RECURSIVE GENERATION AND COMPUTATION OF FAST UNITARY TRANSFORMS

by

Bernard J. L. Fino

Memorandum No. ERL-M415

2 January 1974

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

To Nadia

A C K N O W L E G M E N T S

I am indebted to Professor V. Ralph Algazi, my research adviser, for his guidance in this thesis and his criticism at the various stages of development. I would like to express also my gratitude to Professor David J. Sakrison for some fruitful and crucial discussions, to Professor Elie I. Jury for suggesting the research that lead to chapters 4 and 5, and to Mr. Jacques Poncin for introducing me to the world of fast unitary transforms while I was with his group at the Centre National d'Etudes des Telecommunications.

I would like finally to thank the agencies which sponsored my studies at Berkeley: the Institut de Recherches en Informatique et Automatique in 1968-1970, the European Space Research Organization and NASA in 1972, and the Electronic Research Laboratory of the University of California at Berkeley in 1973.

This research was sponsored in part by the National Aeronautic and Space Adminitration under Grant NASA-NGR-05-003-538, and in part by the National Science Foundation under Grant NSF-GK-37282.

C O N T E N T S

C H A P T E R I

INTRODUCTION

1-1. Historical notes :

By the end of the 19 th century, two sets of orthogonal functions were already well known by electrical engineers: the block pulses simply generated by switches and the Sine-Cosine functions generated by resonant circuits made of linear, time invariant components. The appearance of semi-conductors in the middle of this century made possible the generation of other sets of orthogonal functions such as the Walsh-Hadamard functions. Recently, digital circuits are being prefered to analog circuits, enlarging the need for theoretical tools and methods adapted to digital techniques. The advent, at last, of the sophisticated digital computers finally forced the electrical engineers to "think digital" even when he is studying continuous phenomena and simulating them on a digital computer. This transition has been first directed to the use of digital methods as approximations to continuous problems and this explains the success of the Fast Fourier Transform (FFT) algorithm known as the Cooley-Tukey algorithm [1]. Only very recently, we have seen the "explosion" of the field for theoretical as well as practical studies of intrinsically digital methods. Every year since 1970, the Symposium on the Applications of the Walsh functions [2] draws a considerable interest. Several books [3] [4] [5] [6] try to present unified views of the field. It seems clear however that the field is only in its infancy and that important

further developments can be expected in the future.

1-2. The set of Walsh-Hadamard functions:

Known for centuries by artists[1], since the beginning of this century by mathematicians [7] [8] [9], they are now a fashion for electrical engineers. Their rapid success comes certainly from both their simplicity, their fast algorithm [10] and their analogies with the Fourier functions. The time shift of the Fourier analysis is replaced by the dyadic shift of the Walsh analysis. The frequency ordering is replaced by the sequency ordering. But it became rapidly apparent that differences, practical as well as theoratical, existed in the use of these functions for signal analysis. Applications which bear no analogy to the known usage of Fourier functions, are currently sought and some surprising applications have been discovered : Harmuth's book [4] points many challenging areas for basic research in most of the fields traditionally studied with Fourier analysis, and also other intrinsically digital fields such as coding theory where some early results seem promising.

1-3 Fast unitary transforms and their applications to signal processing :

Fast unitary transforms have been used mostly for signal encoding and , to a lesser extent, for signal representation, dimensionality reduction and signal filtering. For these applications, the expression of the signal in a transform domain leads to simpler processing at the cost of slightly lower performance, compared to

[1] Mexican drawings at Mitla (near Oaxaca).

optimal processing. In chapter 7, we comment on the application of these transform techniques and review the main published works. Here, we wish to point out that only a few transforms have been considered (Fourier, Walsh-Hadamard, Haar, Slant transforms) and that these transforms are considered as given and used mostly on a"try and look" basis. One may wonder if these transforms have common properties and structure, and if other transforms of interest exist. Developing a structure common to all known transforms and studying their common properties is a goal of this thesis. In each specific application of unitary transforms, a major question of interest is the comparison of the transforms. We shall see that, in fact, we are faced with a ternary trade-off in which computational complexity has to be included and we shall discuss some practical cases of interest.

1-4. Organization of this thesis :

This research was motivated by some previous work on the application of transform techniques in image processing [11]. It appeared rapidly that various transforms were used "at random" and that some thoughts about their relations, about other transforms with a similar fast algorithm were necessary.

In chapter 2, we first relate the Haar transform with the well known Walsh-Hadamard (W-H) transform and present some theoratical consequences of practical importance.

In chapter 3, we present a unified treatment of unitary transforms having a fast algorithm. The use of recursive rules to describe unitary transforms allows a systematic way to view known transforms, to generate new transforms and provides a general approach to the

evaluation of the computational complexity.

In chapter 4 and 5, we use the framework of chapter 3 to analyse the round-off errors in the algorithms of the fast unitary transforms, with fixed-point representation of numbers (chapter 4), or floating-point representation (chapter 5). Our treatment is valid for most fast unitary transforms and considers all the cases of practical interest. In particular, the previous works concerned mainly with the analysis of the FFT are included with more accuracy and new results are developed.

In chapter 6, we consider fast unitary transforms with a given set of basis vectors and we study in detail a family of generalized Slant transforms, most of which are new.

In chapter 7, we describe the main applications of fast unitary transforms and present, with an example, a comparison of their performances.

The relations between these chapters follows the diagram of Fig.1-1.

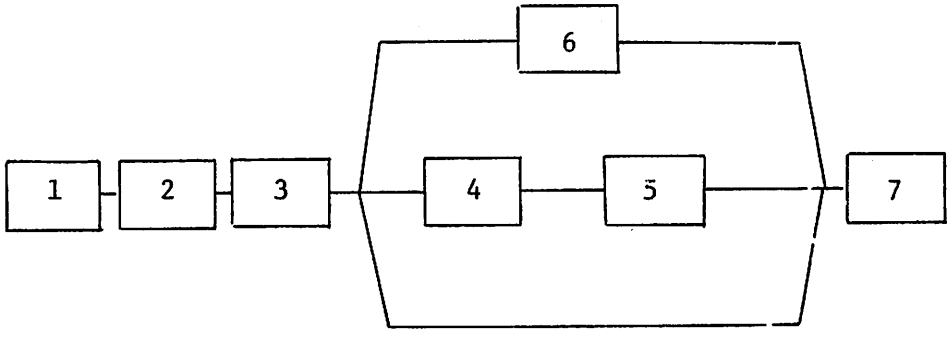


Fig. 1-1. : Relations between chapters

1-5. Review of generalized transforms :

In this section, we review and comment upon the previous contributions to the generalized fast unitary transforms.

1-5-1. Basic transforms : Fourier, Walsh-Hadamard, Haar, Slant

Fourier: The continuous Sine-Cosine functions and Fourier analysis have been known for a long time, but their extensive use had to wait for the factorisation property developed by Good [12] leading to the Cooley-Tukey algorithm [1]. This algorithm has been improved through the years. Gentleman & Sande [13] have developed a different organization of the algorithm. The interest of mixed radix algorithms was pointed by Gentleman & Sande [13](radix 4), Bergland [14] (radix 8) ; Singleton [15] reviewed and extended these results. The matrix notations for the FFT has been presented by Theiheimer [16] and Kahaner [17]. Glassman [18] studied the FFT algorithm for composite orders. Many publications [19] [20] [21] discuss other aspects of the FFT.

Walsh-Hadamard (also called BIFORE - BInary FOurier REpresentation) The Walsh functions were introduced by Walsh [8], some of their basic properties, leading in particular to the fast W-H algorithm [10], were presented by Paley [9], Fine [22], Pichler [23]. The interested reader can find in Harmuth' book [4] and in [24] excellent presentations of their properties. However to date not all their properties are yet clear, as indicated by the proliferation of notes and papers (see [25] for references of the most recent publications and the 1970-1971-1972-1973 Proc. of the Symp. on Applications of the Walsh functions). However their relation with the Fourier transform has been thoroughtly considered [26] [4].

Haar : The set of orthonormal Haar functions were found by Haar [27] and some convergence properties investigated by Kaczmarz [28] and Alexits [29]. Strangely enough, the Haar transform was considered in applications only by Andrews [5] for image encoding and rapidly rejected for unclear reasons. However, we have found [11] that the Haar transform behaves in image processing subjectively as well (if not better) as the W-H transform and we strongly believe that it deserves further consideration. In chapter 2 we prove formal relations between the Haar and W-H transforms and present some interesting theoretical consequences.

Slant transform : A slant transform has been introduced by Shibata and Hatori [30] [31] for TV signals. Pratt & Chen [32] have defined a slant transform of order 2^n. The results obtained with this transform are promising. Chapter 6 considers fast transforms including a set of slant vectors. Among them we find the slant transform.

1-5-2. Generalizations of the basic transforms:

Different independent generalizations have been proposed.

Extension to other roots of unity than ±1 for Haar and W-H :

The Haar and W-H transforms have only the roots of unity ±1 as coefficients (and 0 for Haar). Chrestenson [33] for the W-H transform and Watari [34] for the Haar transform, have described extensions using other roots of unity.

Different original matrices for W-H :

The W-H transform is built from the "original" matrix $\left[F_2\right] = 1/\sqrt{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Andrews and Caspari [35] [36] proposed a family of transforms built

on other matrices. Harmuth [4] defined the " generalized two-valued" transforms by considering other matrices still with entries ±1 as the original matrix. Similarly, he defined the " generalized three-valued transforms" by a "squeeze and shift" process.

Extension to complex transforms:

The FFT is naturally defined as a complex transform in order to make use of the shift property of the N th roots of unity which leads to the fast algorithm. The W-H transform is defined with real entries and an extension called complex W-H or complex BIFORE transform has been proposed by Gibbs [37] and Ohnsorg [38] and studied by Ahmed & al. [39] [40] . Ahmed & al. [41] found a modified complex W-H transform which, in fact is closely related to a complex Haar transform as it will appear later.

In the case of the FFT, the inverse process of reduction to a real transform leads also to the Discrete Cosine transform [42] recently introduced by Ahmed and al.

Families between Fourier and W-H :

Ahmed & al. [43] have defined a finite family of transforms including the W-H, complex W-H, and Fourier transforms. Their study of invariant quantities in the spectrum of these transforms has lead them to define the Modified Generalized transforms [44], which, as it will appear later are a finite family between the Haar and Fourier transforms. The definitions of these families as they appear in the original papers are very complex and will be omitted here. In chapter 3, we shall give simple definitions for these families.

Andrews & al [45] [35] [36] have seen the importance of the Kronecker

product of matrices to factorize the matrices of some fast unitary transforms and obtain a fast algorithm. We shall generalize this approach and introduce a generalized Kronecker product.

To conclude this introductory chapter we sum up in Fig. 1-2 the fast unitary transforms which appeared in the literature with their connections. This diagram clearly shows the need for a better structure : such a structure will be presented in the following chapters.

