W(2) \end{bmatrix} x(3) \tag{2.39}$$

Using matrix partitioning it follows that

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \frac{1}{8} W(2) \cdot \begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} \tag{2.40}$$

$$\begin{bmatrix} X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \frac{1}{8} W(2) \cdot \begin{bmatrix} x_1(4) \\ x_1(5) \\ x_1(6) \\ x_1(7) \end{bmatrix} \tag{2.41}$$

where

$$x_1(1) = x(1) + x(4+1) \qquad 1 = 0, 1, 2, 3 \tag{2.42}$$

$$x_1(1) = x(1-4) - x(1) \qquad 1 = 4, 5, 6, 7 \tag{2.43}$$

The sequence of additions and subtractions in Eqs.(2.40) and (2.41) are shown in the signal flow graph in Fig. 2.11, and are indicated by Iteration # 1. Again, the application of Eq.(2.23) to Eqs.(2.40) and (2.41) results in

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \frac{1}{8} \begin{bmatrix} W(1) & W(1) \\ W(1) & -W(1) \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} \tag{2.44}$$

$$\begin{bmatrix} X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \frac{1}{8} \begin{bmatrix} W(1) & W(1) \\ W(1) & -W(1) \end{bmatrix} \begin{bmatrix} x_1(4) \\ x_1(5) \\ x_1(6) \\ x_1(7) \end{bmatrix} \tag{2.45}$$

From matrix partitioning indicated in Eq.(2.44) and (2.45), we obtain

$$\begin{bmatrix} x(0) \\ x(1) \end{bmatrix} = \frac{1}{8} W(1) \begin{bmatrix} x_1(0) + x_1(2) \\ x_1(1) - x_1(3) \end{bmatrix} = \frac{1}{8} W(1) \begin{bmatrix} x_2(0) \\ x_2(1) \end{bmatrix} \tag{2.46}$$

$$\begin{bmatrix} x(2) \\ x(3) \end{bmatrix} = \frac{1}{8} W(1) \begin{bmatrix} x_1(0) - x_1(2) \\ x_1(1) + x_1(3) \end{bmatrix} = \frac{1}{8} W(1) \begin{bmatrix} x_2(2) \\ x_2(3) \end{bmatrix} \tag{2.47}$$

$$\begin{bmatrix} x(4) \\ x(5) \end{bmatrix} = \frac{1}{8} W(1) \begin{bmatrix} x_1(4) + x_1(5) \\ x_1(6) - x_1(7) \end{bmatrix} = \frac{1}{8} W(1) \begin{bmatrix} x_2(4) \\ x_2(5) \end{bmatrix} \tag{2.48}$$

$$\begin{bmatrix} x(6) \\ x(7) \end{bmatrix} = \frac{1}{8} W(1) \begin{bmatrix} x_1(4) - x_1(5) \\ x_1(6) + x_1(7) \end{bmatrix} = \frac{1}{8} W(1) \begin{bmatrix} x_2(6) \\ x_2(7) \end{bmatrix} \tag{2.49}$$

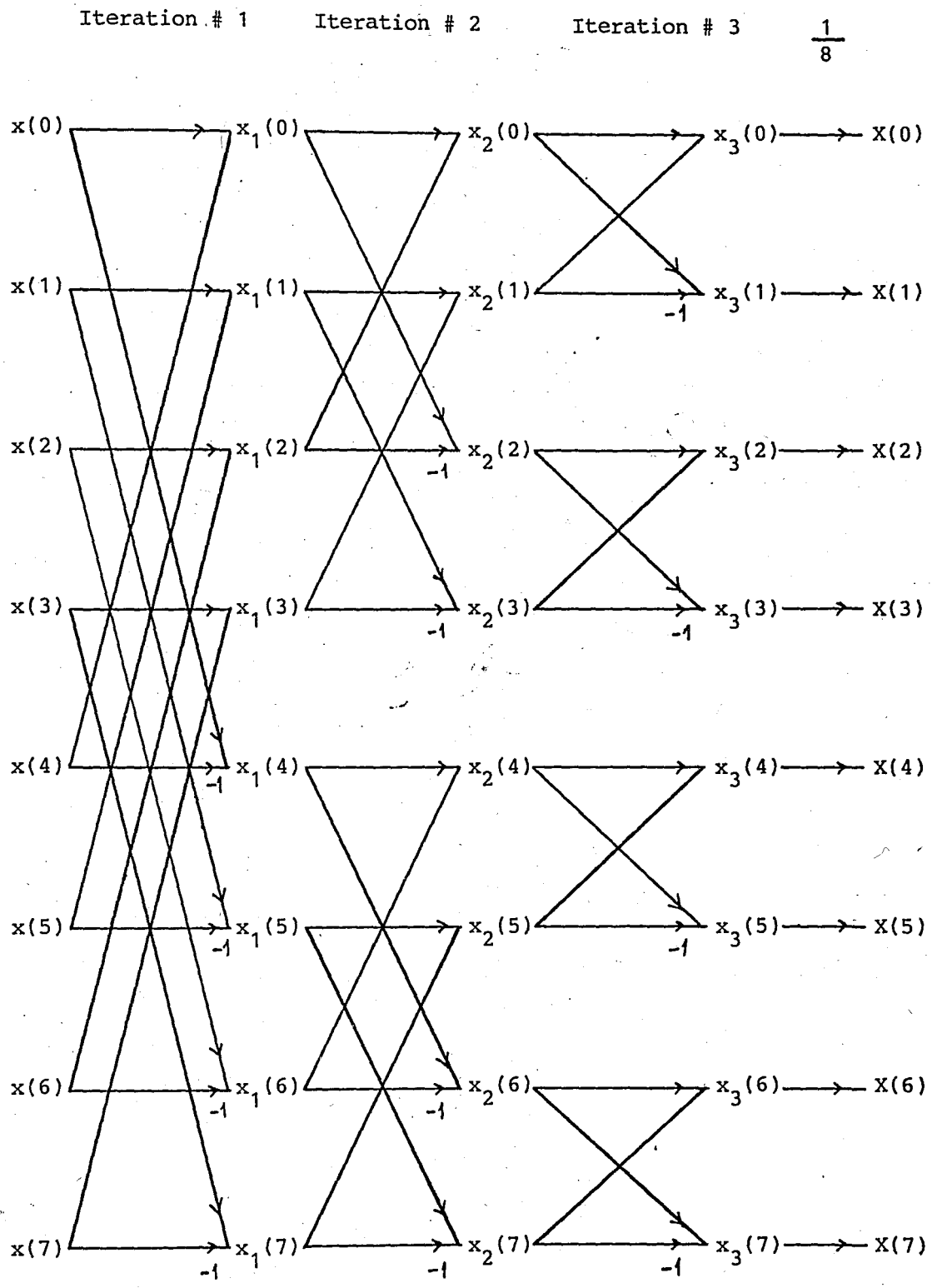


FIGURE 2.11 Fast Walsh transform signal flow graph

The sequence of additions and subtractions in Eqs. (2.46) through (2.49) are indicated by Iteration # 2 in Fig. 2.11. Since

$$W(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

these equations reduces to

$$\begin{aligned}
X(0) &= \frac{1}{8}\left[x_2(0) + x_2(1)\right] = \frac{1}{8}x_3(0) \\
X(1) &= \frac{1}{8}\left[x_2(0) - x_2(1)\right] = \frac{1}{8}x_3(1) \\
X(2) &= \frac{1}{8}\left[x_2(2) + x_2(3)\right] = \frac{1}{8}x_3(2) \\
X(3) &= \frac{1}{8}\left[x_2(2) - x_2(3)\right] = \frac{1}{8}x_3(3) \\
X(4) &= \frac{1}{8}\left[x_2(4) + x_2(5)\right] = \frac{1}{8}x_3(4) \\
X(5) &= \frac{1}{8}\left[x_2(4) - x_2(5)\right] = \frac{1}{8}x_3(5) \\
X(6) &= \frac{1}{8}\left[x_2(6) + x_2(7)\right] = \frac{1}{8}x_3(6) \\
X(7) &= \frac{1}{8}\left[x_2(6) - x_2(7)\right] = \frac{1}{8}x_3(7)
\end{aligned} \qquad (2.50)$$

The sequence of additions and subtractions in Eq.(2.50) are indicated by Iteration # 3 in Fig. 2.11. From the signal flow graph, it is apparent that apart from the 1/8 multiplier, only additions and subtractions are performed. The number of additions and subtractions required to compute the eight Walsh Transform coefficients is $8.\log_2 8 = 24$.

It should be mentioned that there is a drowback with the Fast Walsh Transform. Since the Hadamard matrices possess a simple recursive

formula, it is necessary to rearrange the input into bit reversed order before the transformation. The output coefficients are in bit reversed order compared with the coefficients of an increasing ordered sequency. Therefore, the output needs the rereversing of the bits to reorder the coefficients in increasing sequency [17,18].

Program-4 is written to compute the Fast Walsh Transform of a sampled function. The number of data in the array to be transformed should be an integer power of two. The flow diagram of Program-4 is shown in Fig. 2.12.

Generally, for the case $N=2^n$ the following remarks can be made:

1. The total number of iterations is given by $n=\log_2 N$. If IT is an iteration index, then IT = 1, 2, 3, ..., n.

2. The $IT^{\underline{th}}$ iteration results in 2^{IT-1} groups with $N/2^{IT-1}$ members in each group. Half of the members in each group are associated with an addition operation, while the remaining half are associated with a subtraction operation.

3. The total number of arithmetic operations to compute all of the transform coefficients is approximately $N\log_2 N$.

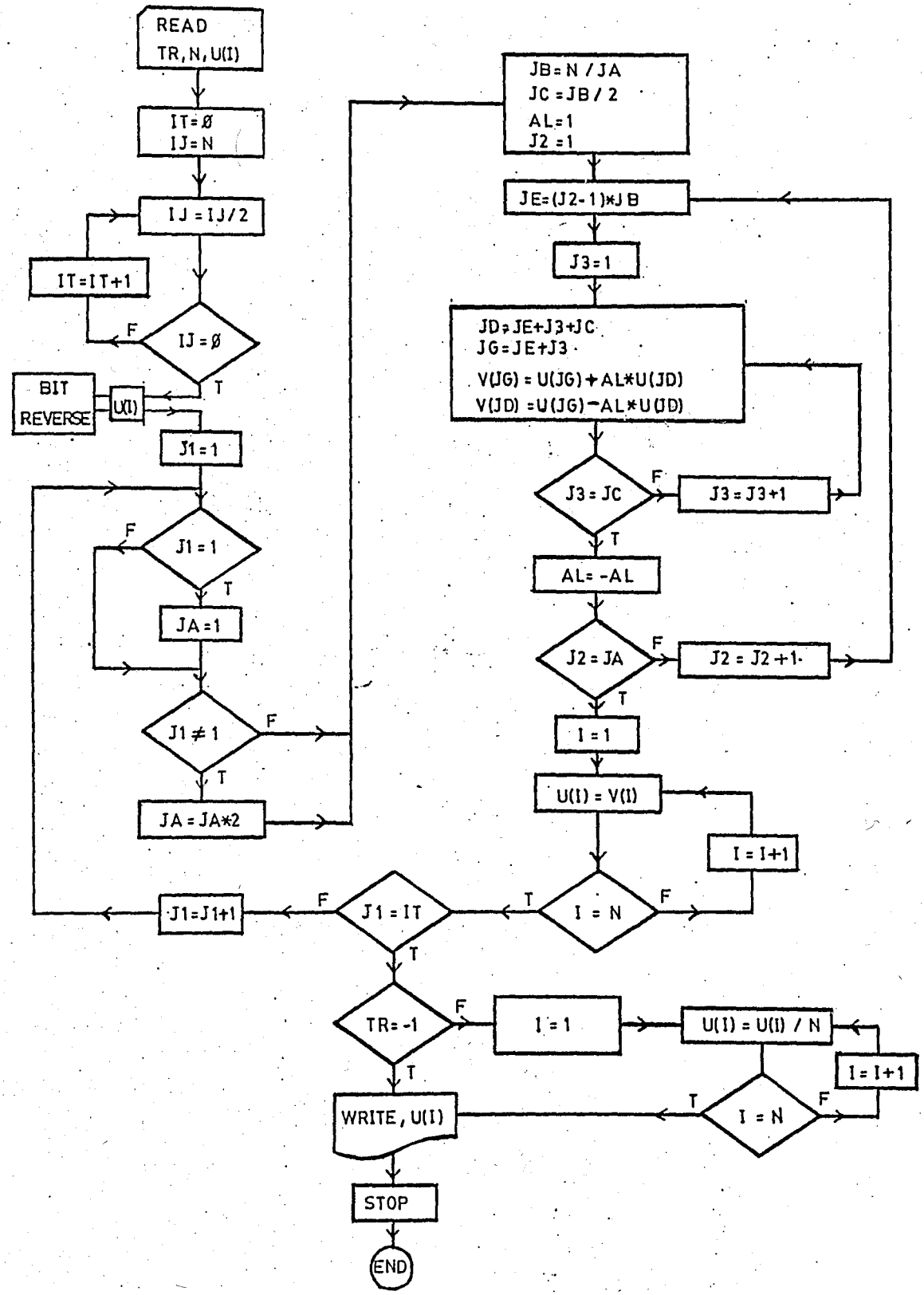


FIGURE 2.12 Flow diagram for fast Walsh transform

III. ADAPTIVE FILTERS

3.1 ADAPTIVE PROCESSING

Conventional signal processing systems for the extraction of information from an incoming signal, such as matched filters, operate in an open-loop fashion, Fig.3.1. This means that, the same processing function is carried out in the present time interval regardless of whether the resulted output is correct in the preceeding time interval.

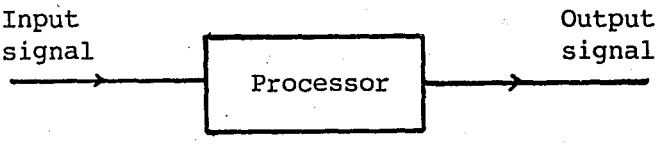


Figure 3.1 Open-loop processing system

An adaptive signal processing system can operate in an open-loop or in a closed-loop fashion with respect to the processed signals. In an open-loop adaptive signal processing system, the processing function

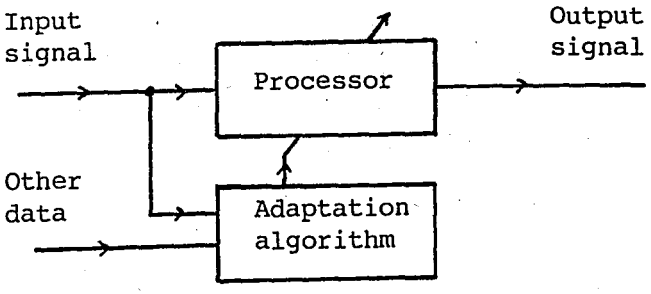


Figure 3.2 Open-loop adaptive processing system

changes while the incoming signals change, Fig.2.2. The incoming signals are applied to a computational algorithm and the results are used to set the adjustments of the adaptive system to improve the output signal characteristics. In a closed-loop adaptive signal processing system, on the other hand, the adjustments are changed by procesing both the inputs and resulted signals to optimize the system performance. A closed-loop adaptive signal processing system is shown in Fig. 3.3.