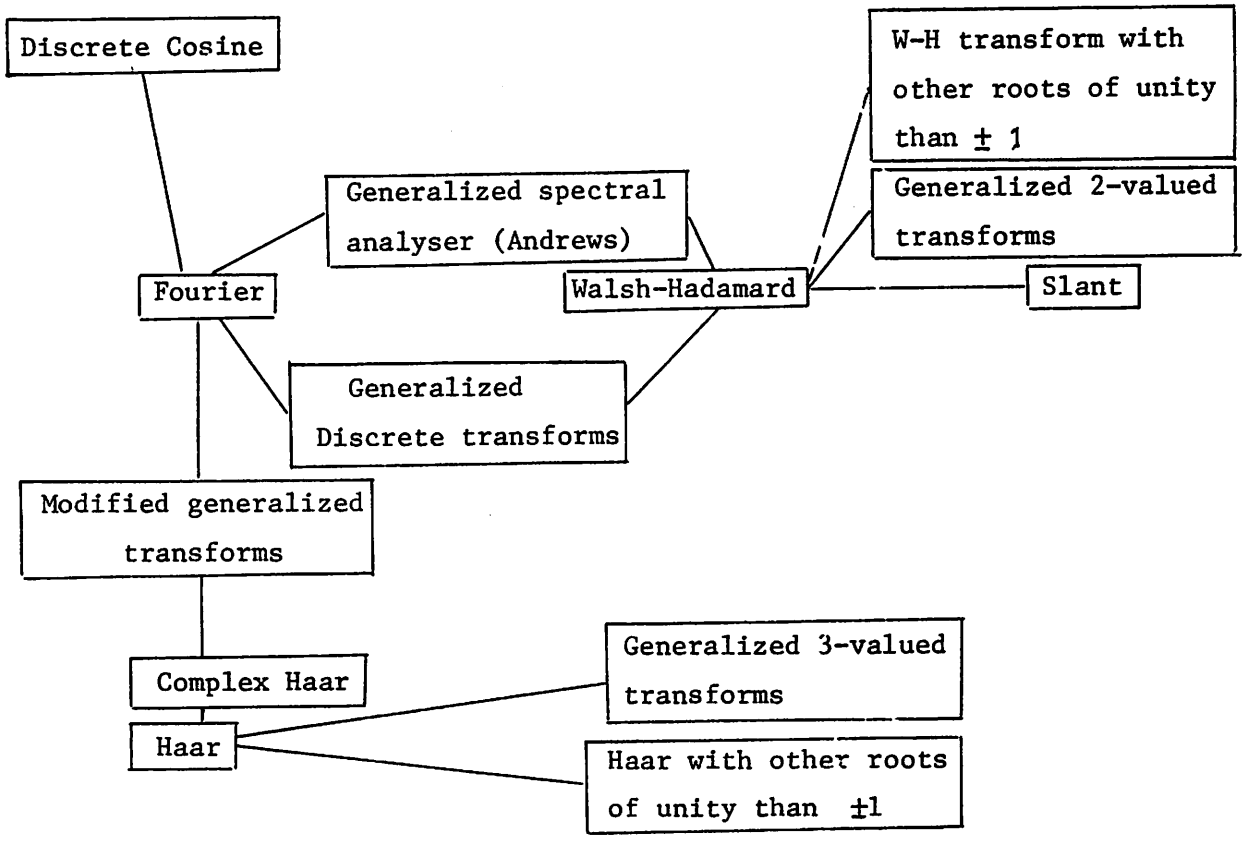


Fig. 1-2. Generalized transforms.

C H A P T E R II

RELATIONS BETWEEN HAAR AND WALSH-HADAMARD TRANSFORMS[1]

2-1. Introduction:

The basic results of this chapter are the formal matrix relations between the Haar and W-H transforms. First, we partition the Haar and "zequency"[2] ordered W-H matrices into submatrices and define the matrix operators $\mathcal{H}$ and $\mathcal{W}$ which express simply the recursive generations of the Haar and W-H matrices through their submatrices. Then, using the properties of the operators $\mathcal{H}$ and $\mathcal{W}$, we prove by induction the basic relations between the Haar and W-H submatrices.

Consequently, we derive the relations between the transform coefficients of an input vector by the two transforms: we define the "zones" in these transform coefficients and show that they are related through W-H transforms of lower orders. So, if we approximate a vector with the same zones of its transform coefficients by the two transforms, the Haar transform performs the same approximation with fewer elementary operations: this result is important in signal representation.

Then we show that the basic relations allow the recursive decomposition of the W-H transform algorithm into Haar transforms. Conversely, we propose a slightly modified W-H transform which

---

[1] Most of the results of this chapter appear in:
B.J. Fino, "Relations between Haar and Walsh-Hadamard transforms", Proc. of the IEEE (Letter), vol. 60, No. 5, pp. 647-648, May 1972.

[2] This terminology has been proposed by Yuen [1] because "sequency" received different definitions. Here, "zequency" is the number of zero crossings.

performs an efficient "pipe-line" Haar transform.

Finally, the decomposition of the algorithms of the two transforms suggests a family of unitary transforms between the Haar and W-H transforms.

2-2. Basic relations:

a) partition and recursive definition of the Haar matrices:

The original definition of the Haar functions [2] gives the following definition of the Haar matrix of order $N = 2^n$, denoted $\left[H_{2^n}\right]$:

$$H_{0,j} = \sqrt{2^{-n}} \quad \text{for all } j$$

$$H_{2^{k-1}+i,j} = \begin{cases} \sqrt{2^{k-n-1}} & \text{for } i\,2^{n-k+1} \leq j < 2^{n-k}(2i+1) \\ -\sqrt{2^{k-n-1}} & \text{for } 2^{n-k}(2i+1) \leq j < 2^{n-k+1}(i+1) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $k = 1,\ldots,n$, $i = 0,\ldots,2^{k-1}-1$, $j = 0,\ldots,2^n$

The Haar matrix of order 8 , $\left[H_8\right]$ is given in Figure 2-1.

Let us partition this square unitary matrix $\left[H_{2^n}\right]$ into (n+1) rectangular submatrices denoted $\left[MH^k_{2^n}\right]$ of dimensions (2^{k-1} x 2^n) and such that:

$$\text{entry}(i,j) \text{ of } \left[MH^k_{2^n}\right] = H_{2^{k-1}+i,j} \quad \text{for } i = 0,\ldots,2^{k-1}-1, \; k = 1,\ldots,n$$

and $$\text{entry}(0,j) \text{ of } \left[MH^0_{2^n}\right] = \sqrt{2^{-n}}$$

The submatrices of order 8 are shown in Figure 2-1.

We define now the matrix operator $\mathcal{H}$ which applied to an (m x p) matrix $[M]$ gives the following (2m x 2p) matrix $\left[\mathcal{H}([M])\right]$

$$[H_8] = \frac{1}{\sqrt{8}}$$

1	1	1	1	1	1	1	1	$[MH_8^0]$
1	1	1	1	-1	-1	-1	-1	$[MH_8^1]$
2	2	-2	-2	0	0	0	0	
0	0	0	0	2	2	-2	-2	$[MH_8^2]$
2	-2	0	0	0	0	0	0	
0	0	2	-2	0	0	0	0	
0	0	0	0	2	-2	0	0	$[MH_8^3]$
0	0	0	0	0	0	2	-2	

Fig.2-1: Haar matrix and submatrices of order 8

$$[WH_8] = \frac{1}{\sqrt{8}}$$

1	1	1	1	1	1	1	1	$[MW_8^0]$
1	1	1	1	-1	-1	-1	-1	$[MW_8^1]$
1	1	-1	-1	-1	-1	1	1	
1	1	-1	-1	1	1	-1	-1	$[MW_8^2]$
1	-1	-1	1	1	-1	-1	1	
1	-1	-1	1	-1	1	1	-1	
1	-1	1	-1	-1	1	-1	1	$[MW_8^3]$
1	-1	1	-1	1	-1	1	-1	

Fig. 2-2. : Walsh-Hadamard matrix and submatrices of order 8

$$\left[\mathcal{H}(\left[M\right])\right] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \left[M\right] = \left|\begin{array}{c|c} \left[M\right] & 0 \\ \hline 0 & \left[M\right] \end{array}\right|$$

where $\otimes$ denotes the Kronecker product of matrices [3]. If we apply $\mathcal{H}$ to the submatrices $\left[MH_{2^n}^{k}\right]$, it is easy to show that:

$$\left[MH_{2^n}^{k}\right] = \left[\mathcal{H}\left(\left[MH_{2^{n-1}}^{k-1}\right]\right)\right] \qquad k = 2,\ldots,n \tag{2}$$

b) <u>partition and recursive definition of the W-H matrices</u>:

There are different definitions for the W-H matrices giving different orderings of the rows (see discussion in chapter 3). The natural ordering appears when we define recursively the W-H matrices, denoted $\left[W_{2^n}\right]$, by Kronecker products :

$$\left[W_2\right] = 1/\sqrt{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\left[W_{2^n}\right] = \left[W_2\right] \otimes \left[W_{2^{n-1}}\right] \tag{3}$$

However, the zequency ordering, which appears in the original paper by Walsh [4], is more appropriate to signal representation and we use it in the following. In order to express simply the recursive generation of W-H matrices in zequency order, we define the matrix operators $\mathcal{W}$ and $\mathcal{W}'$. These operators, applied to a (m x p) matrix $\left[M\right]$ give respectively the matrices $\left[\mathcal{W}(\left[M\right])\right]$ and $\left[\mathcal{W}'(\left[M\right])\right]$ such that each row $(a_1,a_2,\ldots,a_m)$ of $\left[M\right]$ is replaced by the two rows $\left|\begin{matrix}\alpha \\ \alpha'\end{matrix}\right.$ for $\mathcal{W}$ and $\left|\begin{matrix}\alpha' \\ \alpha\end{matrix}\right.$ for $\mathcal{W}'$, in which

$$\alpha = (a_1,a_2,\ldots,a_m,-\text{sign}(a_1\ a_m) \times (a_1,a_2,\ldots,a_m))$$

and $\mathcal{L}' = (a_1, a_2, \ldots, a_m, -\text{sign}(a_1\, a_m) \times (a_1, a_2, \ldots, a_m))$ (4b)

If we apply $\mathcal{W}$ to a matrix $[M]$ of p consecutive rows of the W-H matrix of order 2^n and in zequency order, it is easy to show that $[\mathcal{W}([M])]$ is a matrix of 2p consecutive rows of the W-H matrix of order 2^{n+1}. In particular, if we partition the zequency ordered W-H matrices into submatrices $\left[MW_{2^n}^{k}\right]$ similar to the Haar submatrices, we have:

$$\left[MW_{2^n}^{k}\right] = 1/\sqrt{2}\ \left[\mathcal{W}\left(\left[MW_{2^{n-1}}^{k-1}\right]\right)\right] \qquad k = 1, \ldots, n \tag{5}$$

and also

$$\left[WH_{2^n}\right] = 1/\sqrt{2}\ \left[\mathcal{W}\left(\left[WH_{2^{n-1}}\right]\right)\right] \tag{6}$$

The W-H matrix and the W-H submatrices of order 8 are presented in Figure 2-2.

The relations (5) and (6) are matrix expressions of the difference equations taken as the definition for the W-H functions by Harmuth [5]. Note that the set of W-H functions defined by Harmuth has some sign differences with ours.

c) <u>properties of the operators $\mathcal{H}$, $\mathcal{W}$ and $\mathcal{W}'$</u> :

1) We denote $[S_m]$ the permutation matrix of order m which reverses the ordering of rows or columns:

$$[S_m] = \begin{bmatrix} & & & 1 \\ & & \cdot^{\cdot} & \\ & \cdot^{\cdot} & & \\ 1 & & & \end{bmatrix}$$

It is easy to see that:

$$\left[S_{2m}\right]\ \left[\mathcal{W}(\left[M\right])\right] = \left[\mathcal{W}'(\ \left[S_m\right]\ \left[M\right]\ )\right] \tag{7}$$

$$\text{and}\ \left[S_{2m}\right]\ \left[\mathcal{W}'(\left[M\right])\right] = \left[\mathcal{W}(\ \left[S_m\right]\ \left[M\right]\ )\right] \tag{8}$$

2) Let $\left[M\right] = \left\{m_{ij}\right\}$ be a (m x p) matrix, $\left[N\right]$ a (p x q) matrix so that their matix product $\left[P\right] = \left[M\right]\ \left[N\right] = \left\{p_{ij}\right\}$ exists. We say that the product matrix $\left[P\right]$ is sign invariant /contravariant if, for all i , $(p_{il} \times p_{iq})$ has same/ opposite sign as $(m_{il} \times m_{ip})$. Then, if P is sign invariant,

$$\left[\mathcal{W}(\left[M\right])\right]\ \left[\mathcal{H}(\left[N\right])\right] = \left[\mathcal{W}(\left[P\right])\right] \tag{9}$$

if P is sign contravariant,

$$\left[\mathcal{W}(\left[M\right])\right]\ \left[\mathcal{H}(\left[N\right])\right] = \left[\mathcal{W}'(\left[P\right])\right] \tag{10}$$

To prove (9), we consider the product of the (2i) th row (type α)or of the (2i+1) th row (type α') of $\left[\mathcal{W}(\left[M\right])\right]$ by $\left[\mathcal{H}(\left[N\right])\right]$:

$$(m_{il}, m_{i2}, \ldots\ldots, m_{ip}, \varepsilon\ \text{sign}(m_{il}\ m_{ip}) \times (m_{il}, m_{i2}, \ldots\ldots, m_{ip}))\ \left[\begin{array}{c|c} [N] & 0 \\ \hline 0 & [N] \end{array}\right]$$

where $\varepsilon = +1$ for α and -1 for α'.

By matrix multiplication, the product is :

$$(p_{il}, \ldots\ldots, p_{iq}, \varepsilon\ \text{sign}(m_{il}\ m_{ip}) \times (p_{il}, \ldots\ldots, p_{iq}))$$

So $\mathcal{W}$ applied to $\left[P\right]$ will give the same result as the matrix product $\left[\mathcal{W}(\left[M\right])\right]\ \left[\mathcal{H}(\left[N\right])\right]$, if $\left[P\right]$ is sign invariant. A similar proof holds for (10).

d) <u>basic relations between Haar and Walsh-Hadamard transforms</u>:

Alexits [6] has suggested that a matrix relation exists between the submatrices $\left[MH_{2^n}^{k}\right]$ and $\left[MW_{2^n}^{k}\right]$. We now prove by induction on n that this relation is :

$$\left[MW_{2^n}^{k}\right] = \left[S_{2^{k-1}}\right] \left[WH_{2^{k-1}}\right] \left[MH_{2^n}^{k}\right] \qquad (11)$$
$$k = 1,\ldots\ldots,n$$

Assume that (11) is true at the order (n-1), we have:

$$\left[MW_{2^{n-1}}^{k}\right] = \left[S_{2^{k-1}}\right] \left[WH_{2^{k-1}}\right] \left[MH_{2^{n-1}}^{k}\right] \qquad k = 1,\ldots\ldots,n-1 \qquad (12)$$

Applying $\mathcal{W}$ to both sides of (12) and using (5) for the left hand side, (8) for the right hand side, we obtain:

$$\sqrt{2}\left[MW_{2^n}^{k+1}\right] = \left[S_{2^k}\right] \left[\mathcal{W}'\left(\left[WH_{2^{k-1}}\right] \left[MH_{2^{n-1}}^{k}\right]\right)\right]$$

It is easy to see that the product of a matrix by a Haar submatrix is sign contravariant ; thus, using (10), (2) and (6) , we obtain (11) for k = 2,.....,n. As (11) is obvious for k = 1, this completes the proof of (11).

As $\left[WH_{2^{k-1}}\right]$ and $\left[S_{2^{k-1}}\right]$ are real, symmetric and unitary, they are self-inverse and we have the converse relation:

$$\left[MH_{2^n}^{k}\right] = \left[WH_{2^{k-1}}\right] \left[S_{2^{k-1}}\right] \left[MW_{2^n}^{k}\right] \qquad (13)$$

2-3 Zonal relations between Haar and W-H transform vectors:

$\vec{V}$ $(V_0,\ldots\ldots,V_{2^n-1})$ is an input vector ; $\vec{VH}$ $(VH_0,\ldots\ldots,VH_{2^n-1})$ and $\vec{VW}$ $(VW_0,\ldots\ldots,VW_{2^n-1})$ are the vectors obtained from $\vec{V}$ by the Haar and W-H transforms :

$$\vec{VH} = \left[H_{2^n}\right] \vec{V}$$

$$\vec{VW} = \left[WH_{2^n}\right] \vec{V}$$

Then the matrix relations (11) and (13) imply some relations between $\vec{VH}$ and $\vec{VW}$. If we right multiply (11) by $\vec{V}$, we obtain :

$$\begin{pmatrix} VW_{2^{k-1}} \\ \vdots \\ VW_{2^k-1} \end{pmatrix} = \begin{bmatrix} S_{2^{k-1}} \end{bmatrix} \quad \begin{bmatrix} WH_{2^{k-1}} \end{bmatrix} \quad \begin{pmatrix} VH_{2^{k-1}} \\ \vdots \\ VH_{2^k-1} \end{pmatrix} \qquad (14)$$

$$k = 1,\ldots,n$$

The converse relation is similarly obtained from (13)

Let a "zone" be the set of coefficients of the transform vectors which appear in the relations (14) for k= 1,....,n. We see that corresponding zones of the two transforms are related by a W-H matrix and a permutation matrix, therefore by a unitary matrix. Consequently, a zone in a transform vector determines the corresponding zone of the other transform vector. This property shows that, if we approximate $\vec{V}$ by the same subset of zones of the vectors $\vec{VH}$ and $\vec{VW}$ or in particular if we truncate these vectors at the end of a zone, we obtain the same approximation vectors after inverse transformation.

As the zones are related through a unitary transform, it follows by Parseval's theorem (valid for any unitary transform) that the energies in corresponding zones are identical. These results show that the performances of the two transforms for signal representation are identical if the zones are maintained and should be close if they are not. This result has been verified for image encoding with transform techniques [7] . The computational complexity of these transforms has been studied by Andrews [8] (see also chapter 3) : the W-H transform of order 2^n requires $n\,2^n$ additions and the Haar transform of same order requires $2(2^n-1)$ additions and 2^{n-1} multiplications[3] so

[3] In some applications such as threshold encoding [7] the multiplications required by the normalizations may not be necessary.

$5\ 2^{n-1}-2$ elementary operations. The ratio of the total number of elementary operations for both transform appear in Table I.

Table I

n	1	2	3	4	5	6	7	8	9
$N=2^n$	2	4	8	16	32	64	128	256	512
n N	2	8	24	64	160	384	896	2048	4608
$52^{n-1}-2$	2	8	18	38	78	158	318	638	1278
Ratio	1	1	1.33	1.71	2.05	2.42	2.83	3.21	3.60

We may expect the Haar transform to be faster than the W-H transform for most implementations.

Therefore, for computations in which the zones are maintained, the Haar transform performs as well and is faster than the W-H transform.

2-4. Application of the basic relations to fast algorithms:

The flowgraph of a fast algorithm for the Haar transform is given in Figure 2-3 for vectors of order 8. Several fast algorithms for the W-H transform have been proposed [9] [10] [11]. Using recursively the relation (11), we can decompose the W-H algorithm into a Haar transform of same order followed by W-H transforms of lower orders which can also be decomposed. This procedure gives, for the W-H transform of order 8, the fast algorithm of Figure 2-4, where the Haar transforms appear inside dotted lines. We call the reordering shown on Figure 2-4 a " bit-inversion" ; we shall discuss it in more detail in chapter 3. The algorithms for the two transforms, as they appear in Figures 2-3 and 2-4, require the same number of stages of computation, but not the