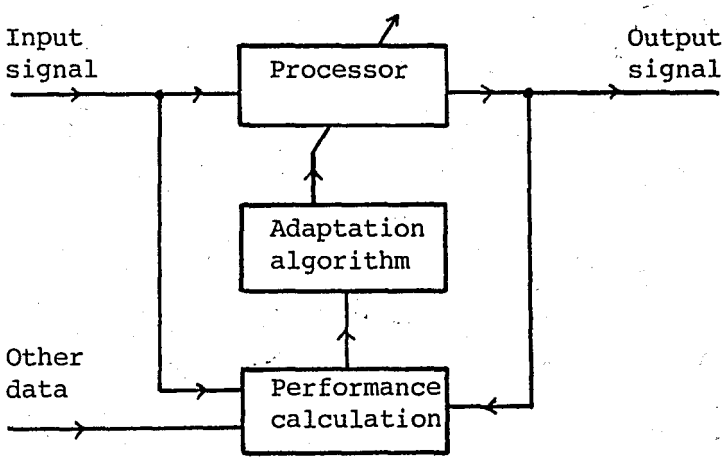


Figure 3.3 Closed-loop adaptive processing system

3.2 ADAPTIVE FILTERING

The usual method of estimating a signal corrupted by noise is to pass it through a filter that tends to supress the noise while leaving the signal relatively unchanged. Subtracting noise from a received signal would seem to be a dangerous procedure. It could result in an increase in output noise power if it is done improperly. If, however, filtering and subtraction are controlled by an appropriate adaptive

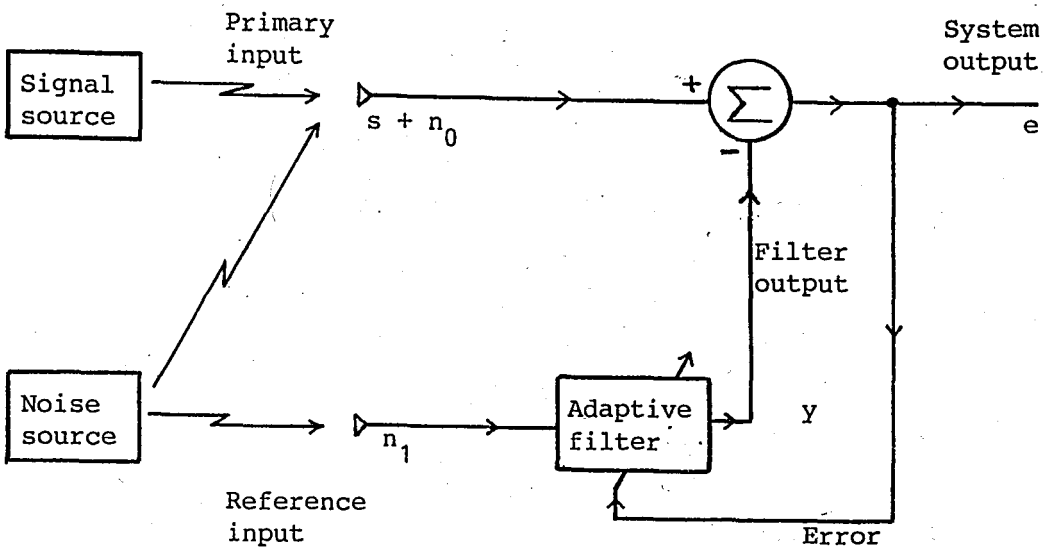


Figure 3.4 Adaptive noise cancelling system

process, noise reduction can be performed with little risk of distorting the signal or increasing the output noise level. Fig.3.4 shows the basic problem and the adaptive noise cancelling solution to it. A signal, s, is transmitted over a channel to a sensor. This sensor also receives a noise, n_0, uncorrelated with the signal. The combined signal and noise form the primary input to the canceller. A second sensor receives another noise, n_1, uncorrelated with the signal but correlated in same unknown way with the noise n_0. This sensor provides the reference input to the canceller. The noise n_1 is filtered to produce an output, y. This output is subtracted from the primary input $s + n_0$ to produce the system output.

$$e = s + n_0 - y \qquad (3.1)$$

Assume that s, n_0, n_1 and y are statistically stationary and have zero means. Squaring Eq.(3.1), we can obtain

$$e^2 = s^2 + (n_0 - y)^2 + 2\, s\, (n_0 - y) \tag{3.2}$$

Taking expectation of both sides of Eq.(3.2), and taking into account that, s is uncorrelated with n_0 and y, it yields

$$\begin{aligned} E[e^2] &= E[s^2] + E[(n_0 - y)^2] + 2\, E[s(n_0 - y)] \\ &= E[s^2] + E[(n_0 - y)^2] \end{aligned} \tag{3.3}$$

The signal power $E[s^2]$ will be unaffected as the filter is adjusted to minimize $E[e^2]$. Accordingly, the minimum output power is

$$\min E[e^2] = E[s^2] + \min E[(n_0 - y)^2] \tag{3.4}$$

When the filter is adjusted so that $E[e^2]$ is minimized, $E[(n_0-y)^2]$ is also minimized. Adjusting or adapting the filter to minimize the total output power causes minimizing the output noise power. Since the signal at the output remains constant, minimizing the total output power maximizes the output signal-to-noise ratio [19].

As it is seen from Eq.(3.3) that the smallest possible output power is $E[e^2] = E[s^2]$. When this is performed, we get

$$E[(n_0 - y)^2] = 0 \tag{3.5}$$

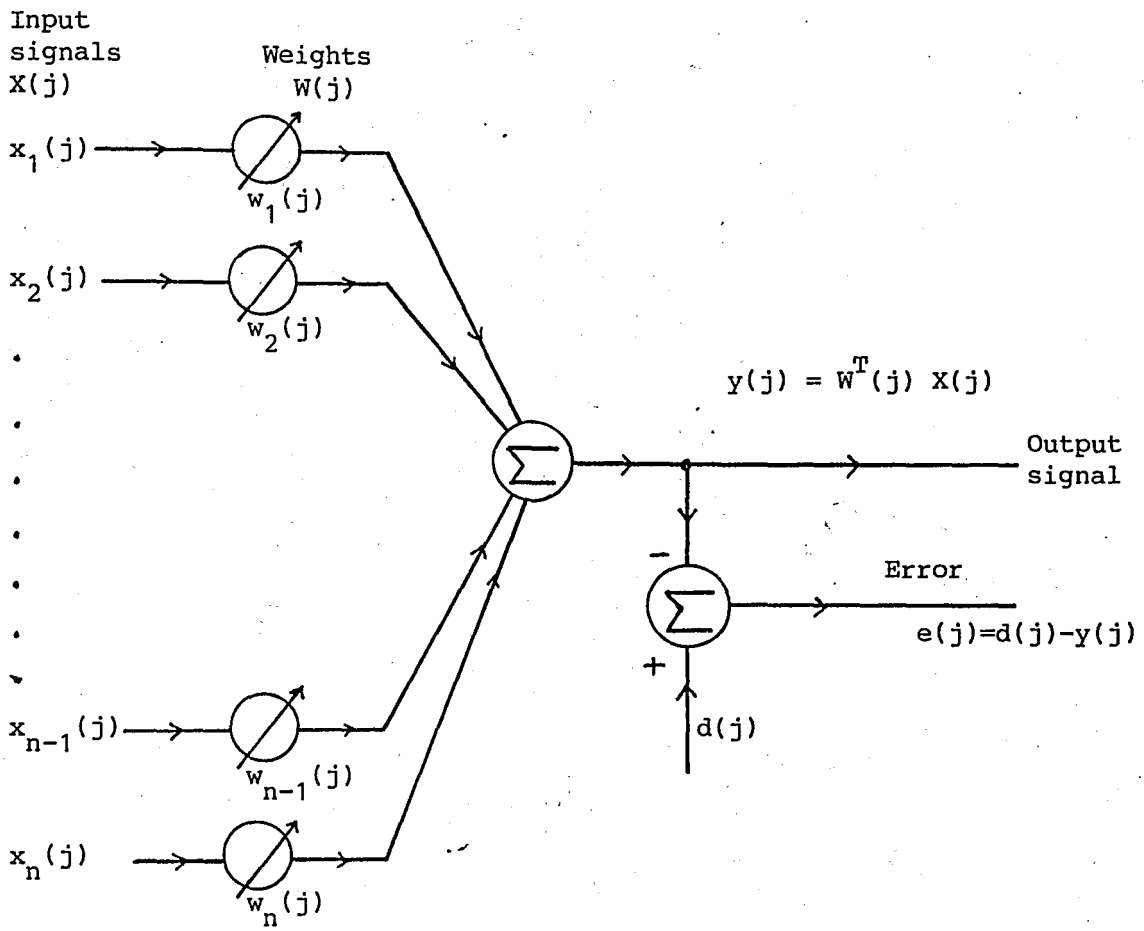


Figure 3.5 Adaptive linear combinatorial system

Therefore, $y = n_0$ and $e = s$. In this case, minimizing the output power causes the output signal to be noise free.

3.3 ADAPTIVE FILTER OPERATION

The analysis of the adaptive filter can be developed by considering the adaptive linear combinatorial system shown in Fig. 3.5. In the

system shown in Fig. 3.5, a set of stationary input signals is weighted and summed to form an output signal. The input signals in the set are assumed to occur simultaneously and discretely in time. The j^{th} set of input signals can be represented by means of a vector X(j) as

$$X(j) = [\ x_1(j)\ x_2(j)\ \ldots\ldots\ldots\ldots\ x_n(j)\ ]^T \tag{3.6}$$

The set of weights can be represented by the vector W(j) as

$$W(j) = [\ w_1(j)\ w_2(j)\ \ldots\ldots\ldots\ldots\ w_n(j)\ ]^T \tag{3.7}$$

Then, the output is

$$y(j) = \sum_{i=1}^{n} w_i(j) \cdot x_i(j) \tag{3.8}$$

This can be written in matrix form as

$$y(j) = W^T(j) \cdot X(j) = X^T(j) \cdot W(j) \tag{3.9}$$

Assuming that d(j) is the desired response for the j^{th} set of input signals, the error at the j^{th} time is written as

$$e(j) = d(j) - y(j) = d(j) - W^T(j) \cdot X(j) \tag{3.10}$$

The square of this error is

$$e^2(j) = d^2(j) - 2\, d(j)\, X^T(j) \cdot W(j) + W^T(j) \cdot X(j)\, X^T(j) \cdot W(j) \tag{3.11}$$

The expected value of $e^2(j)$ is the mean-square error.

$$E[e^2(j)] = \hat{d}^2(j) - 2\,\phi(x,d)\,W(j) + W^T(j)\,\phi(x,x)\,W(j) \qquad (3.12)$$

Where, $\phi(x,d)$ is the vector of cross-correlations between the input signals and the desired response, and $\phi(x,x)$ is the correlation matrix of the input signals [20].

It can be observed in Eq.(3.12) that for stationary input signals, the mean-square error is a second-order function of the weights. The mean-square error performance function can be visualised as a bowl-shaped surface, a parabolic function of the weight variables. The adaptive filtering has the meaning of continually seeking the bottom of the bowl where the error is minimum. The minimum of the mean-square error function can be found by differentiating Eq.(3.12) with respect to the weight vector.

$$\nabla[\hat{e}^2(j)] = -2\,\phi(x,d) + 2\,\phi(x,x).W(j) \qquad (3.13)$$

To find the optimal weigth vector, W_{LMS} we should set the gradient equal to zero. Then, we get

$$\phi(x,d) = \phi(x,x).W_{LMS} \qquad (3.14)$$

and

$$W_{LMS} = \phi^{-1}(x,x).\phi(x,d) \qquad (3.15)$$

Eq.(3.15) is the Wiener-Hopf equation in matrix form. The mean-square

error can be obtained by substituting Eq.(3.15) into Eq.(3.12), as

$$\hat{e}^2_{min} = \hat{d}^2(j) - W^T_{LMS} \cdot \emptyset(x,d) \qquad (3.16)$$

3.4 THE LMS ADAPTATION ALGORITHM

The purpose of the adaptation process is to find an exact or approximate solution to the Wiener-Hopf equation. In practice it is not possible to find a perfect solution of Eq.(3.15) because of the fact that an infinite statistical sample would be required to estimate perfectly the elements of the correlation matrices. The LMS algorithm can be used to find an approximate solution to Eq.(3.15). This algorithm does not even require squaring, averaging, or differentiating in order to make use of gradients of mean-square error functions.

When using the LMS, changes in the weight vector are made along the reverse direction of the estimated gradient vector. Accordingly

$$W(j+1) = W(j) - \mu \cdot \hat{\nabla} [\hat{e}^2(j)] \qquad (3.17)$$

Where

$W(j)$ = Weight vector before adaptation

$W(j+1)$ = Weight vector after adaptation

μ = A scalar constant controlling the rate of convergence and stability $0 < \mu < 1$

$\hat{\nabla}[\hat{e}^2(j)]$ = Estimate of gradient of $E[e^2] = \hat{e}^2$ with respect to W, with $W = W(j)$

One method for obtaining the estimated gradient of the mean-square error function is to take the gradient of a single time sample of the squared error; that is

$$\hat{\nabla}[\hat{e}^2(j)] = \nabla[e^2(j)] = 2\, e(j) \cdot \nabla[e(j)] \tag{3.18}$$

From Eq.(3.10)

$$\nabla[e(j)] = \nabla[d(j) - W^T(j) \cdot X(j)] = - X(j) \tag{3.19}$$

Thus,

$$\hat{\nabla}[\hat{e}^2(j)] = - 2\, e(j), X(j) = -2\, [\, d(j) - W^T(j) \cdot X(j)\,] \cdot X(j) \tag{3.20}$$

Using the gradient estimating formula above, the weight iteration rule given by Eq.(3.17) becomes

$$W(j+1) = W(j) + 2\, \mu e(j) \cdot X(j) \tag{3.21}$$

and the weight vector is obtained by adding the present weight vector to the input vector scaled by the value of the error. This is the LMS algorithm and the convergence is guarantied only if μ is chosen as

$$0 < \mu < \frac{1}{\lambda_{max}} \tag{3.22}$$

Where λ_{max} is the largest eigenvalue of the input correlation matrix [21].