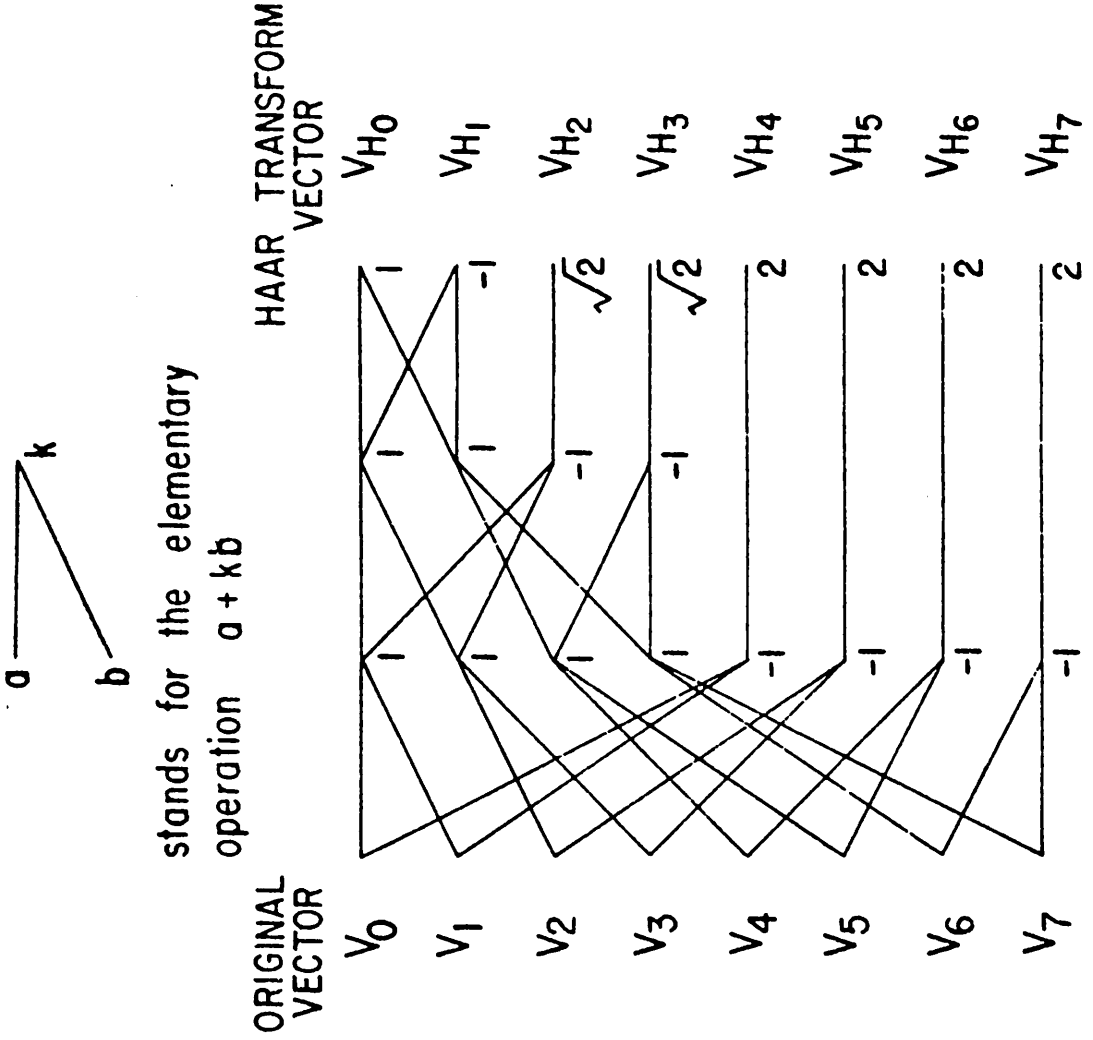


Fig. 2-3. : Fast Haar transform of order 8

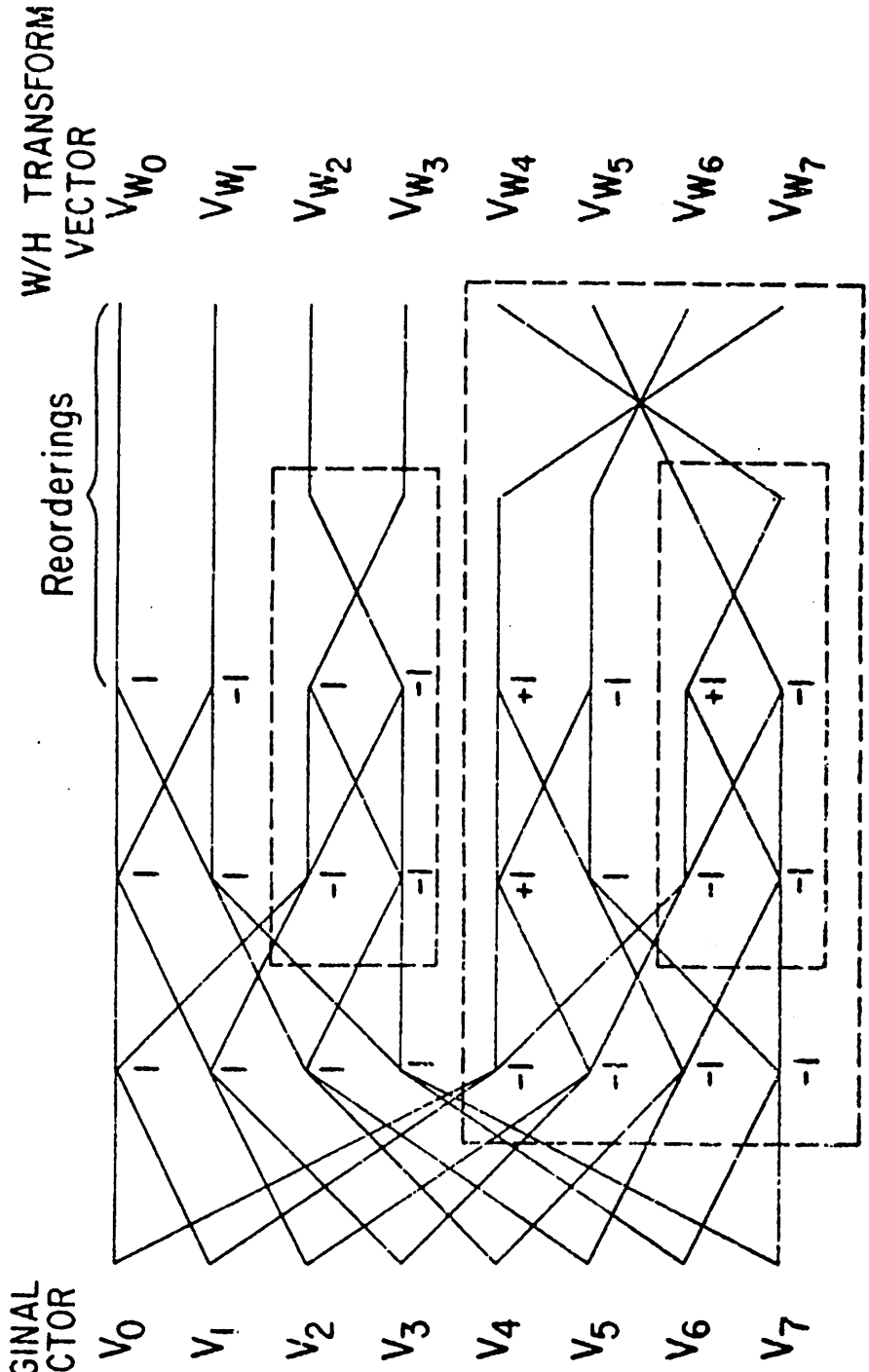


Fig. 2-4. : Fast Walsh-Hadamard transform of order 8

same number of elementary operations. If we can implement these algorithms with all elementary operations of a stage of computation performed in parallel, the Haar transform will not be faster than the W-H transform. However, if we have a sequence of vectors to transform by the Haar transform, we can make use of the relations between the two transforms to design a "pipe-line" algorithm for the Haar transform : at each stage of computation, a W-H stage of computation is constructed from Haar intermediate results of successive vectors each in a different stage of computation with respect to the Haar transform . We present in Fig. 2-5 a possible organization of this algorithm for Haar transforms of order 8. All stages of computation are identical for this organization and only one appears on the figure. On the first 14 cells , 14 adders operate to give the intermediate or final coefficients of the Haar transforms at a successive stage. The 10 last cells are storage cells and the stored data is properly shifted to gives the Haar transform coefficients of each successive vector in the 8 first cells after 3 cycles of computation. On the average three Haar transforms are performed with less than the hardware and time needed for two W-H transforms. More generally, n Haar transforms of order 2^n are produced with the equivalent hardware of two W-H transforms of the same order. This"pipe-line"algorithm is very efficient with parallel circuitry which is available at more and more competitive prices.

2-5. Family of unitary transforms inbetween the W-H and Haar transforms and with a common fast algorithm:

The decomposition of the fast W-H transform into a fast Haar transform of same order and (n-1) W-H transforms of lower orders,

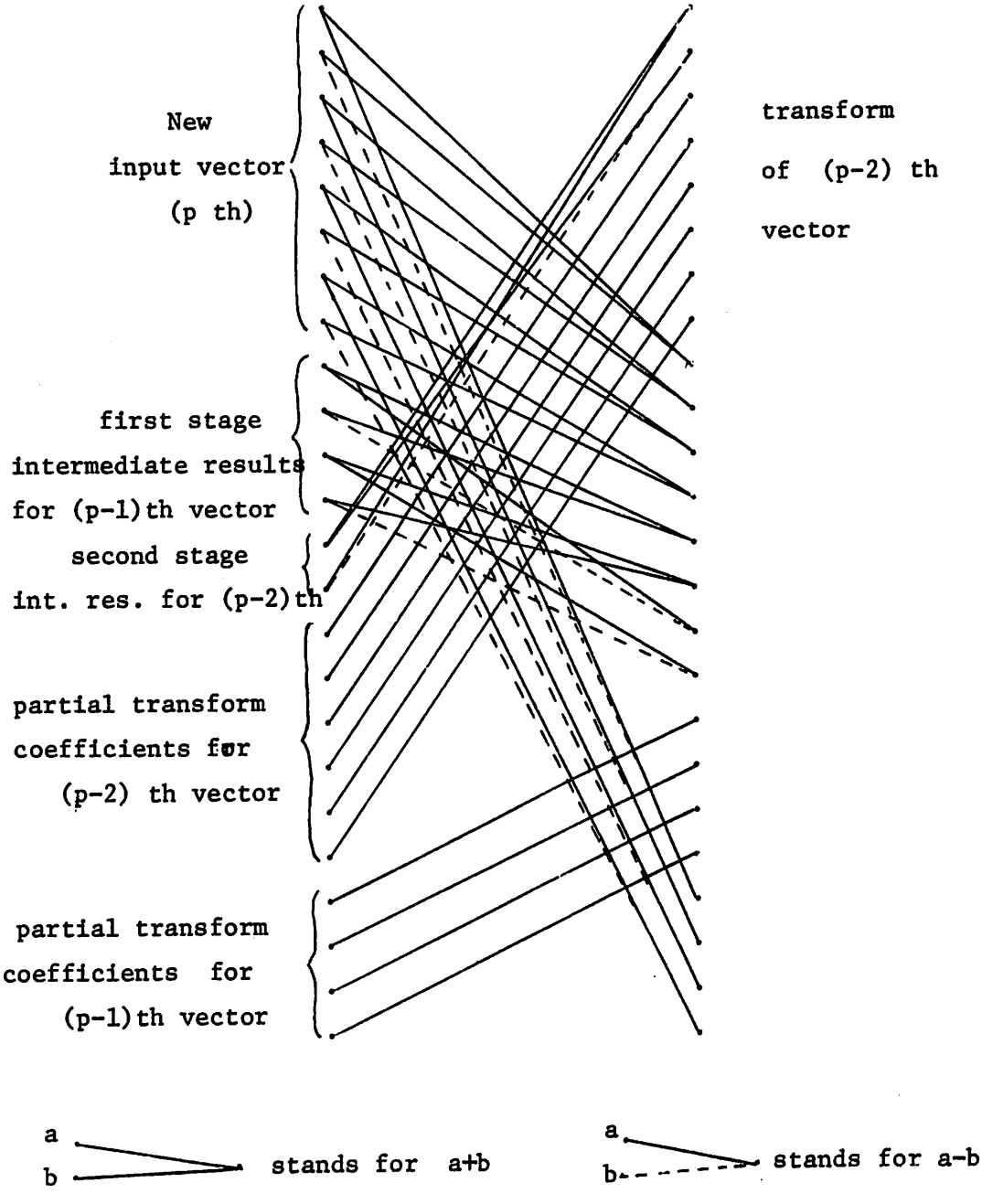


Fig. 2-5. : Pipe-line Haar transform of order 8

suggests that we can replace any of these W-H transforms by a Haar transform of the same order and still have a unitary transform. We can use again this rule in further decompositions. This procedure yields a family of unitary transforms with a common algorithm given in Fig. 2-4. The non-normalized transform coefficients appear at appropriate nodes of the algorithm. The number of members of this family, denoted P_n for the transforms of order 2^n, is given recursively by:

$$P_n = 1 + P_{n-1}\, P_{n-2} \cdots\cdots P_1 = 1 + P_{n-1}(P_{n-1} - 1)$$

The first values of P_n are (with $P_1 = 2 \rightarrow [F_2]$ and identity matrix)

n	1	2	3	4	5	6
P_n	2	3	7	43	1807	3 263 443 ....

In chapter 3, we present an even larger family of unitary transforms between the Haar and W-H transforms. This family includes the present family but has not the property of a common algorithm.

2-6. Conclusions :

In this chapter, we have developed some ties between the Haar and W-H transforms : formal relations between submatrices, corresponding relations between zones of transform vectors, relations between the fast algorithms and finally a family of transforms including both Haar and W-H transforms.

We have also found that the Haar transform, with a faster algorithm can perform as well as the W-H transform : we conclude that the Haar transform, for long forgotten and too hastly rejected, should recieve more attention.

C H A P T E R III

A UNIFIED TREATMENT OF DISCRETE UNITARY TRANSFORMS WITH A FAST ALGORITHM

3-1. Introduction :

The dissemination of the Fast Fourier Transform algorithms, originally introduced by Good [1], and known as Cooley-Tukey [2] and Sande-Tukey [3] algorithms, has resulted in a large extension in the range of applications of the well known Fourier transform. Recently the Walsh-Hadamard transform, also with a fast algorithm [4] has drawn considerable interest [5]. The Haar transform although closely related to the Walsh-Hadamard transform and potentially of interest [6] [7], has received much less attention. These transforms have been used successfully for error free signal representation [8], pattern classification [4], [9], speech signal encoding [10] and above all for picture encoding [11], [12], [13]. Only a few transforms have been considered in these applications while many other transforms could be of interest. Some workers have considered the definition of generalized transforms and we mention the works by Andrews, et al [14], [15], [16], Rao, et al [17], [18] and Harmuth [19].

In this chapter, we present a unified view of discrete unitary transforms with a fast algorithm. A discrete unitary transform is characterized by a unitary matrix [T] such that $[T][T^{*t}] = [I]$ where * denotes conjugate "t" transpose and [I] is the identity matrix of same order as [T], say N. For mathematicians a unitary matrix expresses a rotation of the orthonormal basis and preserves the Euclidian norm $\|V\| = \vec{V}\cdot\vec{V}^{*t}$, of any vector $\vec{V}$ and all inner products of vectors. In signal representation, this property

means energy conservation and an easy expression of the mean square error when some components of the signal are ignored in the new base. The computation of the transformed vector $\vec{W}$ of $\vec{V}$ by the transform [T] such that $\vec{W} = [T]\vec{V}$ usually requires N^2 multiplications and N(N-1) additions. For some specific transforms of interest such as the Fourier, Walsh-Hadamard transforms a fast algorithm has been found which requires fewer elementary operations. The analysis of these fast algorithms has been done by factorization of the matrix [T] into a set of largely sparse matrices, each expressing a stage of computation. This is the approach followed by Good [1] in his original paper which lead to the Fast Fourier Transform [2], [3] the Fast Walsh Transform [4] and other known fast transforms.

Here we consider recursive rules[1] for the generation of unitary transforms having a fast algorithm. These rules allow us to generate large classes of such transforms, many of which are new and possibly of interest, and to give general formulas for the number of elementary operations required by the corresponding fast algorithm.

3-2 . Recursive Generative Rules:

We shall present three rules which generate a new unitary matrix from some original unitary matrices. For each rule we relate the number of elementary operations for the new transform to the number of elementary operations of the same type required by the original transforms. For rule 1 there is only one original matrix, for rule 2 two, and for rule 3 a set of original matrices.

---

[1] We denote by "rule" a set of operations performed in a prescribed order. We reserve the term "operation" for the elementary operations such as additions, multiplications, etc. which determine the computational complexity of a transform.

Rule 1: Operations on the columns of a unitary matrix:

Given a unitary matrix [T], two obvious operations on the columns yield another unitary matrix of some order:

a) permutation of the columns: This operation does not require any computation. In the computational process, this operation can be performed by applying the permutation to the coefficients of the input vector instead of the columns themselves.

b) multiplication of a column by a root of unity: This operation requires a complex multiplication if the root of unity is not ± 1 or $\pm j$ ($j = \sqrt{-1}$) (see footnote 2).

These operations on the columns may be expressed by a matrix product [T] [D] with [D] such that $D_{ki} = e^{j\theta i}$ if column k is to be replaced by column i multiplied by the root of unity, $e^{j\theta i}$, and all other entries of [D] are null.

Rule 2: Rotation of rows by a unitary matrix

Consider a unitary matrix [T] of order N. The N row vectors form an orthonormal basis for S_N, the N dimensional space they span. m row vectors of [T] form an orthonormal basis for a subspace S_m. If these m vectors are rotated by a unitary matrix [U] of order m, we obtain a new orthonormal basis $\mathcal{B}$ for S_m. The remaining unchanged N-m rows of [T] are an orthonormal basis of the subspace S_{N-m} orthogonal to S_m and form with $\mathcal{B}$ a new orthonormal basis for S_N. Thus, the matrix [T'] obtained after rotation of the m rows by the unitary matrix [U] is unitary.

---

[2] Multiplications by ± 1 and $\pm j$ may be counted as operations if the hardware realization of the algorithm is not able to keep track of them. However, for the error analysis of the algorithm these multiplications, even if they are performed, do not introduce any error.

Some particular cases of interest are:

a) multiplication of the whole matrix by a unitary matrix of the same order

b) permutation of the rows (multiplication by a permutation matrix)

c) multiplication of a row by any root of unity.

The operations b and c can be represented by the matrix product [D] [T] where [D] is, as before, such that $D_{ik} = e^{j\theta i}$ if row i of T is replaced by row k multiplied by the root of unity, $e^{j\theta i}$, and all other entries of [D] are null.

Number of Elementary Operations:

If transforms T and U require respectively t and u elementary operations of a specific kind, it is obvious that the transform T' will require at most t' of these operations with

$$t' = t + u \qquad (1)$$

(It may happen that [T'] so generated has a simpler algorithm).

Equation (1) applies independently to any type of elementary operation, additions, real and complex multiplications as well as any other specific operation (e.g. shift, multiplication by $\sqrt{2}$ . . . etc.)

Rule 3: Generalized Kronecker Product:

Given two sets of unitary matrices, set $\{\mathcal{A}\}$ of m matrices $\left[A^i\right]$ (i=0,...,m-1) all of order n and set $\{\mathcal{B}\}$ of n matrices $\left[B^i\right]$ (i=0,...,n-1), all of order m, we define the generalized Kronecker product of the sets $\{\mathcal{A}\}$ and $\{\mathcal{B}\}$, denoted $\{\mathcal{A}\}\otimes\{\mathcal{B}\}$ to be the square matrix [C] of order (mn) such that

$$C_{i,j} = C_{um+w,\ u'm+w'} = A^{w}_{uu'} \cdot B^{u'}_{ww'} \qquad (2)$$

with $i = um+w$ $\quad u, u'=0, \ldots, n-1$

$j = u'm+w'$ $\quad w, w'=0, \ldots, m-1$

a) [C] is a unitary matrix:

Proof:

$$\sum_{k=0}^{mn-1} C_{ik}\ C^{*}_{jk} = \sum_{v=0}^{n-1} \sum_{z=0}^{m-1} C_{um+w,\ vm+z}\ C^{*}_{u'm+w',\ vm+z}$$

with $k = vm+z$ $\quad v=0, \ldots, n-1$

$z=0, \ldots, m-1$

Using (2)

$$\sum_{k=0}^{mn-1} C_{ik}\ C^{*}_{jk} = \sum_{v=0}^{n-1} \sum_{z=0}^{m-1} A^{w}_{uv}\ A^{*\,w'}_{u'v}\ B^{v}_{wz}\ B^{*\,v}_{w'z}$$

$$= \sum_{v=0}^{n-1} A^{w}_{uv}\ A^{*\,w'}_{u'v} \underbrace{\sum_{z=0}^{m-1} B^{v}_{wz}\ B^{*\,v}_{w'z}}_{\delta_{w\,w'},\ \text{by orthonormality of } \left[B^{v}\right]}$$

$$= \delta_{w,w'} \underbrace{\sum_{v=0}^{n-1} A^{w}_{uv}\ A^{*\,w}_{u'v}}_{\delta_{uu'}} \qquad \text{by orthonormality of } \left[A^{w}\right]$$

$$= \delta_{ww'}\ \delta_{uu'} = \delta_{ij}$$

where δ_{ij} is the Kronecker delta $\delta_{ij} = 1$ if $i=j$

$= 0$ otherwise.

This proves that [C] is a unitary matrix. QED

In the particular case in which the matrices $\left[A^{i}\right] = \left[A\right]$ are all identical, and also the matrices $\left[B^{i}\right] = \left[B\right]$, then the generalized Kronecker

product $\{\mathcal{A}\} \otimes \{\mathcal{B}\}$ reduces to the usual Kronecker product of matrices [14]: $[A] \otimes [B]$.

b) Factorization of [C]: We now prove that

$$[C] = [P^t]\ [\mathrm{Diag}\{\mathcal{A}\}]\ [P]\ [\mathrm{Diag}\{\mathcal{B}\}] \qquad (3)$$

where $[\mathrm{Diag}\{\mathcal{A}\}]$ and $[\mathrm{Diag}\{\mathcal{B}\}]$ are block diagonal matrices formed with the matrices of the sets $\{\mathcal{A}\}$ and $\{\mathcal{B}\}$ (see Fig.3-1) and [P] is the permutation matrix of order mn such that $P_{k\ell} = \delta_{vz'}\ \delta_{zv'}$ when $k = vn+z$, $\ell = v'm+z'$ and $z', v=0, \ldots, m-1; z, v'=0, \ldots, n-1$. Equation (3) is a generalization of the factorization of a simple Kronecker product into Good matrices [14].

Proof:

$$[\mathrm{Diag}\{\mathcal{A}\}]_{k'k} = \delta_{vv''}\ A^{v}_{z''z} \qquad \text{with} \quad k' = v''n+z''$$

$$[\mathrm{Diag}\{\mathcal{B}\}]_{\ell j} = \delta_{u'v'}\ B^{u'}_{z'w'} \qquad j = u'm + w'$$

$$[P^t]_{ik'} = \delta_{uz''}\ \delta_{wv''} \qquad i = um + w$$

We evaluate an element of the matrix on the right hand side of (3)

$$\sum_{k'=0}^{mn-1} \sum_{k=0}^{mn-1} \sum_{\ell=0}^{mn-1} [P^t]_{ik'}\ [\mathrm{Diag}\{\mathcal{A}\}]_{k'k}\ [P]_{k\ell}\ [\mathrm{Diag}\{\mathcal{B}\}]_{\ell j} =$$

$$\sum_{v''=0}^{m-1} \sum_{z''=0}^{n-1} \sum_{v=0}^{m-1} \sum_{z=0}^{n-1} \sum_{v'=0}^{n-1} \sum_{z'=0}^{m-1} \delta_{uz''}\ \delta_{wv''}\ \delta_{vv''}\ A^{v}_{z''z}\ \delta_{vz'}\ \delta_{zv'}\ \delta_{u'v'}\ B^{u'}_{z'w'} =$$

$$A^{w}_{uu'}\ B^{u'}_{ww'} = C_{ij} \qquad \text{QED.}$$

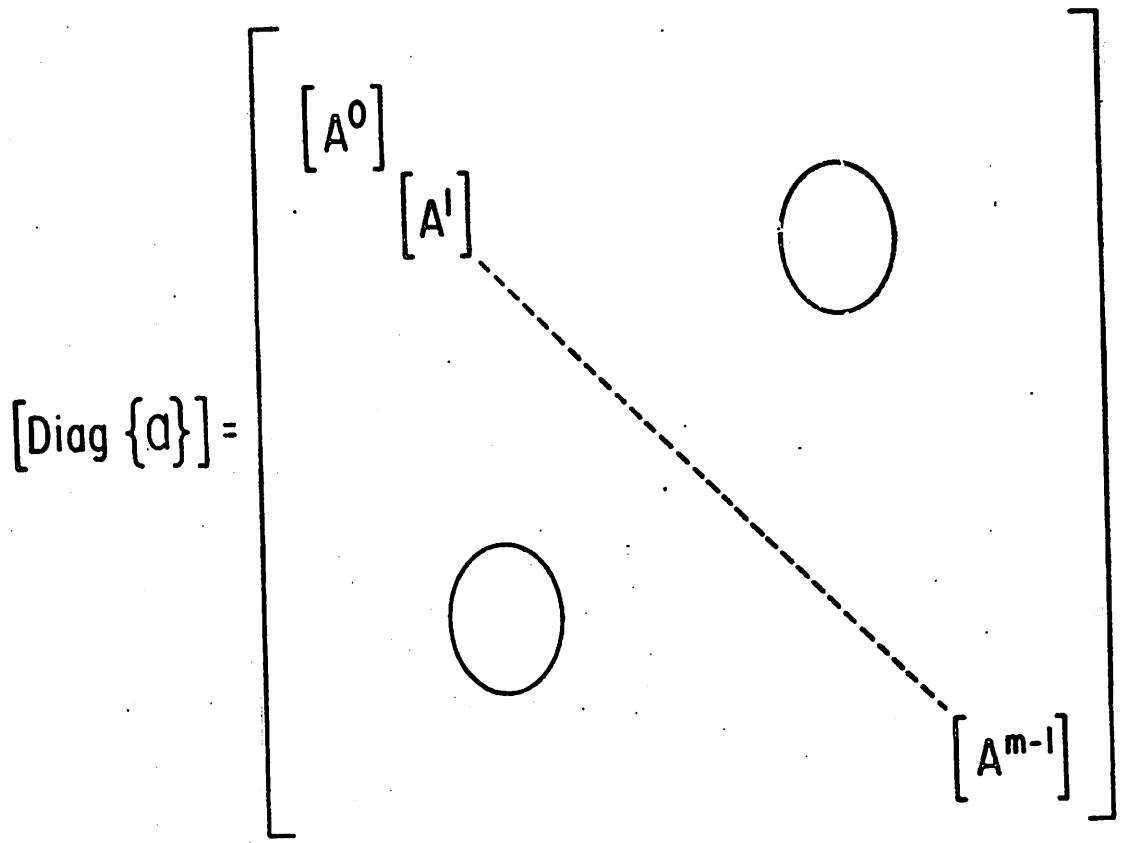


Fig. 3-1. Block diagonal Matrix

c) Number of elementary operations:

With the computational blocks corresponding to the transforms $A^0, \ldots, A^{m-1}$ and $B^0, \ldots, B^{n-1}$, the factorization of equation (3) leads directly to the computational block of the transform C given in Figure 3-2.

From the structure of the algorithm of Fig. 3-2 it is easy to see that if the matrices $[A^i]$ (i=0, ..., m-1) and $[B^u]$ (j=0,...,n-1) have algorithms requiring respectively p_n^i and q_m^j elementary operations of a specific type, their generalized Kronecker product [C] will require P_{mn} of these operations with

$$P_{mn} = \sum_{i=0}^{m-1} p_n^i + \sum_{j=0}^{n-1} q_m^j \qquad (4)$$

In the particular case of a simple Kronecker product $p_n^i = p_n$ and $q_m^j = q_m$ so

$$P_{mn} = m\, P_n + n\, q_m \qquad (5)$$

Note that the use of rule 1 and 2 only increases the number of elementary operations while the order of the generated transform does not change. For rule 3, even if $[A^i]$ and $[B^j]$ do not have fast algorithms and thus require n^2 and m^2 elementary operations, [C] requires a maximum of $(m+n)mn \leq (mn)^2$ (for m, n > 1) elementary operations.

The results of equations (1), (4) and (5) are important: for every transform generated with the recursive rules presented, they give a simple and systematic way to estimate its computational complexity.

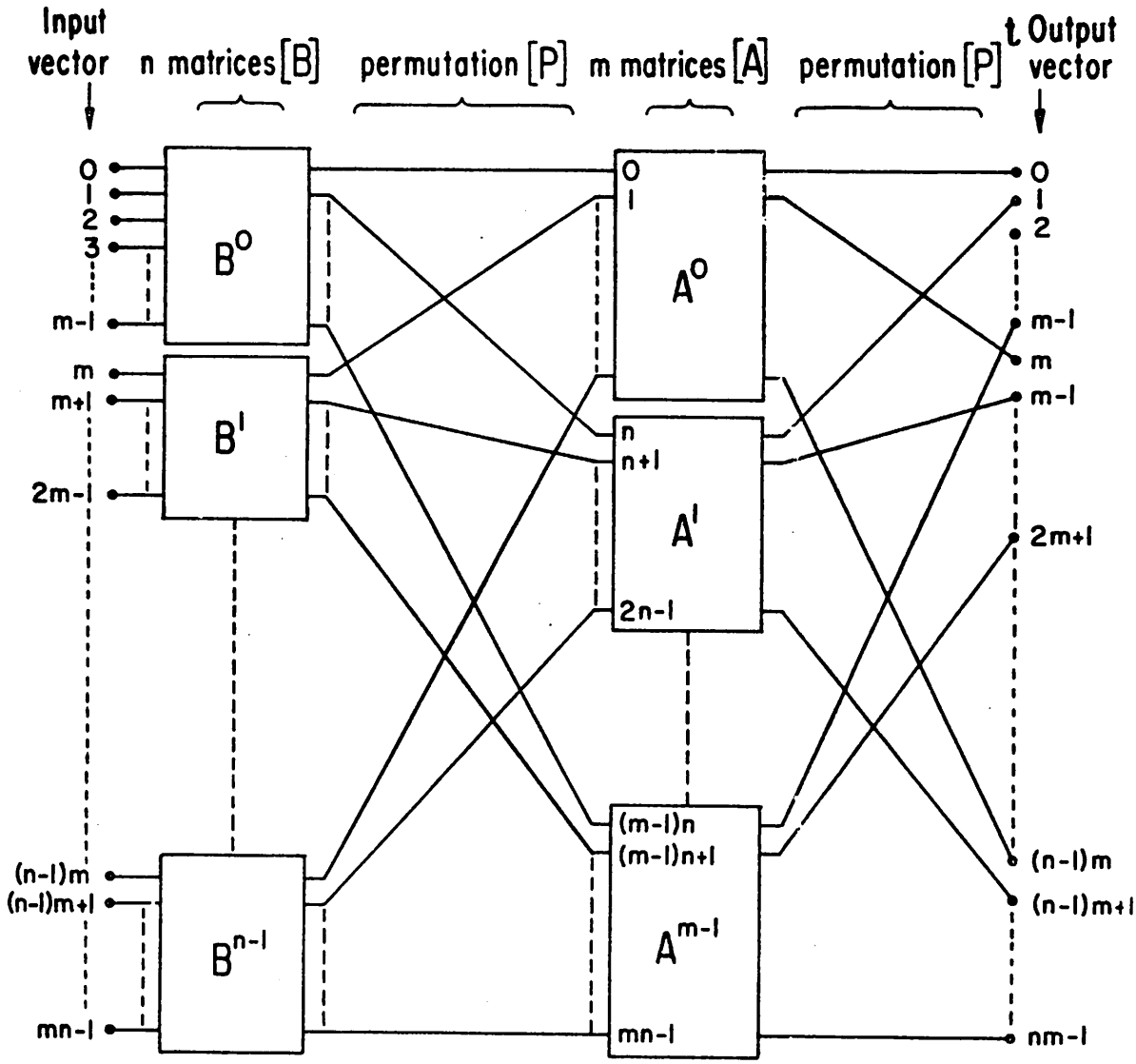


Fig. 3-2. Generalized Kronecker product : fast algorithm

3-3. Identical Computation (IC) Family:

The generative rules defined above create a unified framework for the known fast unitary transforms, introduce new transforms, and allow assessment of the computational complexity of such transforms. In this paper, one large family of transforms is considered: the "identical computation transforms" that we discuss now.