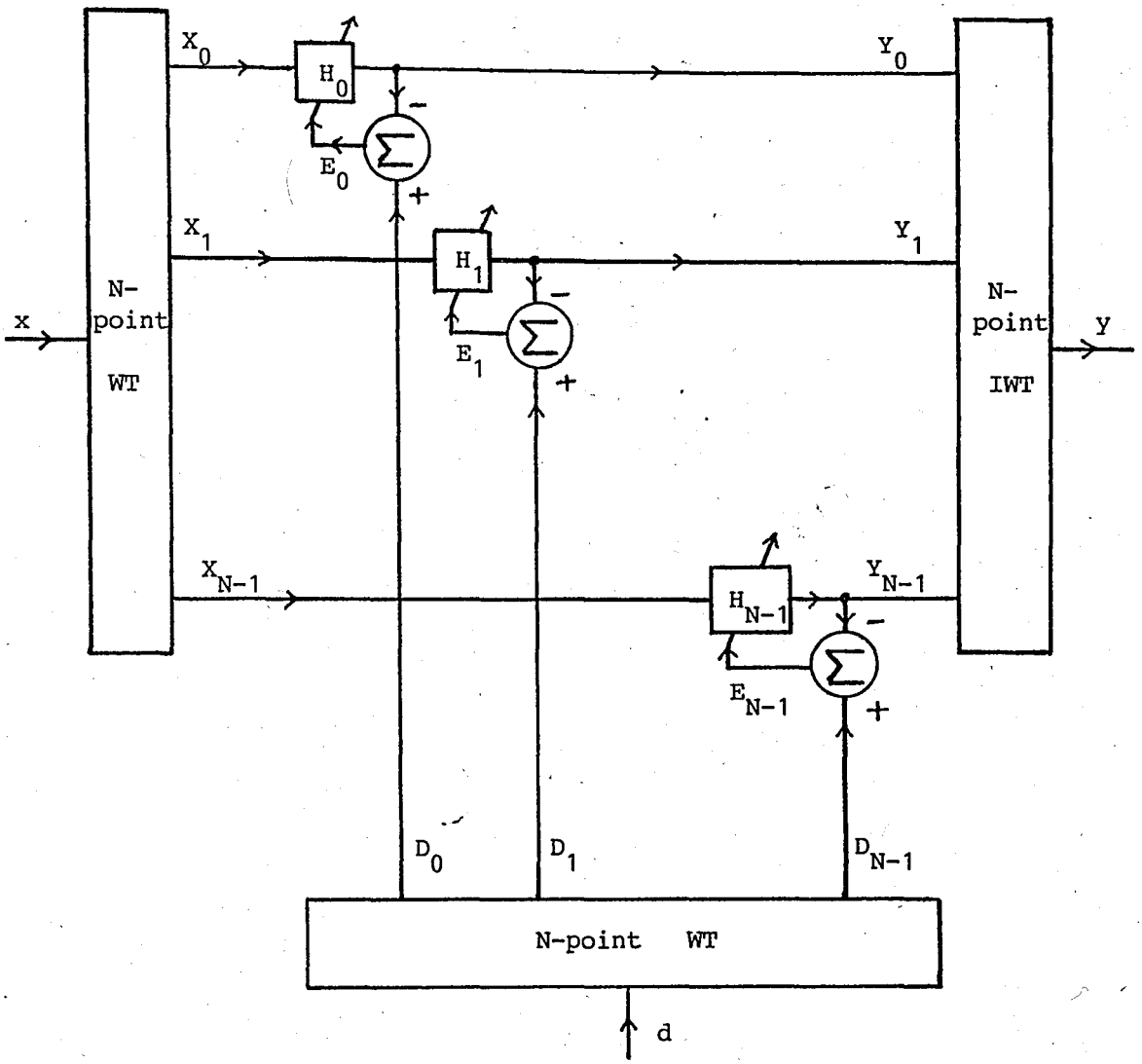


Figure 3.6 Transform domain adaptive filter

3.5 TRANSFORM DOMAIN LMS ALGORITHM

A block diagram of the transform domain adaptive filter is shown in Fig.3.6. The input, x, is sampled at N equidistant points to form an input signal vector x(n) where

$$x(n) = [\, x_0 \; x_1 \; x_2 \; \ldots\ldots\ldots\ldots\ldots \; x_{N-1} \,]^T \qquad (3.23)$$

Then, the input vector is transformed into another vector, X(n)

$$X(n) = [\ X_0\ X_1\ X_2\ \ldots\ldots\ldots\ldots\ldots X_{N-1}\ ]^T \qquad (3.24)$$

using an orthogonal transformation, as

$$X(n) = \frac{1}{N}\ \mathbf{W}(n).x(n) \qquad (3.25)$$

where $\mathbf{W}(n)$ is a unitary matrix of rank N. That is

$$\mathbf{W}(n).\mathbf{W}^T(n) = N\ I \qquad (3.26)$$

The elements of the vector X(n) are multiplied by the elements of the transform domain weight vector

$$H(n) = [\ H_0\ H_1\ \ldots\ldots\ldots\ldots\ldots\ H_{N-1}\ ]^T \qquad (3.27)$$

to form the adaptive filter output, Y(n). The output and the corresponding error signal are given as

$$Y(n) = [\ Y_0\ Y_1\ Y_2\ \ldots\ldots\ldots\ldots\ Y_{N-1}\ ]^T \qquad (3.28)$$

and

$$E(n) = [\ E_0\ E_1\ E_2\ \ldots\ldots\ldots\ldots\ E_{N-1}\ ]^T \qquad (3.29)$$

where

$$Y_i = H_i.X_i \qquad (3.30)$$

and

$$E_i = D_i - Y_i \qquad i = 0, 1, 2, \ldots\ldots, N-1 \tag{3.31}$$

respectively. Eq.(3.31) denotes that, there are N orthogonal error outputs rather than single global error of the time domain approach. To minimize this error, the transform domain weight update equation for the k^{th} iteration can be exressed as

$$H_{i,k+1} = W_{i,k} + 2 \mu X_i \cdot E_i \tag{3.32}$$

The system output is given as

$$y(n) = W^T(n) \cdot Y(n) \tag{3.33}$$

and the error is

$$e = d - y \tag{3.34}$$

Since the filter processes the data in N point blocks, each weight is updated once for each block. This provides superior convergence properties by comparison with time domain approach [22,23,24].

IV. SIMULATION OF WALSH DOMAIN ADAPTIVE FILTER

The implementation of a transform domain adaptive filter can be done by transforming the input signal, multiplying the transformed input signal by a set of stored transform domain weights, followed by inverse transform processing. Although this appears at first sight to be more complicated than time domain processing, the transform domain application requires many fewer multiplication when the filter length is large (e.g., >16).

Simulation of the Walsh domain adaptive filter is made by taking into account the transform domain LMS algorithm (Fig. 3.6). The input signal and the desired response are accumulated in buffer memories to form N-point data blocks. They are then transformed by N-point fast Walsh transforms. Each of the fast Walsh transform outputs comprises a set of N real numbers. The weighted input transform values are subtracted from the desired response transform values to form N error signals. There are N weights, and each of them is independently updated for each data block. The weighted inputs form the output transform values and they are fed to an inverse fast Walsh transform operator to produce the output signal.

A substantial reduction in computation is obtained with the Walsh domain adaptive filter as compared with conventional time domain adaptive filtering. This fact can be demonstrated by examining the number of multiply operations required to process a fixed amount of data. To produce N output data points with an N-tap, time domain LMS

adaptive filter requires $2N^2$ multiplications. To produce the same amount of output with this Walsh domain filter requires three N-point fast Walsh transforms and 2N multiplications for the weighting and updating. Then, the total number of multiplications is $3N\log_2 N + 2N$. For large filters, the computational savings produced by Walsh domain filter is substantial.

Program-5 is written to filter a signal, corrupted by white noise, adaptively in Walsh domain. Fig.4.1 shows the flow diagram of this computer program. The whole program is given in the Appendix as Program-5. It consists of two main parts, one is the basic routine and the other is the subroutines. Although the filter length can be chosen as any number being an integer power of two, the computer memory restricts the maximum to 64. Because of the plotter capacity, the maximum number of iterations is limited at 1600.

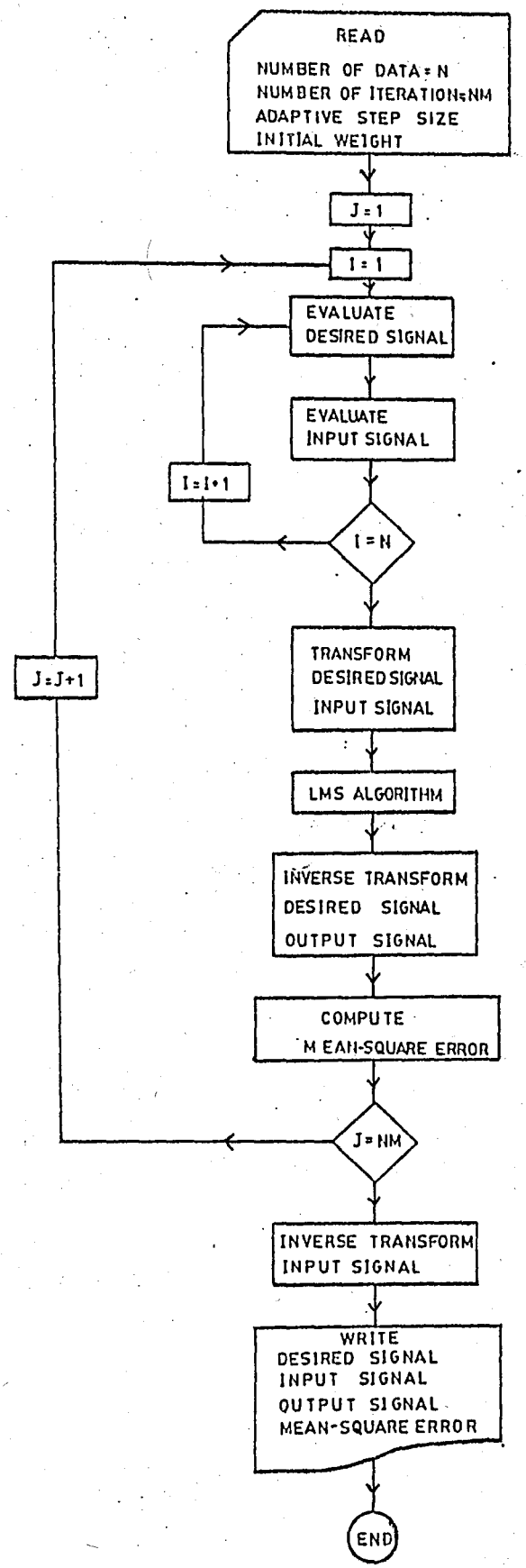
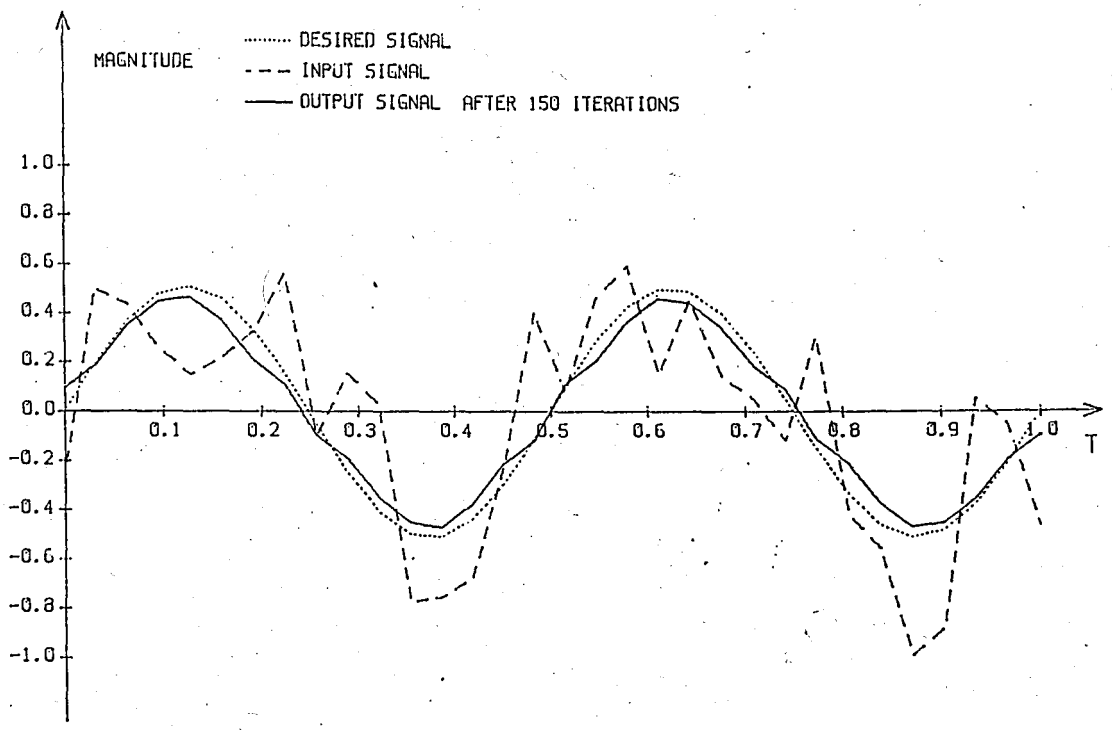


FIGURE 4.1 Flow diagram for Walsh domain adaptive filter

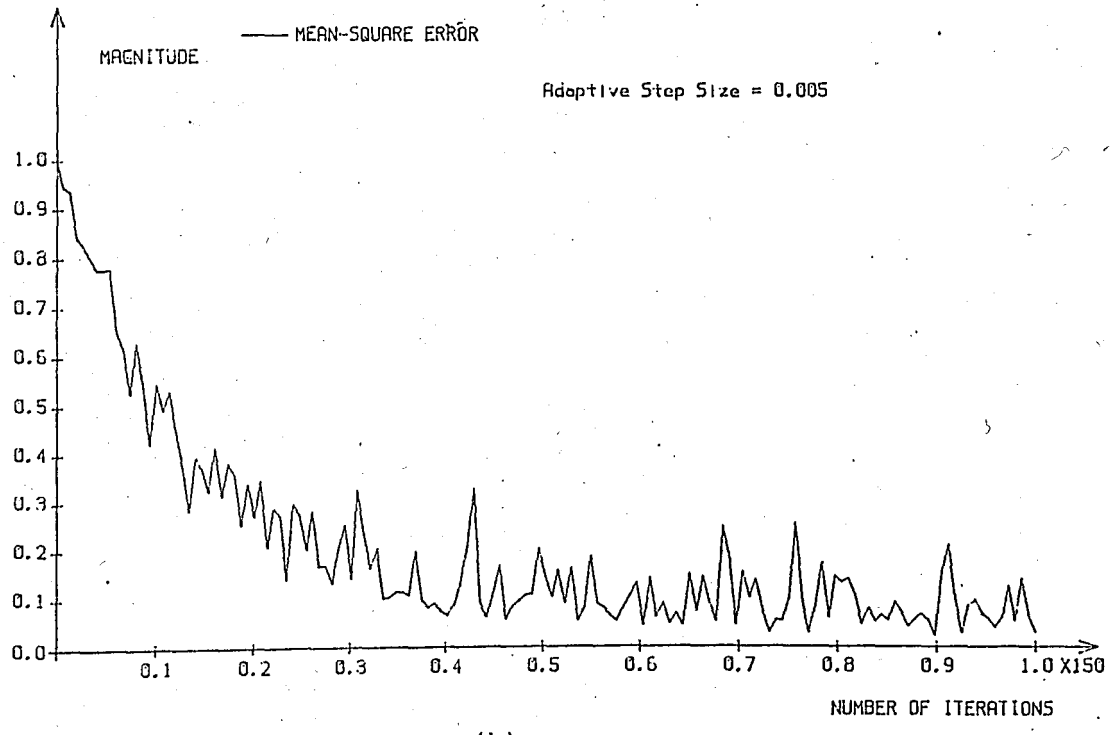
V. RESULTS AND CONCLUSIONS

5.1 ADAPTIVE FILTERING OF A SINE WAVE

The results of the Walsh domain adaptive filtering of a sine wave are shown in Fig. 5.1. The input signal is a sine wave corrupted by white noise. Fig. 5.1(b) shows the mean-square error versus iteration number. As it is shown in Fig.5.1(b), the mean-square error decreases very rapidly at first. With an adaptive step size of 0.005, the adaptation process is completed after about 60 iterations. After the adaptation process is completed, the change in the rate of decrease of the mean-square error becomes insignificant. The waveforms of the input, desired, and output signals are shown in Fig. 5.1(a) for the last iteration. Although there is quite a difference between the input and the desired signal, the output signal is almost the same as the desired signal. Walsh spectra of the waveforms in Fig. 5.1(a) are shown in Fig.5.2. Fig. 5.2(a) shows that there are only a few Walsh transform coefficients to represent the desired signal in the Walsh domain. Lots of the Walsh transform coefficients of the desired signal are equal or very close to zero. In order to make the output noise free, we have to eliminate the Walsh transform coefficients of the input signal which are not present in the Walsh spectrum of the desired signal, and try to evaluate the Walsh transform coefficients of the desired signal from the Walsh transform coefficients of the input signal. This is performed by the transform domain LMS algorithm. At the begining of the adaptation an arbitrary value is chosen as the elements of the weight vector (initial weight). The products of the input signal Walsh transform coefficients