We denote by $\{\mathcal{A}\} \otimes [B_q]$ the generalized Kronecker product of a set $\{\mathcal{A}\}$ of q matrices $[A_p^k]$ (k=0, ..., q-1) of order p and a set $\{\mathcal{B}\}$ of p identical matrices $[B_q]$ of order q. $[B_q]$ will be called a core matrix and $[A_p^k]$ a parent matrix. The IC transforms are recursively generated from a unique class, $\mathcal{C}$, of parent matrices of some order f and an original core matrix $[\mathcal{O}]$ of order q. An IC transform of order (qf^n) is then obtained from the original core matrix $[\mathcal{O}]$ by the recursive formulas:

$$[IC_{qf}] = [D_{qf}]\ [\{\mathcal{A}\} \otimes [\mathcal{O}]]\ [D'_{qf}]$$

$$[IC_{qf^n}] = [D_{qf^n}]\ [\{\mathcal{A}_n\} \otimes [IC_{qf^{n-1}}]]\ [D'_{qf^n}] \qquad (6)$$

where the matrices [D] and [D'] express respectively a reordering followed by multiplications by roots of unity of the rows and the columns. All parent matrices of $\{\mathcal{A}_i\}$ .... $\{\mathcal{A}_n\}$ belong to $\mathcal{C}$.

The common characteristic of all the transforms of the IC family is that their algorithms only use in any computation intermediate results obtained from the input vector through identical computations (so the name of the family). This property provides a uniform treatment of successive components of the input vector if we consider that any parent matrix

treats uniformly its input vector. For this family, all the normalizations can be delayed to the last stage of computation.

We shall consider different choices for the original matrix $[\mathcal{O}]$, the class $\mathcal{C}$ of parent matrices, the matrices [D] and [D'] and the sets $\{\mathcal{A}_k\}$. We first show that the basic transforms, Fourier, Walsh-Hadamard and Haar, are IC transforms.

3-4. Basic Transforms: Fourier, Walsh-Hadamard, Haar:

In this section with the help of the generative rules, we examine the well known Fourier, W-H , and Haar transforms. This approach allows the derivation of some new results concerning the number of multiplications required by a FFT of composite order, a concise presentation of the different definitions of the W-H transform, and simple definitions of the Haar transform. In addition it makes apparent the common structure of these transforms. This will lead in the next section to the definition of families of transforms between the basic transforms.

In the following we emphasize specific orderings for the basic transforms: frequencies for the Fourier transform, zequencies[3] for the W-H transform and rank for the Haar transform. These orderings have proved to be useful in signal encoding because they concentrate the signal energy into the first transform coefficients, for some image models [21].

---

[3] This terminology has been introduced by Yuen [20]. The zequency is the number of zero crossings.

3-4-1. Generalized Fast Fourier Transform of Composite Order

a) decomposition theorem:

Given the Fourier matrices $[F_p]$ and $[F_q]$ of orders p and q respectively, the matrix $[F_{pq}]$ such that

$$[F_{pq}] = [\{[F_q^k]\} \otimes [F_p]]\ [P]^t \tag{7}$$

is the Fourier matrix of order pq. The set $\{[F_q^k]\}$, k=0, ... p-1 is such that

$$[F_q^k] = [F_q]\ [D_k] \text{ where} \tag{8}$$

$[D_k]$ is a diagonal matrix such that $(D_k)_{u'u'} = \exp(-2\pi j \frac{ku'}{pq})$

$[P]^t$ is the permutation matrix such that

$$P_{st} = \delta_{uz}\ \delta_{kw} \text{ with } s = uq + k \quad t = wp + z$$

$$u, z < p$$

$$k, w < q$$

Proof: We denote $[\{[F_q^k]\} \otimes [F_p]]$ by $[F'_{pq}]$.

$$(F'_{pq})_{ug+k,\ u'g+k'} = (F_p^k)_{uu'} \cdot (F_q)_{kk'}$$

$$= (F_p)_{uu'} \cdot e^{-2\pi j \frac{ku'}{pq}} \cdot (F_q)_{kk'} = \frac{1}{\sqrt{2^{pq}}}\, e^{-2\pi j (\frac{uu'}{p} + \frac{ku'}{pq} + \frac{kk'}{q})}$$

$$[F_{pq}] = [F'_{pq}] \cdot [P]^t \Rightarrow (F_{pq})_{uq+k,\ wp+z} = (F'_{pq})_{uq+k,\ u'q+k'} \cdot \delta_{zu'}\ \delta_{wk'}$$

$$= e^{-2\pi j\left(\frac{uz}{p} + \frac{kz}{pq} + \frac{kw}{q}\right)} = e^{-2\pi j \frac{(uq+k)\ (wp+z)}{pq}} \qquad \text{QED.}$$

Note that $[F_{pq}]$ is symmetric and orthonormal so that $([F_{pq}]^{-1})^* = [F_{pq}]$. Making use of (3) and (7) it is possible to derive a new expression for $[F_{pq}]$:

$$[F_{pq}] = [[F_p] \otimes \{[F'^{k}_{q}]\}]\ [P] \qquad (9)$$

$$\text{with } [F'^{k}_{2}] = [D_k]\ [F_q]$$

If $[F_p]$ and $[F_q]$ require respectively $\mathcal{A}_p$ and $\mathcal{A}_q$ complex additions, $\mathcal{M}_p$ and $\mathcal{M}_q$ complex multiplications, $[F_{pq}]$ will require by application of (4).

$$\mathcal{A}_{pq} = p\mathcal{A}_q + q\mathcal{A}_p \qquad \text{complex additions} \qquad (10)$$

$$\mathcal{M}_{pq} = p\mathcal{M}_q + q\mathcal{M}_p + C_{p,q} \qquad \text{complex multiplications} \qquad (11)$$

where $C_{p,q}$ is the number of complex multiplications introduced by (8).

$C_{p,q} = pq$ if all the factors including ± 1, $\pm j$ are considered.

$C_{p,q} = (p-1)\ (q-1)$ if the factors ± 1 are discarded

$C_{p,q} = (p-1)\ (q-1)-1$ if the factors $\pm j$ are also discarded (when (pq) is a power of 2).

b) Generalized FFT of composite order

If the order of the Fourier transform is composite, i.e. $N = \rho_1 \ldots \rho_n$, the previous decomposition theorem yields the well known FFT

algorithms [2] [3] detailed by Glassman [22] in the most general case. The recursive use of the formulae (10) and (11) gives the number of required operations. In the case of $N = r^n$ we can solve these recursive equations: this is the case of FFT of radix r.

$$\mathcal{A}_{r^n} = r^{n-1}\mathcal{A}_r + r\,\mathcal{A}_{r^{n-1}} \text{ or } \mathcal{A}_{r^n} = nr^{n-1}\mathcal{A}_r$$

$$\mathcal{M}_{r^n} = r^{n-1}\mathcal{M}_r + r\mathcal{M}_{r^{n-1}} + (r-\alpha)(r^{n-1}-\alpha) - \beta \text{ or} \tag{12}$$

$$\mathcal{M}_{r^n} = n\,r^{n-1}\mathcal{M}_r + (r-\alpha)\left[(n-1)\,r^{n-1} - \alpha\frac{(r^{n-1}-1)}{r-1}\right] - \beta\,\frac{(r^{n-1}-1)}{r-1} \tag{13}$$

(α, β depend on the value of $C_{p,q}$)

The radices 2, 4, 8 and 16 have been considered in the literature.

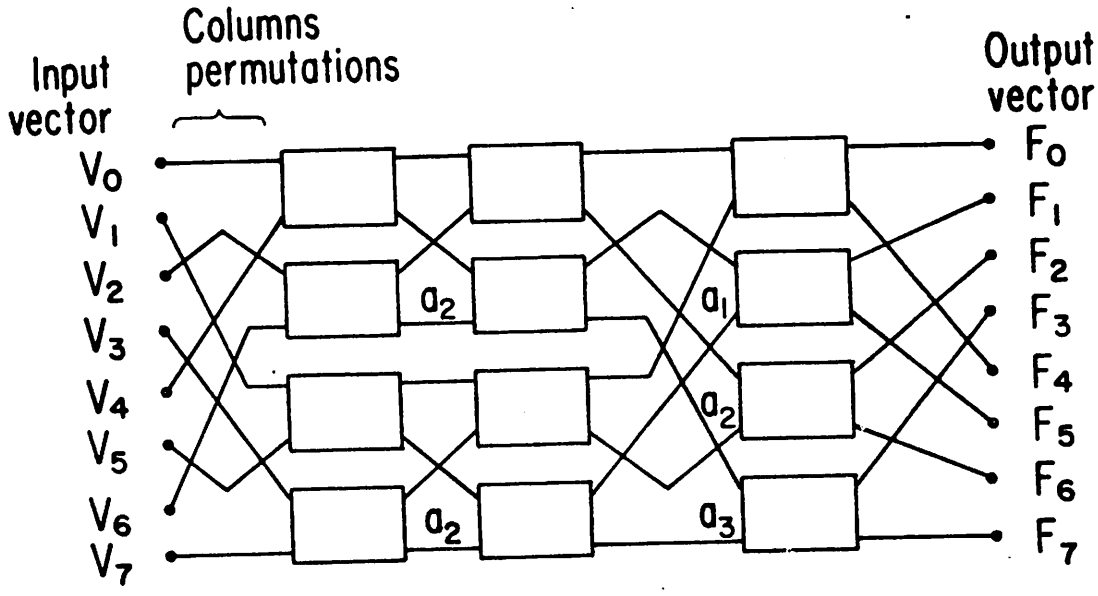
For the radix 2, which gives the most popular FFT, the recursive relations given by the decomposition theorem are

$$[F_{2^n}] = [\{[F_2^k]\} \circledast [F_{2^{n-1}}]]\,[P]^t \tag{14}$$

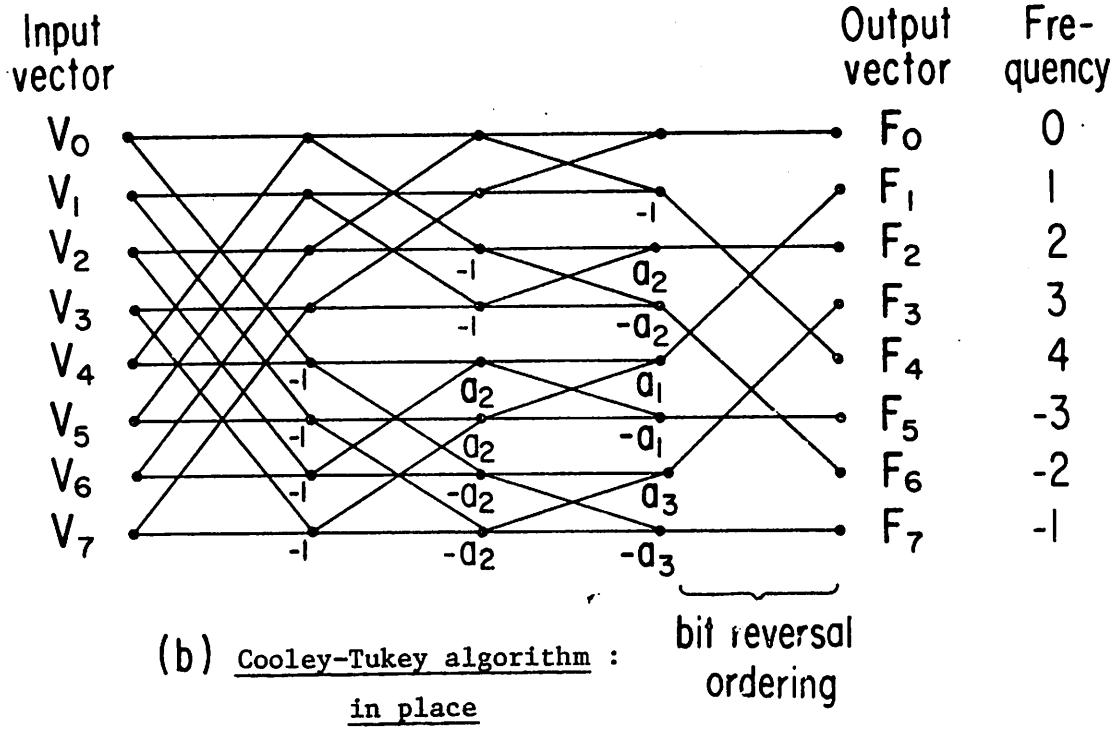
and $[F_{2^n}] = [[F_{2^{n-1}}] \circledast \{[F_2'^k]\}]\,[P]$

$$\text{with } [F_2^k] = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & e^{-2\pi j \frac{k}{2^n}} \\ 1 & -e^{-2\pi j \frac{k}{2^n}} \end{bmatrix} \text{ and } [F_2'^k] = \frac{2}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ e^{-2\pi j \frac{k}{2^n}} & -e^{-2\pi j \frac{k}{2^n}} \end{bmatrix} \tag{15}$$

The algorithm corresponding to the recursive formula (14) and obtained by recursive use of Fig. 3-2 is shown in Fig. 3-3a ; it can be arranged equivalently with all operations "in place" as shown in Fig. 3-3b



(a) Cooley-Tukey algorithm deduced from Fig. 3-2



(b) Cooley-Tukey algorithm : in place

Fig. 3-3. Fast Fourier Transform of order 8, radix 2

which is the classical diagram of the Cooley-Tukey [2] algorithm with decimation in time.

The algorithm corresponding to the formula (15) is the Sande-Tukey [3] algorithm with decimation in frequency and is shown in Fig. 3-3 c and d.

For these Figures the factors are

$$a_o = 1$$

$$a_1 = \exp(-2\pi j/8)$$

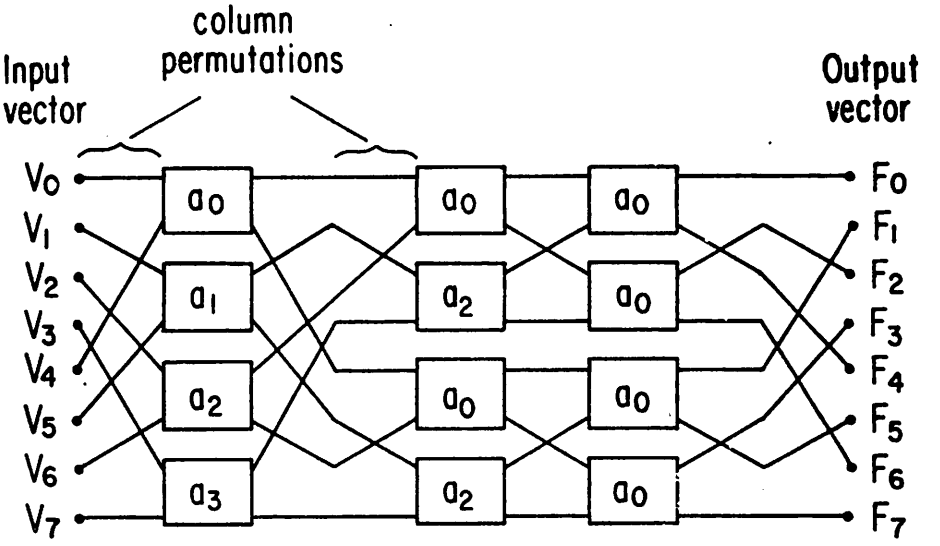
$$a_2 = \exp(-4\pi j/8) = -j$$

$$a_3 = \exp(-6\pi j/8)$$

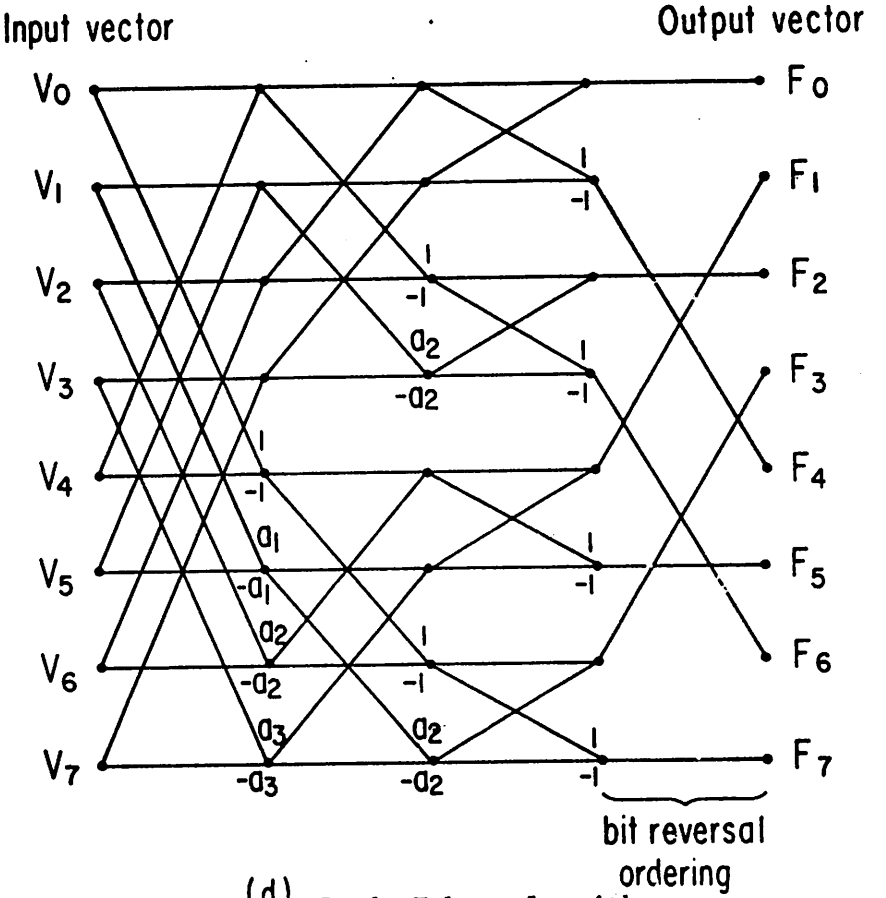
We can compare the FFT with radices 2, 4, 8 and 16 for transforms of order $N = 2^n = r^{\frac{n}{\log_2 r}}$ (n is then a multiple of 12). The formulas (12) and (13) then give:

Radix	$\mathcal{A}_r$	$\mathcal{M}_r$	$\mathcal{A}_{r^n}$	$\mathcal{M}^1_{r^n}$ (all factors)	$\mathcal{M}^2_{r^n}$ (no factors ± 1)	$\mathcal{M}^3_{r^n}$ (no factors $\pm 1 \pm j$)
2	2	0	$n2^n$	$(n-1)\,2^n$	$n2^{n-1} - 2^n + 1$	$n2^{n-1} - 3.2^{n-1} + 2$
4	8	0		$(\frac{n}{2}-1)\,2^n$	$3n2^{n-3} - 2^n + 1$	$3n2^{n-3} - \frac{13.2^{n-2}-4}{3}$
8	24	1		$(\frac{3n}{8}-1)\,2^n$	$\frac{n2^n}{3} - 2^n + 1$	$\frac{n2^n}{3} - \frac{57.2^{n-3}-8}{7}$
16	64	6		$(\frac{11n}{32}-1)\,2^n$	$\frac{21n2^n}{64} - 2^n + 1$	$\frac{21n2^n}{64} - \frac{241.2^{n-4}-16}{15}$

The column $\mathcal{M}^2_{r^n}$ has been given by Singleton [23]. In fact our approach allows the evaluation of the computational complexity for any composite order, in particular for mixed radix FFT.



(c) Sande-Tukey algorithm deduced from Fig. 3-2



(d) Sande-Tukey algorithm : in place

Fig. 3-3. Fast Fourier Transform of order 8, radix 2

The factors ± 1 are easy to track in the algorithms and for most realizations multiplications by ± 1 are not performed. The factors $\pm j$ appear in various places in the algorithms and in most realizations multiplications by $\pm j$ are performed; however, in an error analysis these multiplications do not introduce any rounding error and the column $\mathcal{U}^3_{r^n}$ is then of interest.

3-4-2 Walsh-Hadamard Transform:

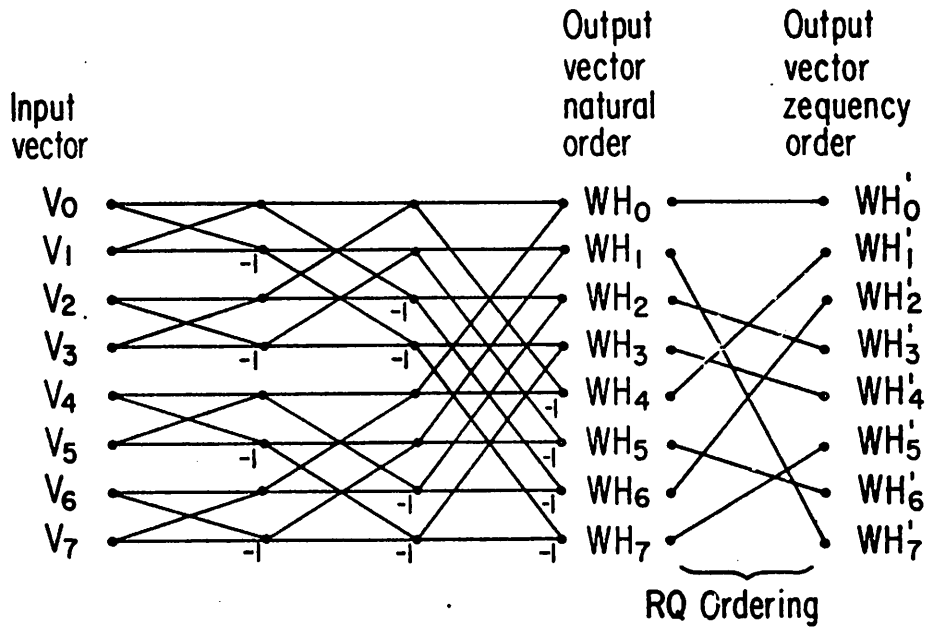
The W-H functions are well known and the results presented in this section are explicit or implicit in many publications. Here, we wish to express these results in terms of our generative rules; we think that the following compact notation clarifies the relations between the various orderings of the W-H functions and the different algorithms. This approach also makes apparent the common structure of the W-H transform with the Fourier and Haar transforms.