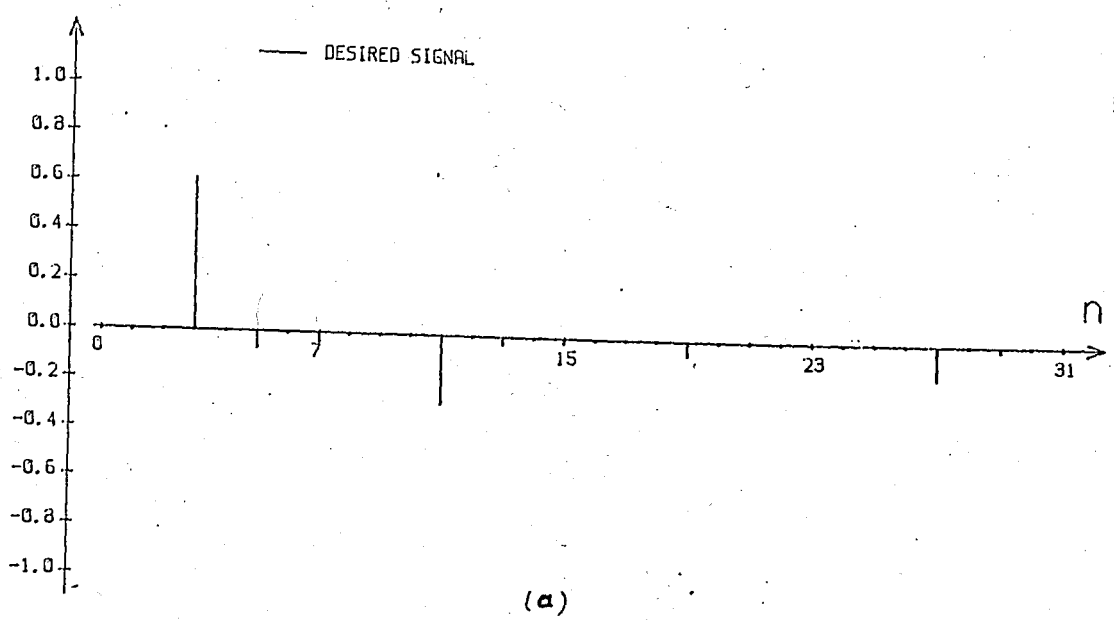


(a)

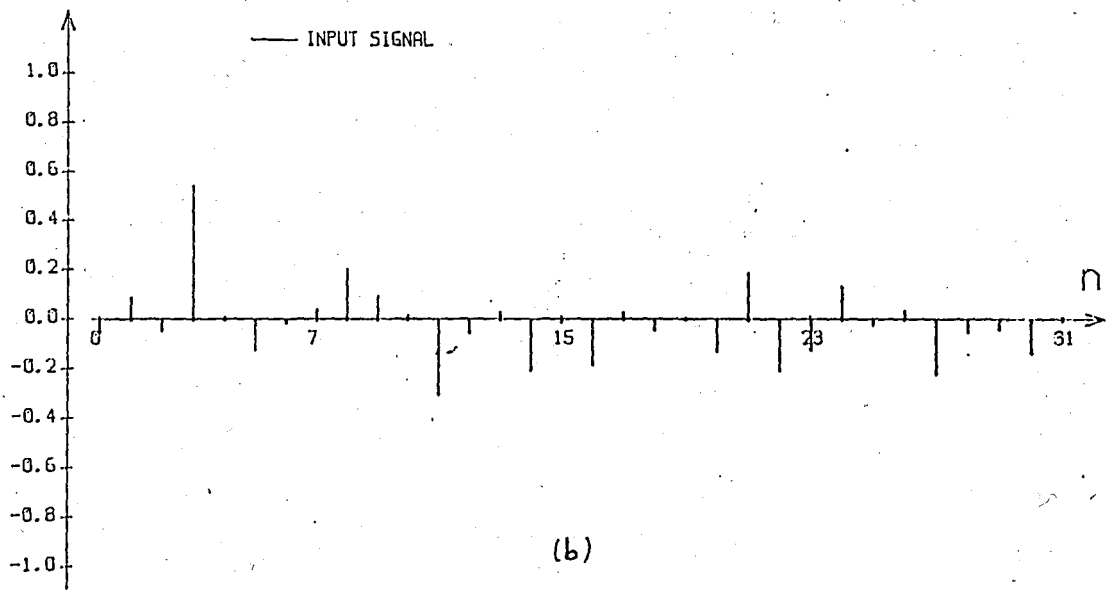


(b)

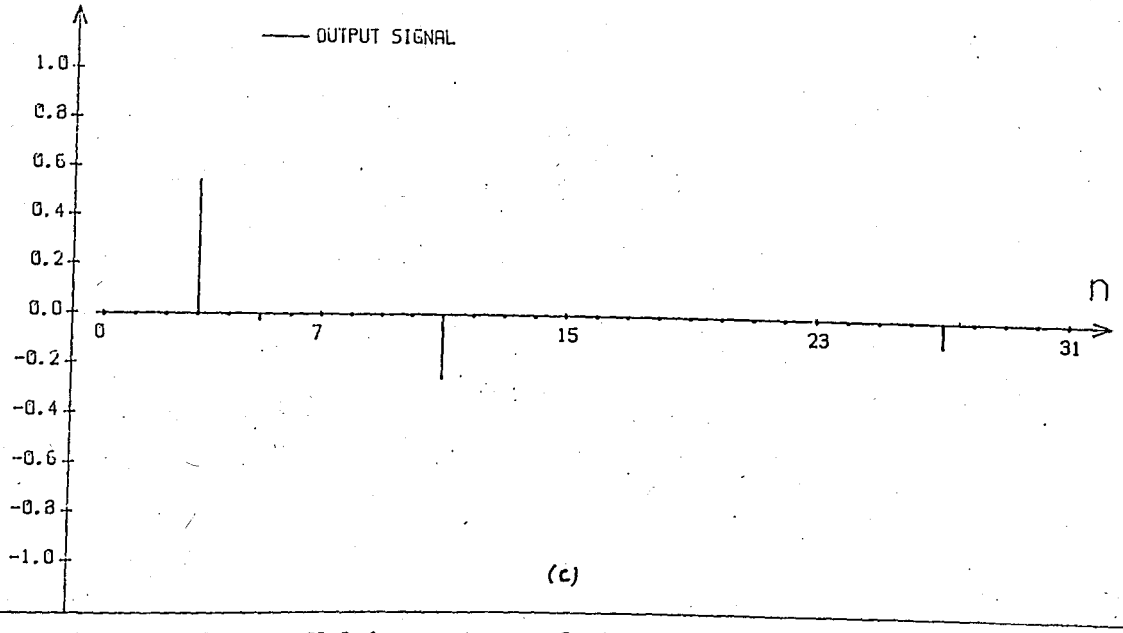
FIGURE 5.1 Transform domain adaptive filtering of a sine wave



(a)



(b)



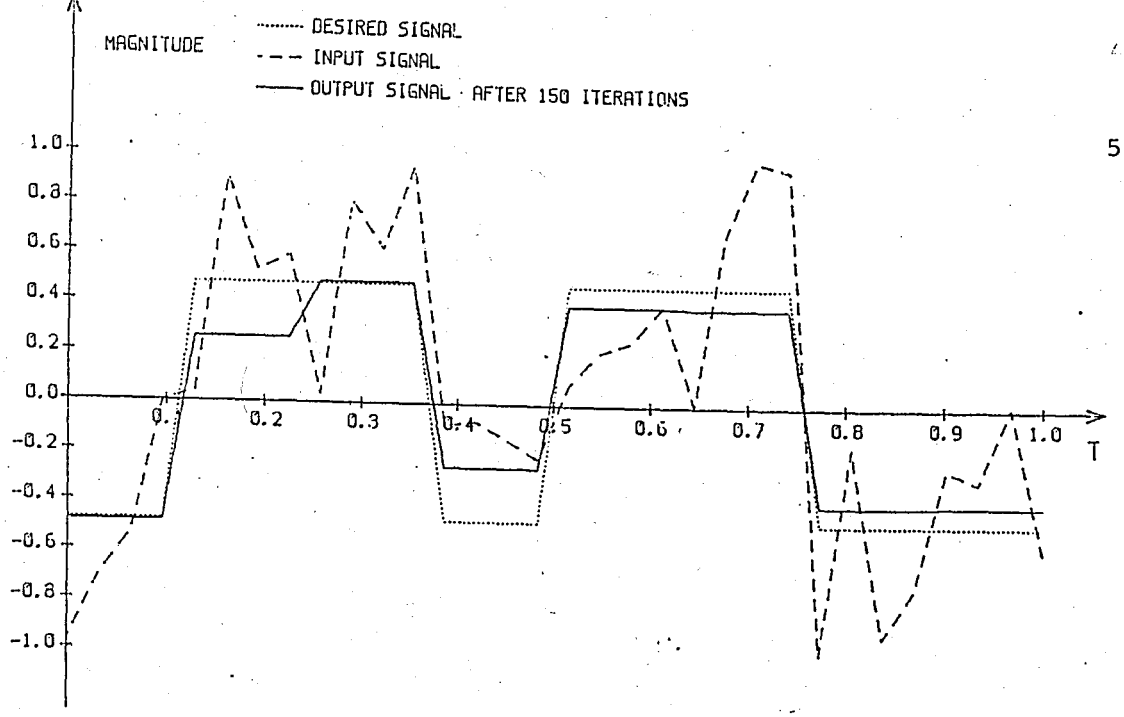
(c)

FIGURE 5.2 Walsh spectrum of the waveform shown in Fig.5.1(a)

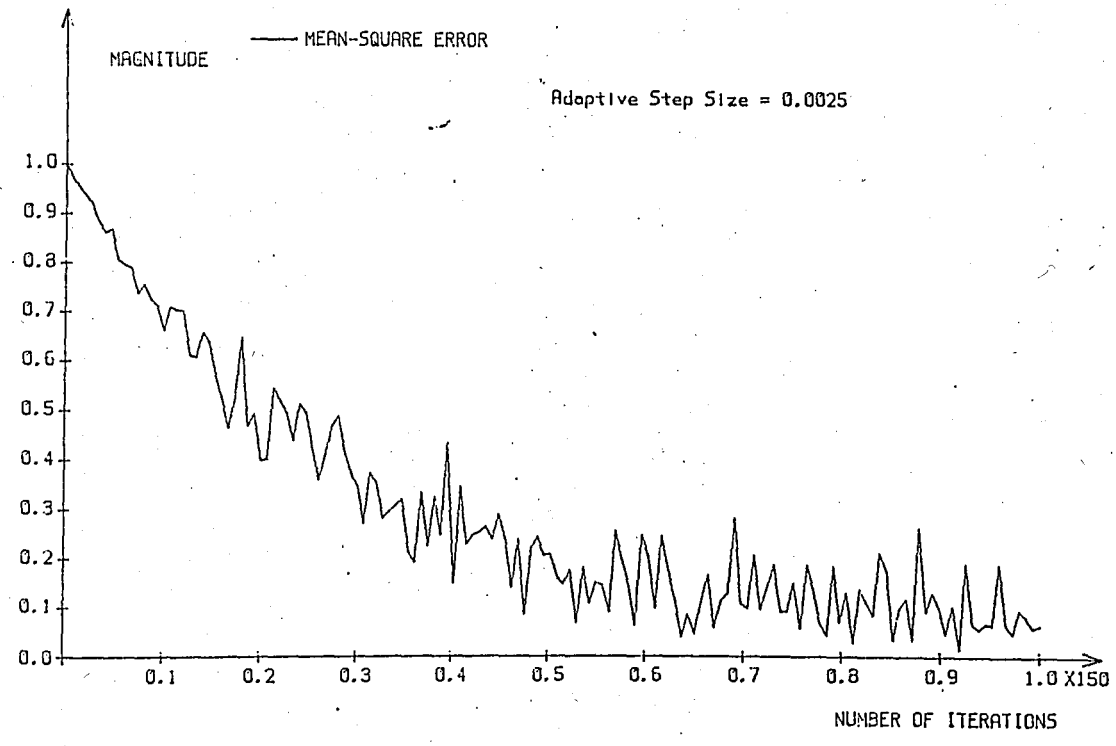
and the corresponding weight vector elements form the output signal Walsh transform coefficients. After comparing the output signal Walsh transform coefficients with the corresponding desired signal Walsh transform coefficients, the elements of an error vector are evaluated. This error vector is used to update the present weight vector elements and the first iteration process is completed. Then, the updated weight is used to compute the following output signal, and the same process goes on. The reduction in the noise amplitude is generally greater than 30 dB. After the adaptation is completed, the variation in the magnitude of the mean-square error is less than 10 dB.

5.2 ADAPTIVE FILTERING OF A RECTANGULAR WAVE

The results of the Walsh domain adaptive filtering of a rectangular wave are shown in Fig. 5.3. The input signal is a rectangular wave corrupted by white noise. The waveforms of the input, desired and, output signals are shown in Fig.5.3(a). Fig.5.3(b) shows the mean-square error versus iteration number. As it is shown in Fig.5.3(a) the effect of noise on the output signal is eliminated but there is still a difference between the input and output signal. With an adaptive step size of 0.0025, the adaptation is completed after about 70 iterations. At the end of the last iteration the reduction in the noise amplitude is greater than 30 dB. After the adaptation is completed the variation in the magnitude of the mean-square error is generally less than 10 dB. Fig.5.4 shows the Walsh spectrum of the desired, input, and output signal. As it is shown in Fig.5.4, the Walsh transform coefficients of the input signal which are not present in the desired signal Walsh



(a)



(b)

FIGURE 5.3 Transform domain adaptive filtering of a rectangular wave

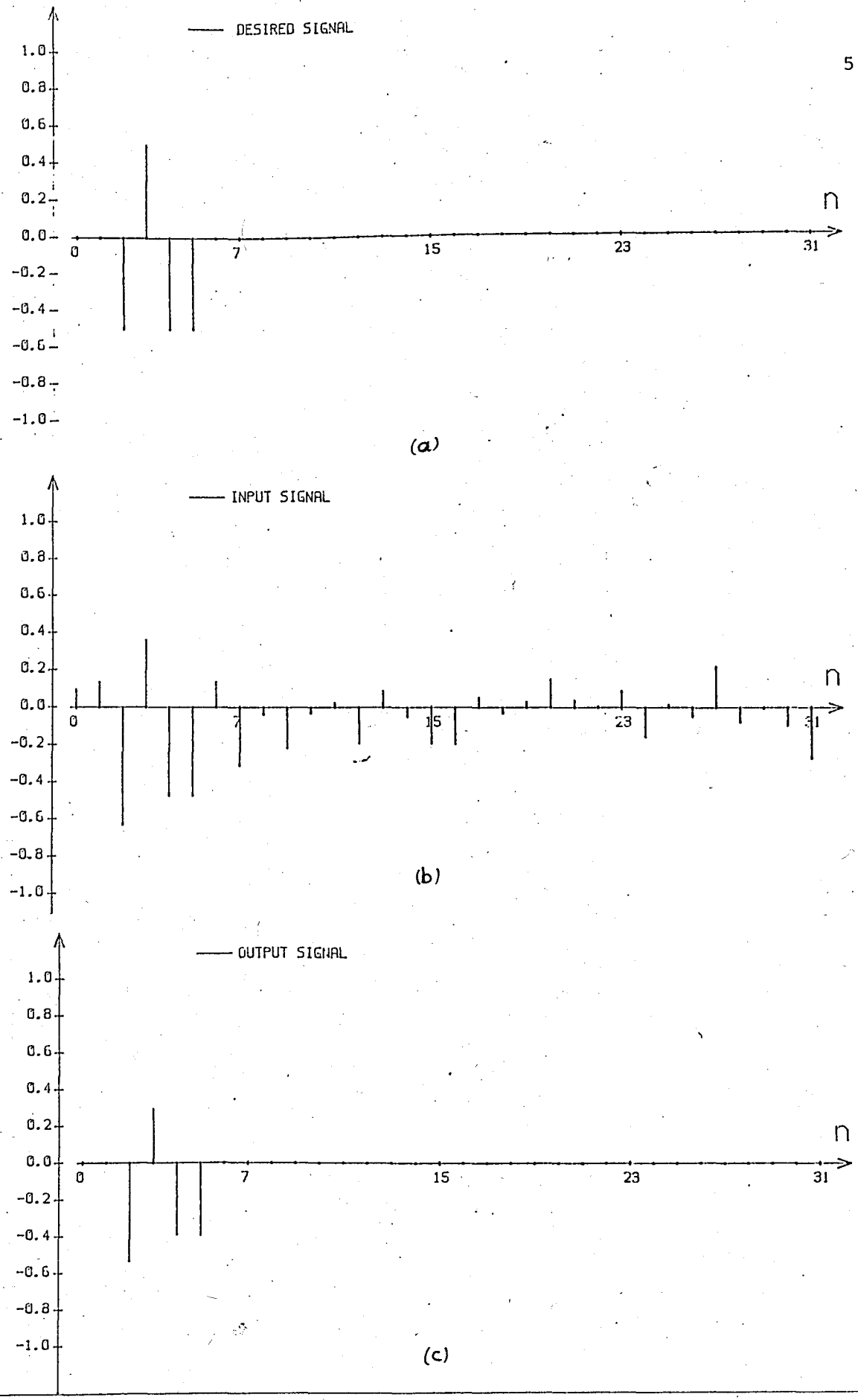
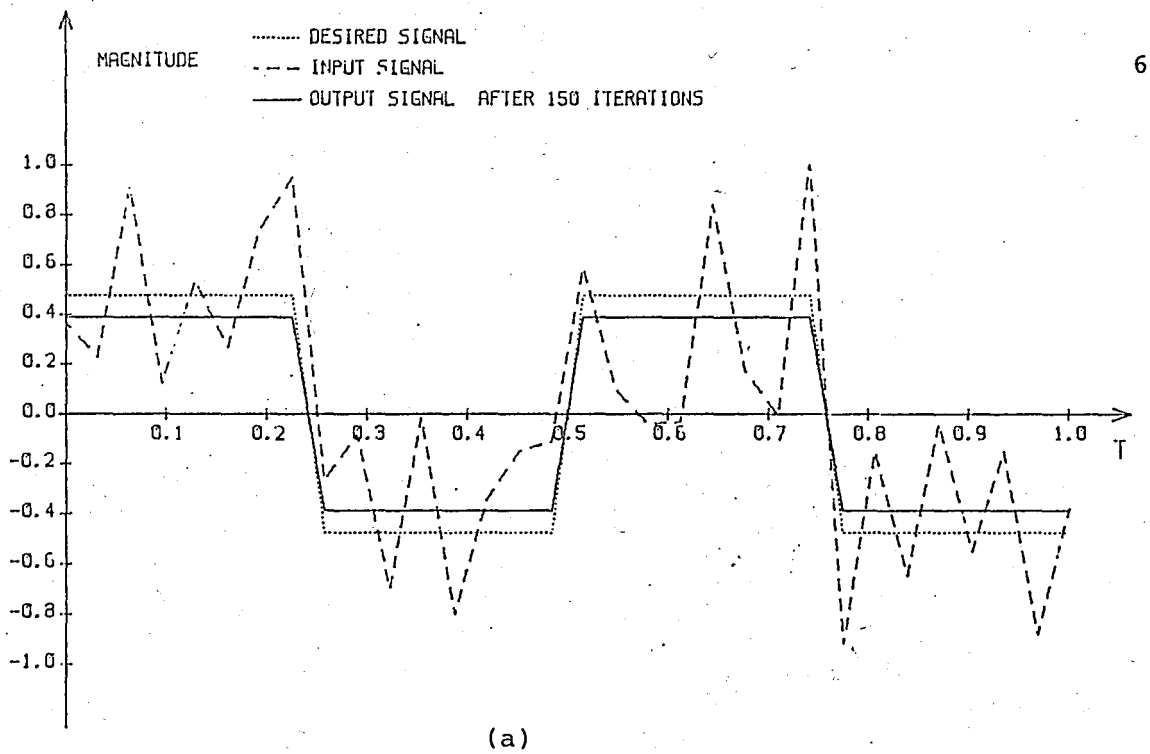


FIGURE 5.4 Walsh spectrum of the waveform shown in Fig.5.3(a)

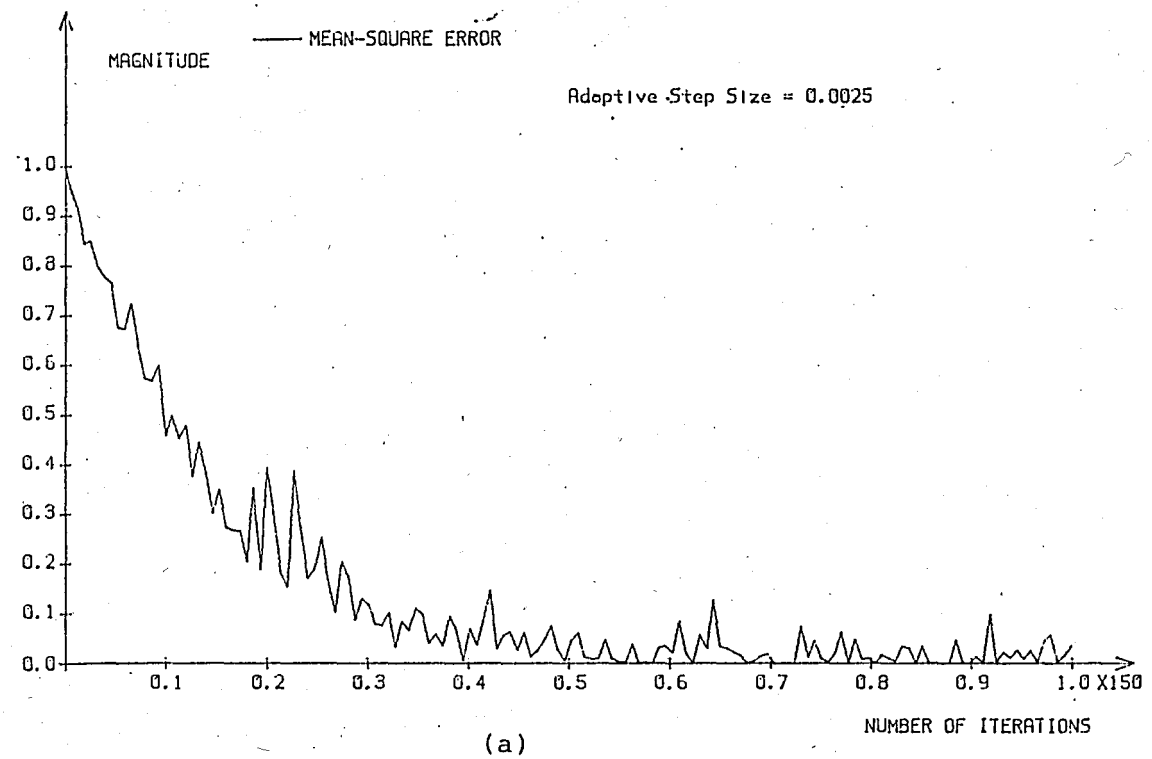
spectra are eliminated. Although they have the same index number, there is a difference between the magnitude of the corresponding desired signal and the output signal Walsh transform coefficients. Because of the differences between the magnitudes of the desired signal and the output signal Walsh transform coefficients, the output signal waveform is not same as the desired signal. Since the whole signal is represented by only a few Walsh transform coefficients, a small change in the magnitude of any Walsh transform coefficient can strongly change the shape of the output waveform. The difference between the magnitudes of the output and desired signal Walsh transform coefficients can be minimized but there will be always a small error because of the nature of the adaptation process.

5.3 ADAPTIVE FILTERING OF A SQUARE WAVE

The results of the Walsh domain adaptive filtering of a square wave are shown in Fig.5.5. The input signal is a square wave corrupted by white noise. Fig.5.5(b) shows the mean-square error versus iteration number. As shown in Fig.5.5(b) the reduction in the magnitude of the mean-squre error is significant. As it is shown in Fig.5.6, a square wave can be represented by only a single Walsh transform coefficient. Since there is only a single Walsh transform coefficient to be evaluated after the adaptation, all of the other Walsh transform coefficients should be eliminated. After the unwanted Walsh transform coefficients are eliminated, the output signal becomes noise free. With an adaptive step size of 0.0025, the adaptation process is completed after about 40 iterations. The reduction in the noise amplitude is greater than 40 dB.



(a)



(a)

FIGURE 5.5 Transform domain adaptive filtering of a square wave

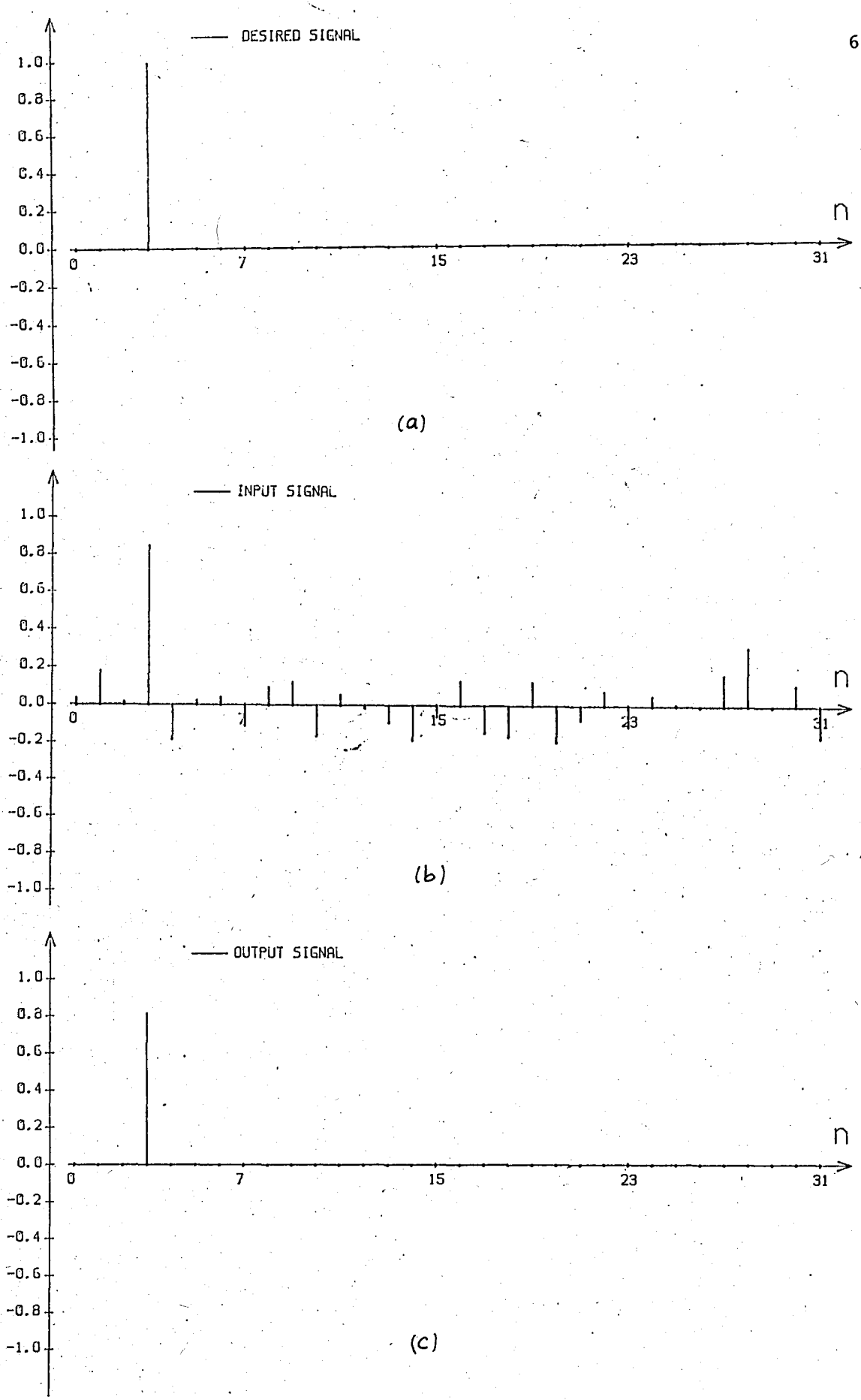


FIGURE 5.6 Walsh spectrum of the waveform shown in Fig.5.5(a)

After the adaptation is completed, the variation in the amplitude of the mean square error is less than 5 dB. The output signal is almost the same as the desired signal. There is only a small difference between the magnitudes of the desired signal and the output signal. This error is inevitable because of the nature of the adaptation process. However, the result is quite satifactory.

5.4 EFFECTS OF INITIAL WEIGHT AND ADAPTIVE STEP SIZE ON CONVERGENCE CHARACTERISTICS.

The effects of the initial weight, bias weight, and the adaptive step size on the mean-square error are shown in Figs. 5.7 and 5.8. As it is shown in Figs. 5.2, 5.4, and 5.6, the Walsh spectra of the desired signal comprises only a few Walsh transform coefficients. After setting an element of the initial weight vector equal to zero, if the corresponding Walsh transform coefficient of the desired signal is equal to zero, the related Walsh transform coefficient of the output signal will be directly evaluated without taking into account the Walsh transform coefficient of the input signal (3.30).

The choice of the adaptive step size effects both the number of iteration to complete the adaptation and the magnitude of the mean-square error. The adaptive step size is chosen emprically for optimum convergence in the plots. When the adaptive step size is chosen very close to 1 no convergence can be obtained. As the value of the adaptive step size decreases the number of iteration to complete the adaptation increases, but the final value of the mean square error also decreases.

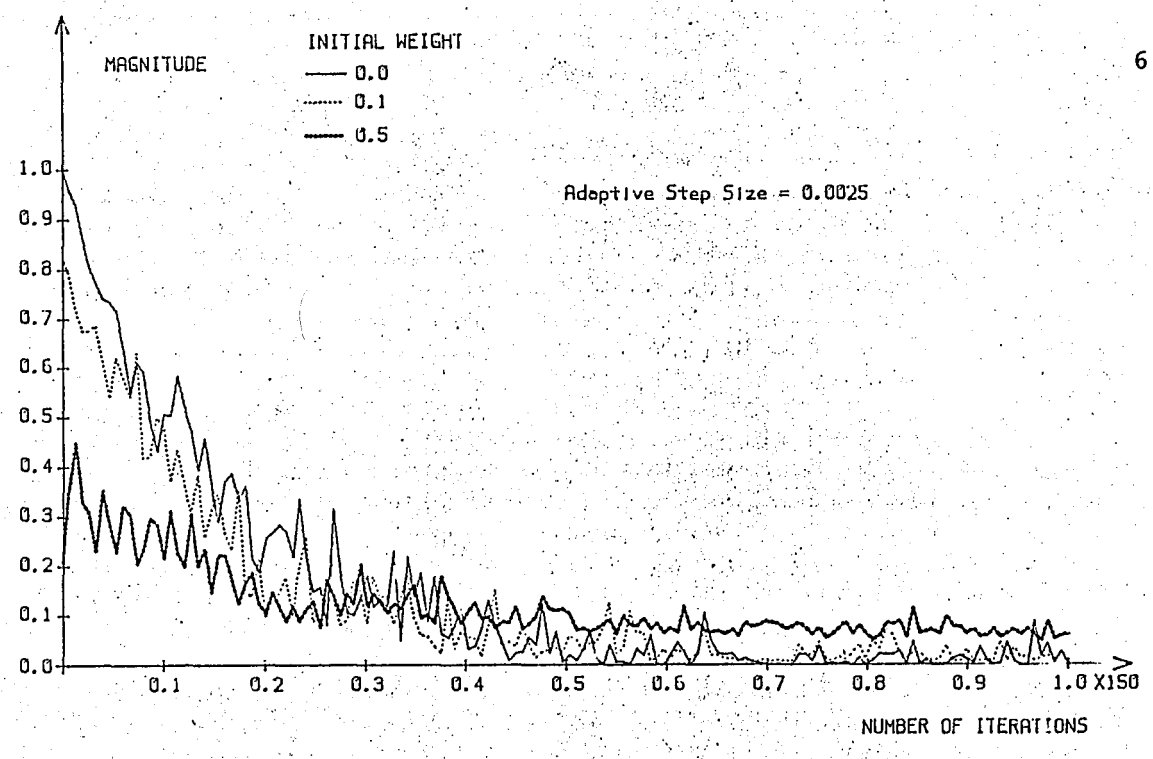


FIGURE 5.7 Convergence characteristics of the mean-square error as the initial weight varies from 0 to 0.5

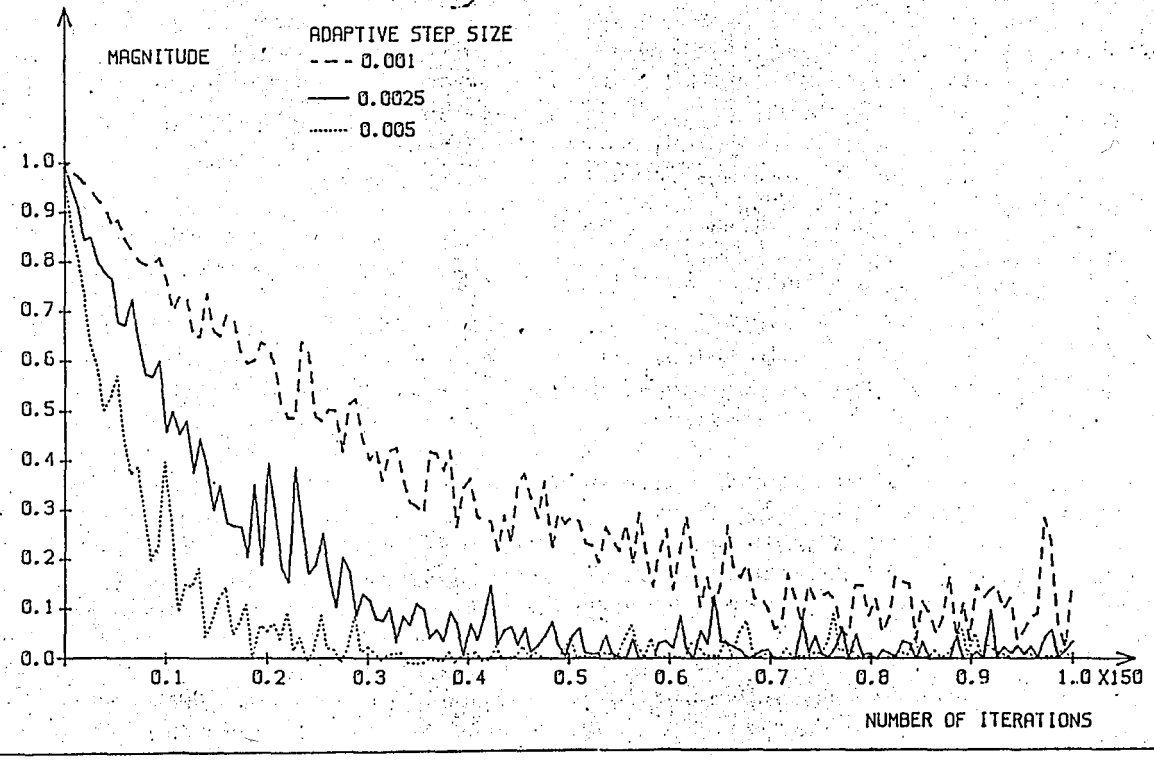


FIGURE 5.8 Convergence characteristics of the mean-square error as the adaptive step size varies from 0.001 to 0.005

5.5 CONCLUSIONS

After setting up the computer model of the Walsh domain adaptive filter, the convergence performance has been investigated through an extensive study with many types of input signals embedded in white noise. Although no rigorous theoretical proof about the convergence properties has been given, a great deal of emprical evidence illustrating the stability of the Walsh domain adaptive filter has been presented using computer simulations. It should be mentioned that the Fourier transform is favourable when we deal with the continuous waveform analysis. On the other hand, a discontinuous waveform, a rectangular waveform, is more easily reconstructed when the Walsh transform is employed. The Walsh transform and its inverse can be obtained by matrix multiplication using the digital computer. Since the matrices are symmetrical for the Walsh transform, (unlike the Fourier transform) then both the transform and its inverse are identical, except for a scaling factor, 1/N. Consequently, the use of Walsh domain brings out computational savings and easy implementation over the Fourier domain techniques.

amplitude of
The discrete Fourier transform is invariant to the phase of the input signal so that the same spectral decomposition can be obtained independently of the phase or circular time-shift of the input signal. This is not possible for the discrete Walsh transform.

Since the sine and cosine functions cannot be represented exactly by a number of bits, a source of truncation noise is introduced by the

discrete Fourier transform which involves repeated multiplication by a complex number. The Walsh transform, on the other hand, involves only addition and subtraction and precise representation is possible. This can be interpreted that the Walsh transform does not bring a truncation noise.

For the case of sinusoidal inputs embedded in white noise, it has been shown that many of the Walsh transform coefficients of the desired signal are equal to zero. This brings out rapid convergence performance when the initial weight is chosen relatively small or very close to zero.

Special interest was given to the square waves and it has been demonstrated that almost a noise free output is evaluated when the desired signal is represented by a single Walsh transform coefficient.

The effects of initial weight and adaptive step size on convergence performance were studied in detail. If one sets an element of the initial weight vector equal to zero, the corresponding output signal Walsh transform coefficient can be obtained independently for the input signal when the related Walsh transform coefficient of the desired signal is equal to zero. Smaller adaptive step size brings small variations on the magnitude of the mean square error after the adaptation is completed, but the number of iteration to complete the adaptation increases.

5.6 SUGGESTIONS FOR FURTHER STUDY

A hardware implementation of the Walsh domain adaptive filter is practical and can be performed in a further study. Since the Walsh transform depends on Modulo-2 addition, it possesses dyadic convolution. The effect of dyadic convolution on convergence performance can be investigated analytically. Further study might concentrate on the bounds of the adaptive step size which is an open problem in the Walsh domain adaptive filtering for the time being. The effect of zero intial weight on the mean-square error should be investigated and analytical results should be established if possible. The initial phase dependency of the Walsh transform for the case of sinusoidal signal can also be investigated analytically.

APPENDIX

A number of programs referred to earlier in the study are given here. The programs are all written in BASIC for the RADIO SHACK TRS-80 MODEL 16 digital computer. Program-1 can be used to rearrange a series into bit reversed order. Program-2 is written to evaluate Hadamard matrices of any order restricted having the integer power of two. Program-3 can be used to compute Hadamard ordered Walsh transform. Program-4 can be used to compute fast Walsh transform. Computer simulation of the Walsh domain adaptive filtering is presented in Program-5. The input medium is the keybord of the computer. The output medium is either a printer or a plotter. After the run command, the explanation promts apper where they are necessary. More specific information about the plotter commands can be found in the user's manual of the computer. The important variables of Program-5 are listed below.

D Desired response

DW Transformed desired response

E Error

N Number of data in one iteration

NB Bit reverse order index

NM Number of iteration

X Input signal

TR Transform key

U Dummy argument

V Dummy argument

Y Output signal

W Weight vector

```
100 REM**              PROGRAM-1                    **
110 REM** THIS PROGRAM REARRANGES A GIVEN **
120 REM** SERIES INTO BIT REVERSED ORDER  **
130 CLS
140 DEFINT I-N
150 DIM X(32),Y(32),IB(10)
160 PRINT@(10,0),
170 PRINT"ENTER THE NUMBER OF DATA IN THE SEQUENCE"
180 PRINT"TO BE REARRANGED INTO BIT REVERSED ORDER"
190 INPUT"IT SHOULD BE A POWER OF  2  ";N
200 CLS
210 FOR I=1 TO N
220 INPUT X(I)
230 NEXT I
240 PRINT"THE ORIGINAL SEQUENCE"
250 PRINT
260 FOR I=1 TO N STEP 8
270 PRINT USING"+#####.##";X(I);X(I+1);X(I+2);X(I+3);
280 IF I+4>N THEN GOTO 310
290 PRINT USING"+#####.##";X(I+4);X(I+5);X(I+6);X(I+7)
300 NEXT I
310 PRINT
320 FOR I=1 TO N 'BIT REVERSE
330 IA=I-1
340 IC=1
350 ID=IA/2
360 IBIT(IC)=1
370 IF IA=(ID*2) THEN IB(IC)=0
380 IF ID=0 THEN GOTO 420
390 IA=ID
400 IC=IC+1
410 GOTO 350
420 IE=1
430 IG=N
440 FOR I1=1 TO IC
450 IG=IG/2
460 IE=IE+IG*IB(I1)
470 NEXT I1
480 Y(IE)=X(I)
490 NEXT I
500 FOR I=1 TO N
510 X(I)=Y(I)
520 NEXT I
530 PRINT" THE BIT REVERSED SEQUENCE "
540 PRINT
550 FOR I=1 TO N STEP 8
560 PRINT USING"+#####.##";X(I);X(I+1);X(I+2);X(I+3);
570 IF I+4>N THEN GOTO 600
580 PRINT USING"+#####.##";X(I+4);X(I+5);X(I+6);X(I+7)
590 NEXT I
600 END
```

```
100 REM**                  PROGRAM-2                              **
110 REM** THIS PROGRAM EVALUATES HADAMARD MATRICES               **
120 DEFINT I-N
130 DIM A(2,2),AR(32,32),HA(32,32)
140 CLS :PRINT"ENTER THE MATRIX DIMENSION "
150 INPUT"IT SHOULD BE AN INTEGER POWER OF  2  ";N
160 A(1,1)=1:A(1,2)=1:A(2,1)=1:AR(1,1)=1
170 AR(1,2)=1:AR(2,1)=1:A(2,2)=-1:AR(2,2)=-1
180 MT=-1:IA=N
190 IA=IA/2:IF IA=0 THEN GOTO 210
200 MT=MT+1:GOTO 190
210 NA=2
220 FOR M=1 TO MT
230 NA=NA*2
240 FK=1:FL=1:II=1:JJ=1:K=1
250 FOR I=1 TO NA
260 L=1
270 FOR J=1 TO NA
280 HA(I,J)=AR(K,L)*A(II,JJ)
290 IF JJ=2 THEN GOTO 310
300 JJ=2:GOTO 320
310 JJ=1
320 FL=FL+0.5:LI=FL
330 IF LI=FL THEN L=L+1
340 NEXT J
350 IF II=2 THEN GOTO 370
360 II=2:GOTO 380
370 II=1
380 FK=FK+0.5:KI=FK
390 IF KI=FK THEN K=K+1
400 NEXT I
410 FOR I=1 TO N
420 IF M=MT THEN GOTO 450
430 FOR J=1 TO N:AR(I,J)=HA(I,J)
440 NEXT J:NEXT I:NEXT M
450 CLS :PRINT"THE HADAMARD MATRIX OF ORDER";N
460 FOR J=1 TO N'WRITE THE DESIRED MATRIX
470 FOR I=1 TO N STEP 16:PRINT
480 PRINT USING"+#####";HA(I,J);HA(I+1,J);
490 PRINT USING"+#####";HA(I+2,J);HA(I+3,J);
500 IF I+4>N THEN GOTO 600
510 PRINT USING"+#####";HA(I+4,J);HA(I+5,J);
520 PRINT USING"+#####";HA(I+6,J);HA(I+7,J);
530 IF I+8>N THEN GOTO 600
540 PRINT USING"+#####";HA(I+8,J);HA(I+9,J);
550 PRINT USING"+#####";HA(I+10,J);HA(I+11,J);
560 IF I+12>N THEN GOTO 600
570 PRINT USING"+#####";HA(I+12,J);HA(I+13,J);
580 PRINT USING"+#####";HA(I+14,J);HA(I+15,J);
590 NEXT I
600 PRINT:NEXT J
610 END
```

```
100 REM**                    PROGRAM-3                          **
110 REM** THIS PROGRAM COMPUTES THE WALSH TRANSFORM **
120 REM** OF A GIVEN SERIES. TO COMPUTE THE INVERSE **
130 REM** WALSH TRANSFORM TR MUST BE SET TO -1       **
140 DEFINT I-N
150 DIM A(2,2),AR(32,32),HA(32,32),X(32),Y(32)
160 CLS:INPUT"ENTER TR";TR
170 INPUT"ENTER THE NUMBER OF DATA";N
180 FOR I=1 TO N:INPUT X(I):NEXT I
190 GOSUB 340
200 FOR J=1 TO N:Y(J)=0
210 Y(J)=0:FOR I=1 TO N
220 Y(J)=Y(J)+X(I)*HA(I,J)
230 NEXT I:NEXT J
240 IF TR=-1 THEN GOTO 270
250 FOR I=1 TO N:X(I)=Y(I)/N:NEXT I
260 GOTO 280
270 FOR I=1 TO N:X(I)=Y(I):NEXT I
280 FOR I=1 TO N STEP 8
290 PRINTUSING"+###.#####";X(I);X(I+1);X(I+2);X(I+3);
300 IF I+4>N THEN GOTO 330
310 PRINTUSING"+###.#####";X(I+4);X(I+5);X(I+6);X(I+7)
320 NEXT I
330 END
340 A(1,1)=1:A(1,2)=1:A(2,1)=1:AR(1,1)=1
350 AR(1,2)=1:AR(2,1)=1:A(2,2)=-1:AR(2,2)=-1
360 MT=-1:IA=N
370 IA=IA/2:IF IA=0 THEN GOTO 390
380 MT=MT+1:GOTO 370
390 NA=2
400 FOR M=1 TO MT
410 NA=NA*2
420 FK=1:FL=1:II=1:JJ=1:K=1
430 FOR I=1 TO NA:L=1
440 FOR J=1 TO NA
450 HA(I,J)=AR(K,L)*A(II,JJ)
460 IF JJ=2 THEN GOTO 480
470 JJ=2:GOTO 490
480 JJ=1
490 FL=FL+0.5:LI=FL
500 IF LI=FL THEN L=L+1
510 NEXT J
520 IF II=2 THEN GOTO 540
530 II=2:GOTO 550
540 II=1
550 FK=FK+0.5:KI=FK
560 IF KI=FK THEN K=K+1
570 NEXT I
580 FOR I=1 TO N
590 IF M=MT THEN RETURN
600 FOR J=1 TO N:AR(I,J)=HA(I,J)
610 NEXT J:NEXT I:NEXT M
620 RETURN
```

```
100 REM**                  PROGRAM-4                          **
110 REM** THIS PROGRAM COMPUTES FAST WALSH                    **
120 REM** TRANSFORM OF A GIVEN SERIES. TO COMPUTE            **
130 REM** THE INVERSE WALSH TRANSFORM SET TR=-1              **
140 CLS:DEFINT I-N
150 DIM U(64),V(64),IB(20)
160 INPUT"ENTER -1 TO GET INVERSE TRANSFORM";TR
170 INPUT"ENTER THE NUMBER OF DATA ";N
180 CLS:FOR I=1 TO N:INPUT U(I):NEXT I
190 IT=0:IJ=N 'TAKE LOGARITHM AS BASE 2
200 IJ=INT(IJ/2)
210 IF IJ=0 THEN GOTO 230
220 IT=IT+1:GOTO 200
230 FOR I=1 TO N'BIT REVERSE
240 IA=I-1:IC=1
250 ID=INT(IA/2)
260 IB(IC)=1
270 IF IA=(ID*2) THEN IB(IC)=0
280 IF ID=0 THEN GOTO 310
290 IA=ID:IC=IC+1
300 GOTO 250
310 IE=1:IG=N
320 FOR I1=1 TO IC
330 IG=INT(IG/2)
340 IE=IE+IG*IB(I1)
350 NEXT I1
360 V(IE)=U(I)
370 NEXT I
380 FOR I=1 TO N:U(I)=V(I):NEXT I
390 FOR J1=1 TO IT 'COMPUTE TRANSFORM
400 IF J1=1 THEN JA=1
410 IF J1<>1 THEN JA=JA*2
420 JB=INT(N/JA)
430 JC=INT(JB/2)
440 AL=1
450 FOR J2=1 TO JA
460 JE=(J2-1)*JB
470 FOR J3=1 TO JC
480 JD=JE+J3+JC:JG=JE+J3
490 V(JG)=U(JG)+AL*U(JD)
500 V(JD)=U(JG)-AL*U(JD)
510 NEXT J3
520 AL=-AL:NEXT J2
530 FOR I=1 TO N:U(I)=V(I):NEXT I
540 NEXT J1
550 IF TR=-1 THEN GOTO 570
560 FOR I=1 TO N:U(I)=U(I)/N:NEXT I
570 PRINT"THE TRANSFORMED SEQUENCE "
580 FOR I=1 TO N STEP 8:PRINT
590 PRINT USING"+###.####";U(I);U(I+1);U(I+2);U(I+3);
600 PRINT USING"+###.####";U(I+4);U(I+5);U(I+6);U(I+7)
610 NEXT I
620 END
```

```
1000 CLS2:REM****   PROGRAM 5   ***
1010 PRINT"TO GET A PLOTTER OUTPUT"
1020 INPUT"ENTER  Y  OTHERWISE  N  ";PL$
1030 IF PL$="N" THEN GOTO 1090
1040 PRINT"PRESS ENTER OR RETYPE LINE 4670 AND 5440 "
1050 PRINT"TO THE NUMBER OF ITERATION WILL BE DONE"
1060 INPUT*
1070 PR$="N"
1080 GOTO 1100
1090 PR$="Y"
1100 CLS:DEFINT I-N
1110 DIM X(64),Y(64),D(64),DW(64),U(801),V(801)
1120 DIM E(801),W(64,2),NB(64),IB(20)
1130 INPUT"ENTER THE NUMBER OF ITERATION ";NM
1150 PRINT
1160 PRINT"ENTER THE NUMBER OF DATA "
1170 PRINT"IN ONE ITERATION"
1180 INPUT"  64   32   16   8  ";N
1190 CLS:PRINT@(5,0),
1200 INPUT"ENTER ADAPTIVE STEP SIZE  ";AR
1210 PRINT
1220 PRINT"FOR VARIABLE ADAPTIVE STEP SIZE"
1230 INPUT"ENTER  1  OTHERWISE  0  ";DS
1240 PRINT
1250 INPUT"ENTER INITIAL WEIGHT  ";WW
1260 PRINT
1280 GOSUB 2610
1290 PRINT@(18,0),
1300 PRINT"DESIRED SIGNAL EVALUATION ";
1305 K=N/8
1310 FOR I=1 TO N:D(I)=1:NEXT
1312 FOR I=12 TO N
1314 X(I)=-1
1316 NEXT I
1320 FOR I=12 TO N STEP 6
1330 D(I)=D(I)*X(I)
1332 D(I+1)=D(I+1)*X(I)
1333 D(I+2)=D(I+2)*X(I)
1340 NEXT
1390 PRINT"     WALSH TRANSFORM COMPUTING (D) ";
1400 TR=1
1410 FOR I=1 TO N :U(I)=D(I) :NEXT I
1420 GOSUB 2960
1430 FOR I=1 TO N :DW(I)=U(I) :NEXT I
1440 CLS
1450 FOR J=1 TO NM
1460 CLS1
1470 PRINT@(17,0),
1480 PRINT"ITERATION NUMBER = ";J
1490 PRINT"DESIRED SIGNAL";
1500 GOSUB 6080
1510 PRINT
1520 PRINT"INPUT SIGNAL EVALUATION";
1530 REM ADD SOME NOISE
1540 DN=0
1550 FOR I=1 TO N
```

```
1560 U(I)=RND(17/23)
1570 IF DN<U(I) THEN DN=U(I)
1580 NEXT I
1590 FOR I=1 TO N :U(I)=U(I)/DN :NEXT I
1600 FOR I=1 TO N :U(I)=U(I)-.5 :NEXT I
1610 FOR I= 1 TO N
1620 A3=U(I)*1.0
1630 A3=INT(A3*A0)/A0
1640 X(I)=D(I)+A3
1650 NEXT I
1660 GOSUB 6170
1670 TR=1
1680 PRINT"    WALSH TRANSFORM COMPUTING (X)";
1690 FOR I=1 TO N :U(I)=X(I) :NEXT I
1700 GOSUB 2960
1710 FOR I=1 TO N :X(I)=U(I) :NEXT I
1720 PRINT
1730 PRINT"ADAPTIVE FILTERING";
1740 IF DS<>1 THEN GOTO 1800
1750 FOR I=1 TO N
1760 S1=X(I)*X(I)
1770 U(I)=5*AR/SQR(S1)
1780 NEXT I
1790 GOTO 1810
1800 FOR I=1 TO N :U(I)=2*AR :NEXT I
1810 FOR I=1 TO N
1830 Y(I)=W(I,1)*X(I)
1840 R=DW(I)-Y(I)
1850 W(I,2)=W(I,1)+U(I)*X(I)*R
1860 NEXT I
1870 FOR I=1 TO N:W(I,1)=W(I,2):NEXT I
1880 PRINT"    ";:FOR LL=1 TO 1000:NEXT LL:PRINT
1890 PRINT"INVERSE WALSH TRANSFORM COMPUTING (Y)";
1900 TR=-1
1910 FOR I=1 TO N :U(I)=Y(I) :NEXT I
1920 GOSUB 2960
1930 FOR I=1 TO N :Y(I)=U(I) :NEXT I
1940 GOSUB 6250
1950 PRINT
1960 PRINT"ERROR COMPUTING";
1970 E(J)=0
1980 FOR I=1 TO N
1990 ER=D(I)-Y(I)
2000 E(J)=E(J)+ER*ER
2010 NEXT I
2020 E(J)=E(J)/N
2030 PRINT"                          M.S.E.= ";E(J);
2040 NEXT J
2050 PRINT
2060 PRINT"INVERSE WALSH TRANSFORM COMPUTING (X)";
2070 FOR I=1 TO N :U(I)=X(I) :NEXT I
2080 GOSUB 2960
2090 FOR I=1 TO N :X(I)=U(I) :NEXT I
2100 IF PR$="Y" THEN GOTO 2280
2110 DE=0 :DD=0 :DS=0 :DY=0
```

```
2120 FOR I=1 TO N
2130 IF DD<ABS(D(I)) THEN DD=ABS(D(I))
2140 IF DS<ABS(X(I)) THEN DS=ABS(X(I))
2150 IF DY<ABS(Y(I)) THEN DY=ABS(Y(I))
2160 NEXT I
2170 IF DD<DS THEN DD=DS
2180 IF DD<DY THEN DD=DY
2190 FOR I=1 TO N
2200 D(I)=D(I)/DD
2210 X(I)=X(I)/DD
2220 Y(I)=Y(I)/DD
2230 NEXT I
2240 FOR I=1 TO NM
2250 IF DE<E(I) THEN DE=E(I)
2260 NEXT I
2270 FOR I=1 TO NM:E(I)=E(I)/DE:NEXT I
2280 CLS2
2290 PRINT@(23,0)," "
2300 PRINT"TO DISPLAY THE DESIRED SIGNAL"
2310 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
2320 IF AG$="Y" THEN GOSUB 3270
2330 CLS
2340 PRINT@(23,0)," "
2350 PRINT"TO DISPLAY THE INPUT SIGNAL"
2360 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
2370 IF AG$="Y" THEN GOSUB 3610
2380 CLS
2390 PRINT@(23,0)," "
2400 PRINT"TO DISPLAY THE OUTPUT SIGNAL"
2410 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
2420 IF AG$="Y" THEN GOSUB 3950
2430 CLS
2440 PRINT@(23,0)," "
2450 PRINT"TO DISPLAY THE RESULT AGAIN "
2460 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
2470 IF AG$="Y" THEN GOTO 2280
2480 CLS
2490 PRINT@(22,0)," "
2500 PRINT"TO DISPLAY THE ERROR"
2510 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
2520 IF AG$="Y" THEN GOSUB 5560
2530 CLS
2540 PRINT@(10,0)," "
2550 PRINT"TO STOP THE PROGRAM EXECUTION"
2560 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
2570 IF AG$<>"Y" THEN GOTO 2280
2580 CLS2
2590 END
2600 REM *****SUBROUTINE****
2610 CLS:PRINT@(14,0),
2620 PRINT"BIT REVERSE SEQUENCE ";
2630 IT=0 'TAKE LOGARITHM AS BASE 2
2640 IJ=N
2650 IJ=IJ/2
2660 IF IJ=0 THEN GOTO 2690
2670 IT=IT+1
```

```
2680 GOTO 2650
2690 FOR I=1 TO N 'BIT REVERSE
2700 IA=I-1
2710 IC=1
2720 ID=IA/2
2730 IB(IC)=1
2740 IF IA=(ID*2) THEN IB(IC)=0
2750 IF ID=0 THEN GOTO 2790
2760 IA=ID
2770 IC=IC+1
2780 GOTO 2720
2790 IE=1
2800 IG=N
2810 FOR I1=1 TO IC
2820 IG=IG/2
2830 IE=IE+IG*IB(I1)
2840 NEXT I1
2850 NB(I)=IE
2860 NEXT I
2870 FOR I=1 TO N:W(I,1)=WW:NEXT I
2880 RANDOM
2890 A0=10000
2900 A1=(360*.01745329)/N
2910 NP=INT(640/(N-1))
2920 NL=INT(640/(NM-1))
2930 NS=INT(1600/(N-1))
2940 NE=INT(1600/(NM-1))
2950 RETURN
2960 REM WALSH TRANSFORM
2970 FOR I=1 TO N 'BIT REVERSE
2980 V(NB(I))=U(I)
2990 NEXT I
3000 FOR I=1 TO N :U(I)=V(I) :NEXT I
3010 FOR J1=1 TO IT 'COMPUTE TRANSFORM
3020 IF J1=1 THEN JA=1
3030 IF J1<>1 THEN JA=JA*2
3040 JB=INT(N/JA)
3050 JC=INT(JB/2)
3060 AL=1
3070 FOR J2=1 TO JA
3080 JE=(J2-1)*JB
3090 FOR J3=1 TO JC
3100 JD=JE+J3+JC
3110 JG=JE+J3
3120 V(JG)=U(JG)+AL*U(JD)
3130 V(JD)=U(JG)-AL*U(JD)
3140 NEXT J3
3150 AL=-AL
3160 NEXT J2
3170 FOR I=1 TO N
3180 U(I)=V(I)
3190 NEXT I
3200 NEXT J1
3210 IF TR=-1 THEN RETURN
3220 RR=1.0/N
3230 FOR I=1 TO N
```

```
3240 U(I)=U(I)*RR
3250 NEXT I
3260 RETURN
3270 REM DISPLAY D
3280 IF PR$="Y" THEN GOTO 3430
3290 CLS:GOSUB 6090
3300 PRINT@(0,50),"SCALE FACTOR = ";DD
3310 PRINT@(22,0),"TO SCALE THE PAPER "
3320 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
3330 CLS
3340 IF AG$="Y" THEN GOSUB 4290
3350 PRINT@(23,0),"TO PLOT THE DESIRED SIGNAL"
3360 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
3370 CLS
3380 IF AG$="Y" THEN GOSUB 4970
3390 PRINT@(23,0),"TO CLEAR THE SCREEN"
3400 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
3410 IF AG$="Y" THEN CLS2
3420 GOTO 3440
3430 GOSUB 3450
3440 CLS:RETURN
3450 REM PRINT D
3460 PRINT
3470 PRINT"   THE DESIRED SIGNAL"
3480 PRINT
3490 FOR I=1 TO N STEP 8
3500 PRINT USING"+###.#####";D(I);D(I+1);D(I+2);D(I+3);
3510 PRINT USING"+###.#####";D(I+4);D(I+5);D(I+6);D(I+7)
3520 NEXT I
3530 LPRINT
3540 LPRINT"   THE DESIRED SIGNAL"
3550 LPRINT
3560 FOR I=1 TO N STEP 8
3570 LPRINT USING"+###.#####";D(I);D(I+1);D(I+2);D(I+3);
3580 LPRINT USING"+###.#####";D(I+4);D(I+5);D(I+6);D(I+7)
3590 NEXT I
3600 RETURN
3610 REM DISPLAY X
3620 IF PR$="Y" THEN GOTO 3770
3630 CLS:GOSUB 6170
3640 PRINT@(0,50),"SCALE FACTOR = ";DD
3650 PRINT@(22,0),"TO SCALE THE PAPER"
3660 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
3670 CLS
3680 IF AG$="Y" THEN GOSUB 4290
3690 PRINT@(23,0),"TO PLOT THE INPUT SIGNAL"
3700 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
3710 CLS
3720 IF AG$="Y" THEN GOSUB 5170
3730 PRINT@(23,0),"TO CLEAR THE SCREEN"
3740 INPUT"    ENTER  Y  OTHERWISE  N  ";CS$
3750 IF CS$="Y" THEN CLS2
3760 GOTO 3780
3770 GOSUB 3790
3780 CLS:RETURN
3790 REM PRINT X
```

```
3800 PRINT
3810 PRINT"    THE INPUT SIGNAL"
3820 PRINT
3830 FOR I=1 TO N STEP 8
3840 PRINT USING"+###.####";X(I);X(I+1);X(I+2);X(I+3);
3850 PRINT USING"+###.####";X(I+4);X(I+5);X(I+6);X(I+7)
3860 NEXT I
3870 LPRINT
3880 LPRINT"    THE INPUT SIGNAL"
3890 LPRINT
3900 FOR I=1 TO N STEP 8
3910 LPRINT USING"+###.####";X(I);X(I+1);X(I+2);X(I+3);
3920 LPRINT USING"+###.####";X(I+4);X(I+5);X(I+6);X(I+7)
3930 NEXT I
3940 RETURN
3950 REM DISPLAY Y
3960 IF PR$="Y" THEN GOTO 4110
3970 CLS :GOSUB 6250
3980 PRINT@(0,50),"  SCALE FACTOR = ";DD
3990 PRINT@(22,0),"TO SCALE THE PAPER"
4000 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
4010 CLS
4020 IF AG$="Y" THEN GOSUB 4290
4030 PRINT@(23,0),"TO PLOT THE OUTPUT SIGNAL"
4040 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
4050 CLS
4060 IF AG$="Y" THEN GOSUB 5360
4070 PRINT@(23,0)," TO CLEAR THE SCREEN"
4080 INPUT" ENTER  Y  OTHERWISE  N  ";AG$
4090 IF AG$="Y" THEN CLS2
4100 GOTO 4120
4110 GOSUB 4130
4120 CLS:RETURN
4130 REM PRINT Y
4140 PRINT
4150 PRINT"    THE OUTPUT SIGNAL"
4160 PRINT
4170 FOR I=1 TO N STEP 8
4180 PRINT USING"+###.####";Y(I);Y(I+1);Y(I+2);Y(I+3);
4190 PRINT USING"+###.####";Y(I+4);Y(I+5);Y(I+6);Y(I+7)
4200 NEXT I
4210 LPRINT
4220 LPRINT"    THE OUTPUT SIGNAL"
4230 LPRINT
4240 FOR I=1 TO N STEP 8
4250 LPRINT USING"+###.####";Y(I);Y(I+1);Y(I+2);Y(I+3);
4260 LPRINT USING"+###.####";Y(I+4);Y(I+5);Y(I+6);Y(I+7)
4270 NEXT I
4280 RETURN
4290 REM SCALE PAPER
4300 LPRINT" ;: P6 HA 150,50 D 150,1200 "
4310 LPRINT" 140,1170 150,1200 160,1170 U "
4320 LPRINT" 140,550 D 1850,550 1820,560 "
4330 LPRINT" 1849,550 1820,540 U "
4340 LPRINT" 142,950 D 158,950 U 90,945 S11 1.0"CHR$(95)
4350 LPRINT" 142,870 D 158,870 U 90,865 S11 0.8"CHR$(95)
```

```
4360 LPRINT" 142,790 D 158,790 U 90,785 S11 0.6"CHR$(95)
4370 LPRINT" 142,710 D 158,710 U 90,705 S11 0.4 "CHR$(95)
4380 LPRINT" 142,630 D 158,630 U 90,625 S11 0.2"CHR$(95)
4390 LPRINT" 142,550 D 158,550 U 90,545 S11 0.0 "CHR$(95)
4400 LPRINT" 142,470 D 158,470 U 80,465 S11 -0.2"CHR$(95)
4410 LPRINT" 142,390 D 158,390 U 80,385 S11 -0.4"CHR$(95)
4420 LPRINT" 142,310 D 158,310 U 80,305 S11 -0.6"CHR$(95)
4430 FOR LL=1 TO 20000:NEXT LL
4440 LPRINT" 142,230 D 158,230 U 80,225 S11 -0.8"CHR$(95)
4450 LPRINT" 142,150 D 158,150 U 80,145 S11 -1.0"CHR$(95)
4460 LPRINT" 310,558 D 310,542 U 295,520 S11 0.1"CHR$(95)
4470 LPRINT" 470,558 D 470,542 U 455,520 S11 0.2"CHR$(95)
4480 LPRINT" 630,558 D 630,542 U 615,520 S11 0.3"CHR$(95)
4490 LPRINT" 790,558 D 790,542 U 775,520 S11 0.4"CHR$(95)
4500 LPRINT" 950,558 D 950,542 U 935,520 S11 0.5"CHR$(95)
4510 LPRINT" 1110,558 D 1110,542 U 1095,520 S11 0.6"CHR$(95)
4520 LPRINT" 1270,558 D 1270,542 U 1255,520 S11 0.7"CHR$(95)
4530 LPRINT" 1430,558 D 1430,542 U 1415,520 S11 0.8"CHR$(95)
4540 LPRINT" 1590,558 D 1590,542 U 1575,520 S11 0.9"CHR$(95)
4550 LPRINT" 1750,558 D 1750,542 U 1735,520 S11 1.0"CHR$(95)
4560 LPRINT" 1820,500 S12 T"CHR$(95)
4570 LPRINT" 200,1110 S11 MAGNITUDE"CHR$(95)
4580 LPRINT" P0 HA 300,50 Z "
4590 PRINT@(23,0)," "
4600 INPUT" AFTER THE PLOTTER STOPPED PRESS ENTER";
4610 CLS:RETURN
4620 REM E SCALE
4630 LPRINT" ;: P6 HA 150,140 D 150,1200 "
4640 LPRINT" 140,1170 150,1200 160,1170 U "
4650 LPRINT" 140,150 D 1850,150 1820,160 "
4660 LPRINT" 1850,150 1820,140 U"
4670 LPRINT" 1790,120 S11 X 161"CHR$(95)
4680 LPRINT" 1520,40 S11 NUMBER OF"CHR$(94)
4690 LPRINT" ITERATIONS"CHR$(95)
4700 LPRINT" 200,1110 S11  MAGNITUDE "CHR$(95)
4710 LPRINT" 142,950 D 158,950 U 90,945 S11 1.0"CHR$(95)
4720 LPRINT" 158,870 D 142,870 U 90,865 S11 0.9"CHR$(95)
4730 LPRINT" 158,790 D 142,790 U 90,785 S11 0.8"CHR$(95)
4740 LPRINT" 158,710 D 142,710 U 90,705 S11 0.7"CHR$(95)
4750 LPRINT" 158,630 D 142,630 U 90,625 S11 0.6"CHR$(95)
4760 LPRINT" 158,550 D 142,550 U 90,545 S11 0.5"CHR$(95)
4770 LPRINT" 158,470 D 142,470 U 90,465 S11 0.4"CHR$(95)
4780 LPRINT" 158,390 D 142,390 U 90,385 S11 0.3"CHR$(95)
4790 LPRINT" 158,310 D 142,310 U 90,305 S11 0.2"CHR$(95)
4800 FOR LL=1 TO 20000:NEXT LL
4810 LPRINT" 158,230 D 142,230 U 90,225 S11 0.1"CHR$(95)
4820 LPRINT" 158,150 D 142,150 U 90,141 S11 0.0"CHR$(95)
4830 LPRINT" 310,158 D 310,142 U 295,120 S11 0.1"CHR$(95)
4840 LPRINT" 470,158 D 470,142 U 455,120 S11 0.2"CHR$(95)
4850 LPRINT" 630,158 D 630,142 U 615,120 S11 0.3"CHR$(95)
4860 LPRINT" 790,158 D 790,142 U 775,120 S11 0.4"CHR$(95)
4870 LPRINT" 950,158 D 950,142 U 935,120 S11 0.5"CHR$(95)
4880 LPRINT" 1110,158 D 1110,142 U 1095,120 S11 0.6"CHR$(95)
4890 LPRINT" 1270,158 D 1270,142 U 1255,120 S11 0.7"CHR$(95)
4900 LPRINT" 1430,158 D 1430,142 U 1415,120 S11 0.8"CHR$(95)
4910 LPRINT" 1590,158 D 1590,142 U 1575,120 S11 0.9"CHR$(95)
```

```
4920 LPRINT" 1750,158 D 1750,142 U 1735,120 S11 1.0"CHR$(95)
4930 LPRINT" P0 HA 300,50 Z"
4940 PRINT@(23,0),
4950 INPUT"AFTER THE PLOTTER STOPPED PRESS ENTER";
4960 CLS:RETURN
4970 REM D PLOT
4980 FOR I=1 TO N :V(I)=D(I) :NEXT I
4990 FOR I=1 TO N
5000 V(I)=550+INT(400*V(I))
5010 U(I)=150+INT(NS*(I-1))
5020 NEXT I
5030 LPRINT" ;: P2 HA "
5040 LPRINT" 450,1155 D 505,1155 U 518,1147 "
5050 LPRINT" S11 DESIRED SIGNAL "CHR$(95)
5060 FOR L=1 TO 10000:NEXT L
5070 LPRINT U(1);",";V(1)
5080 LPRINT" D "
5090 FOR I=2 TO N
5100 FOR L=1 TO 40 :NEXT L
5110 LPRINT U(I);",";V(I)
5120 NEXT I
5130 LPRINT" P0 HA 300,50 Z "
5140 PRINT@(23,0)," "
5150 INPUT" AFTER THE PLOTTER STOPPED PRESS ENTER";
5160 CLS:RETURN
5170 REM X PLOT
5180 FOR I=1 TO N :V(I)=X(I) :NEXT I
5190 FOR I=1 TO N
5200 V(I)=550+INT(400*V(I))
5210 U(I)=150+INT(NS*(I-1))
5220 NEXT I
5230 LPRINT" ;: P3 HA 450,1105 D 505,1105 U 518,1097 "
5240 LPRINT" S11 INPUT SIGNAL "CHR$(95)
5250 FOR L=1 TO 10000:NEXT L
5260 LPRINT U(1);",";V(1)
5270 LPRINT" D "
5280 FOR I=2 TO N
5290 LPRINT U(I);",";V(I)
5300 FOR L=1 TO 60 :NEXT L
5310 NEXT I
5320 LPRINT" P0 HA 300,50 Z "
5330 PRINT@(23,0)," "
5340 INPUT" AFTER THE PLOTTER STOPPED PRESS ENTER";
5350 CLS:RETURN
5360 REM Y PLOT
5370 FOR I=1 TO N :V(I)=Y(I) :NEXT I
5380 FOR I= 1 TO N
5390 V(I)=550+INT(400*V(I))
5400 U(I)=150+INT(NS*(I-1))
5410 NEXT I
5420 LPRINT" ;: P4 HA 450,1055 D 505,1055 U 518,1047 "
5430 LPRINT" S11 OUTPUT SIGNAL "CHR$(94)
5440 LPRINT" AFTER 161 ITERATIONS"CHR$(95)
5450 FOR L=1 TO 15000:NEXT L
5460 LPRINT U(1);",";V(1)
5470 LPRINT" D "
```

```
5480 FOR I=2 TO N
5490 LPRINT U(I);",";V(I)
5500 FOR L=1 TO 40 :NEXT L
5510 NEXT I
5520 LPRINT" PØ HA 300,50 Z "
5530 PRINT@(23,Ø)," "
5540 INPUT"AFTER THE PLOTTER STOPPED PRESS ENTER";
5550 CLS:RETURN
5560 REM DISPLAY E
5570 IF PR$="Y" THEN GOTO 5710
5580 CLS:GOSUB 6330
5590 PRINT@(21,Ø),"TO SCALE THE PAPER"
5600 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
5610 CLS
5620 IF AG$="Y" THEN GOSUB 4620
5630 PRINT@(23,Ø),"TO PLOT THE ERROR"
5640 INPUT"ENTER  Y  OTHERWISE  N ";AG$
5650 CLS
5660 IF AG$="Y" THEN GOSUB 5890
5670 PRINT@(23,Ø),"TO CLEAR THE SCREEN"
5680 INPUT"ENTER  Y  OTHERWISE  N  ";AG$
5690 IF AG$="Y" THEN CLS2
5700 GOTO 5720
5710 GOSUB 5730
5720 CLS:RETURN
5730 REM PRINT E
5740 PRINT
5750 PRINT"    THE MEAN SQUARE ERROR"
5760 PRINT
5770 FOR I=1 TO NM STEP 8
5780 PRINT USING"+###.####";E(I);E(I+1);E(I+2);E(I+3);
5790 PRINT USING"+###.####";E(I+4);E(I+5);E(I+6);E(I+7)
5800 NEXT I
5810 LPRINT
5820 LPRINT"    THE MEAN SQUARE ERROR"
5830 LPRINT
5840 FOR I=1 TO NM STEP 8
5850 LPRINT USING"+###.####";E(I);E(I+1);E(I+2);E(I+3);
5860 LPRINT USING"+###.####";E(I+4);E(I+5);E(I+6);E(I+7)
5870 NEXT I
5880 RETURN
5890 REM PLOT E
5900 FOR I=1 TO NM :V(I)=E(I) :NEXT I
5910 FOR I=1 TO NM
5920 V(I)=150+INT(800*V(I))
5930 U(I)=150+INT(NE*(I-1))
5940 NEXT I
5950 LPRINT" ;: P2 HA 450,1155 D 505,1155 U 518,1147 "
5960 LPRINT" S11 MEAN SQUARE ERROR"CHR$(95)
5970 FOR LL=1 TO 12000:NEXT LL
5980 LPRINT U(1);",";V(1)
5990 LPRINT" D "
6000 FOR I=2 TO NM
6010 LPRINT U(I);",";V(I)
6020 FOR L=1 TO 60 :NEXT L
6030 NEXT I
```

```
6040 LPRINT" P0 HA 300,0 Z "
6050 CLS:PRINT@(23,0),
6060 INPUT"AFTER THE PLOTTER STOPPED PRESS ENTER";
6070 CLS:RETURN
6080 REM SCREEN CURRENT RESULTS
6090 FOR I=1 TO N :U(I)=D(I) :NEXT I
6100 FOR I=1 TO N
6110 U(I)=80-(INT(40*U(I)))
6120 NEXT I
6130 FOR I=0 TO N-2
6140 LINE(NP*I,U(I+1))-(NP*I+NP,U(I+2)),,,&HF1F1
6150 NEXT I
6160 RETURN
6170 FOR I=1 TO N :U(I)=X(I) :NEXT I
6180 FOR I=1 TO N
6190 U(I)=80-(INT(40*U(I)))
6200 NEXT I
6210 FOR I=0 TO N-2
6220 LINE(NP*I,U(I+1))-(NP*I+NP,U(I+2)),,,&H3333
6230 NEXT I
6240 RETURN
6250 FOR I=1 TO N :U(I)=Y(I) :NEXT I
6260 FOR I=1 TO N
6270 U(I)=80-(INT(40*U(I)))
6280 NEXT I
6290 FOR I=0 TO N-2
6300 LINE(NP*I,U(I+1))-(NP*I+NP,U(I+2)),,,&HFFFF
6310 NEXT I
6320 RETURN
6330 FOR I=1 TO NM:U(I)=E(I):NEXT I
6340 FOR I=1 TO NM
6350 U(I)=200-(INT(200*U(I)))
6360 NEXT I
6370 FOR I=0 TO NM-2
6380 LINE(NL*I,U(I+1))-(NL*I+NL,U(I+2)),,,&HFFFF
6390 NEXT I
6400 PRINT@(0,50),"SCALE FACTOR =";DE
6410 RETURN
```

REFERENCES

[1] J.L.Walsh, A closed set of orthogonal functions, Amer. J. Math., Vol. 45, pp.5-24, 1923.

[2] N.A.Alexandridis and A.Klinger, "Real time Walsh-Hadamard transformation," IEEE Trans. Comp. pp.288-292 March 1972.

[3] W.K.Pratt, J.Kane, H.C.Andrews, "Hadamard transform image coding," IEEE proceedins, pp.58-68, Jan. 1969.

[4] D.E.Bowyer, Walsh function, "Hadamard matrices and data compresion," IEEE Trans. Electromagnetic Compatibility, pp.33-37, August 1971.

[5] J.D.Kennedy, "Walsh function imagery analysis," IEEE Trans. Electromagnetic Compatibility, pp.7-10, August 1971.

[6] W.K.Pratt, "Linear and nonolinear filtering in the Walsh domain," IEEE Trans. Electromagnetic Compatibility, pp.38-42, August 1971.

[7] S.K.Fletcher, "Walsh transforms in seismic event detection," IEEE Trans. Electromagnetic Compatibility, pp.367-369, August 1983.

[8] K.G.Beauchamp, Walsh Functions and their Applications, Academic Press, 1975.

[9] H.F.Harmuth, "A generalised concept of frequency and some applications," IEEE Trans. Inform. Theory, pp.375-382, May 1968.

[10] N.M.Blachman, "Sinusoids versus Walsh functions," IEEE Proceedings pp.346-354, March 1974.

[11] H.F.Harmuth, Transmission of Information by Orthogonal Functions, Spriger-Verlag, 1972.

[12] L.Zhihua, Z.Qishan, "Ordering of Walsh functios," IEEE Trans. Electromagnetic Compatibility, pp.115-119, May 1983.

[13] Y.Tadokoro, T.Highucci, "Discrete fourier transform computation via the Walsh transform," IEEE Trans.Acoust.Speech and Signal Proc., pp.236-240, June 1978.

[14] N.Ahmet and K,R.Rao, Orthogonal Transforms for Digital Signal Processing, Springer-Verlag 1975.

[15] F.Ying at al., "Speech processing with Walsh-Hadamard transforms," IEEE Trans. Audio and Electroacoustics, pp.174-179 June 1973.

[16] A.C.Davies, "On the definition and generation of Walsh functions," IEEE Trans. Comp. pp.187-189, Feb.1972.

[17] J.W.Manz, "A sequency-ordered fast Walsh transform," IEEE Trans. Audio and Electroacoustics, pp.204-205, August 1972.

[18] R.D.Brown, "A recursive algorithm for sequency ordered Walsh transforms," IEEE Trans. Comp. pp.819-882, August 1977.

[19] B.Widrow at.all, "Adaptive noise cancelling: Principles and applications," IEEE Proceedings, pp.1692-1716, Dec.1978.

[20] B.Widrow, "Adaptive filters," pp.563-587 in R.Kalman and N.DeClaris Aspects of Network and System Theory, Holt, Rinehalt and Winston 1971.

[21] C.F.N.Cowan and P.M.Grant, Adaptive Filters, Printice Hall 1985

[22] A.Morgul at.all, "Wide band hybrid analog/digital frequency domain adaptive filter," IEEE Trans. Acoust.Speech and Signal Processing pp.762-769, August 1984.

[23] S.S.Narayan, "Transform domain LMS algorithm," IEEE Trans. Acoust. Speech and Signal Processsing, pp.609-615, June 1983.

[24] B.Widrow and S.D.Stearns, Adaptive Signal Processing, Printice Hall 1985.