Three distinct orderings or rows of the W-H transform are commonly used and are of interest (see the discussion by Yuen [20]). For each of these orderings there exists a recursive matrix definition:

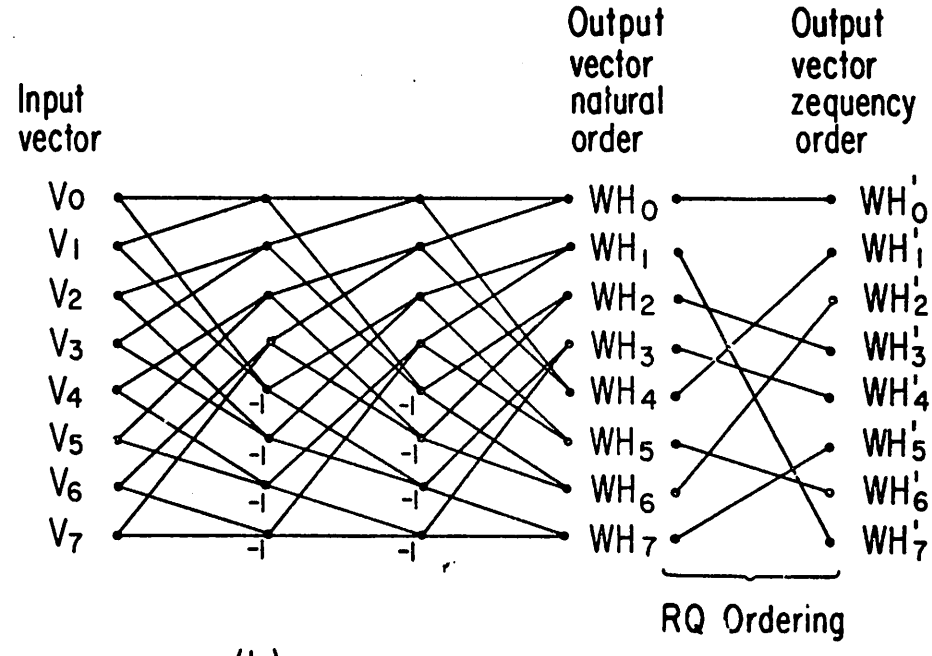
a) "natural order" It is obtained by simple Kronecker product without any permutation

$$[WH_{2^n}\ \text{nat.}] = [F_2] \otimes [WH_{2^{n-1}}\ \text{nat.}] \tag{16}$$

with the original core matrix $[F_2]$. This relation gives directly, by recursive use of Fig. 3-2, the fast algorithm of Fig. 3-4a (without the reorderings). A different presentation of this algorithm with identical stages of computation is given in Fig. 3-4b.



(a) Algorithm with rows in natural order



(b) Algorithm with identical stages

Fig. 3-4. Walsh-Hadamard transform of order 8

b) Paley's ordering: Used originally by Paley [24] it seems more suitable for mathematical developments than the other orderings. The recursive relations introduced by Yuen [20] are expressed by the matrix relation

$$[WH_{2^n}\ \text{pal.}] = [[F_2] \otimes [WH_{2^{n-1}}\ \text{pal.}]]\ [P]^t \qquad (17)$$

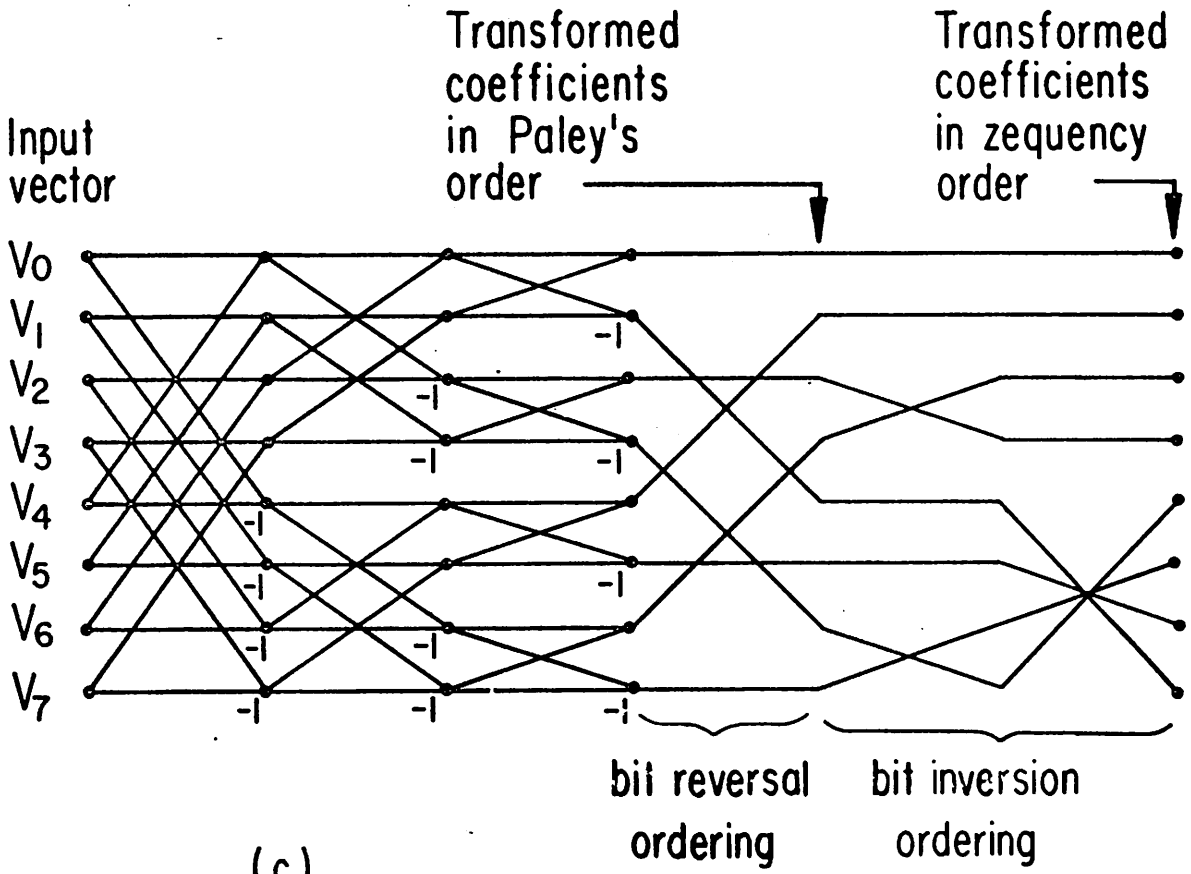
with the original core matrix $[F_2]$ and $[P]$ as defined previously for the Fourier transform (see section 3.3). This relation gives the algorithm of Fig. 3-4 c (without the bit-inversion reordering) by recursive use of Fig.3-2. A different presentation of this algorithm has been given in chapter 2.

c) zequency ordering: This is the original ordering by Walsh [25] and is the ordering of interest for signal encoding because it ranks the transform coefficients roughly according to their variances for signal statistics commonly encountered in practice. The generating process $\mathcal{W}$, of chapter 2, defines recursively the W-H matrices in zequency order. We can express it by the matrix relation

$$[WH_{2^n}\ \text{zeq.}] = [W]\ [[F_2] \otimes [WH_{2^{n-1}}\ \text{zeq.}]] \qquad (18)$$

where [W] denotes the reordering of the process $\mathcal{W}$. A zequency ordered algorithm has been investigated by Manz [26].

Although the zequency ordering could be generated recursively, the corresponding algorithm would not be simple and it may be preferable to obtain the W-H transform in natural or Paley's order and then perform a global reordering.



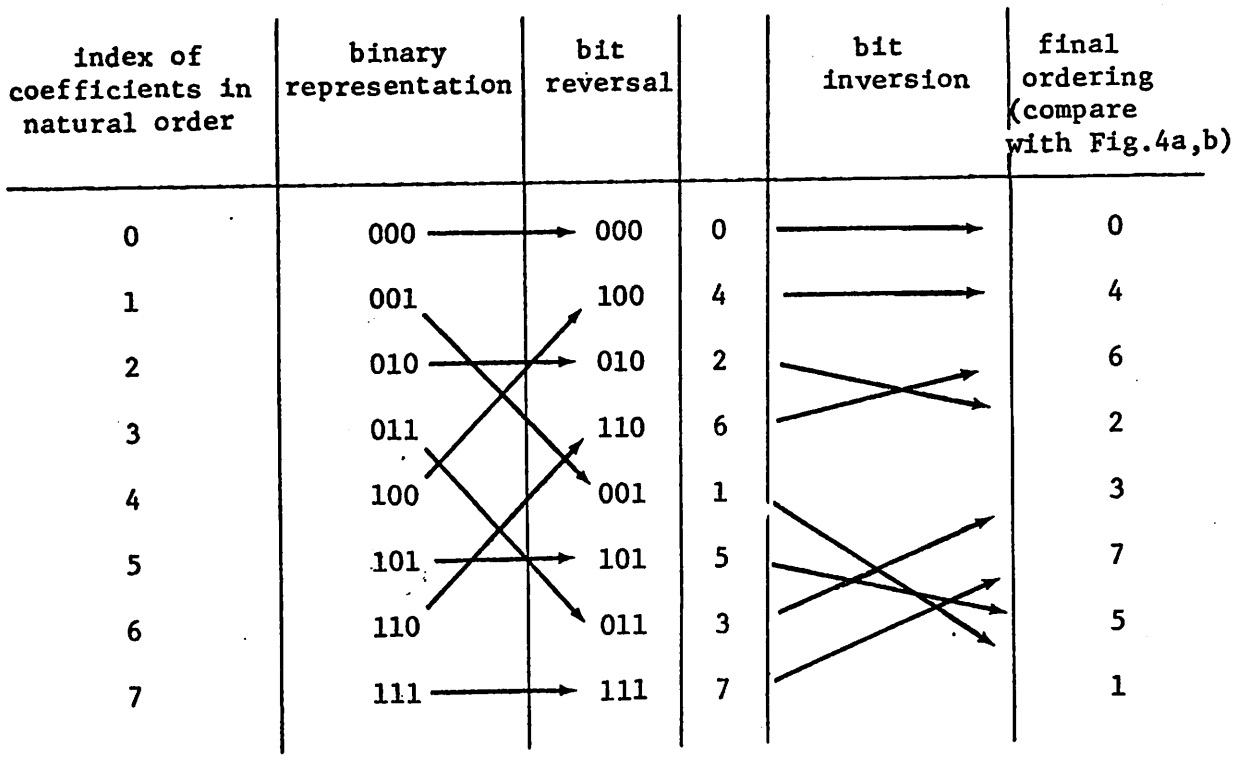
Algorithm with rows in Paley's order

Fig. 3-4. Walsh-Hadamard transform of order 8

Given the transform coefficients in Paley's order a "bit inversion" reordering, denoted in matrix form by [Q], is necessary to put them in zequency order: in a bit-inversion permutation, consecutive coefficients with k^{th} bit of the binary representation of their indexes equal to 1 are put in reverse order. This operation is performed for all bits starting from the rightmost bit. With 8 coefficients to reorder this means the following permutation:

index of coefficients in Paley's order	binary representation	reverse order for middle bit	reverse order for leftmost bit	final order (see Fig. 4c)
0	000	000	000	0
1	001	001	001	1
2	010	011	011	3
3	011	010	010	2
4	100	100	110	6
5	101	101	111	7
6	110	111	101	5
7	111	110	100	4

Given the transform coefficients in natural order, a bit reversal ordering, denoted by the matrix [R], will order them in Paley's order and a bit-inversion will order them in zequency order. For 8 coefficients to reorder we have:



It is important to note that, as the W-H matrices are symmetric in any of the three orderings, these reorderings can be performed on the columns as well as on the rows. Hence we have the matrix relations:

$$[WH_{2^n}\ \text{zeq.}] = [Q]\ [WH_{2^n}\ \text{pal.}] = [WH_{2^n}\ \text{pal.}]\ [Q]^t$$

$$[WH_{2^n}\ \text{zeq.}] = [Q]\ [R]\ [WH_{2^n}\ \text{nat.}] = [WH_{2^n}\ \text{nat.}]\ [R]^t\ [Q]^t$$

Since the W-H matrices are their own inverses, we have also using (16) and (17) the following recursive relations

$$[WH_{2^n}\ \text{nat.}] = [P]^t\ [[WH_{2^{n-1}}\ \text{nat.}] \circledast [F_2]]\ [P]$$

$$[WH_{2^n}\ \text{pal.}] = [[WH_{2^{n-1}}\ \text{pal.}] \circledast [F_2]]\ [P]$$

These relations however do not give different algorithms. All these

algorithms differ only by reordering and so have the same number of additions given by

$\mathcal{A}_{2^n} = 2 \cdot \mathcal{A}_{2^{n-1}} + 2,\ 2^{n-1}$ with $\mathcal{A}_2 = 2$ which gives

$\mathcal{A}_{2^n} = n2^n$, a well known result.

3-4-3. Haar transform:

The Haar transform is usually defined from the Haar functions [11]. The Haar matrix of order 8 $[H_8]$ ordered by ranks is as follows

$$[H_8] = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}$$

Zones: 0, 1, 2 (rows 3–4), 3 (rows 5–8)

Here we use the generative rules to define recursively the Haar matrices and we have found two definitions:

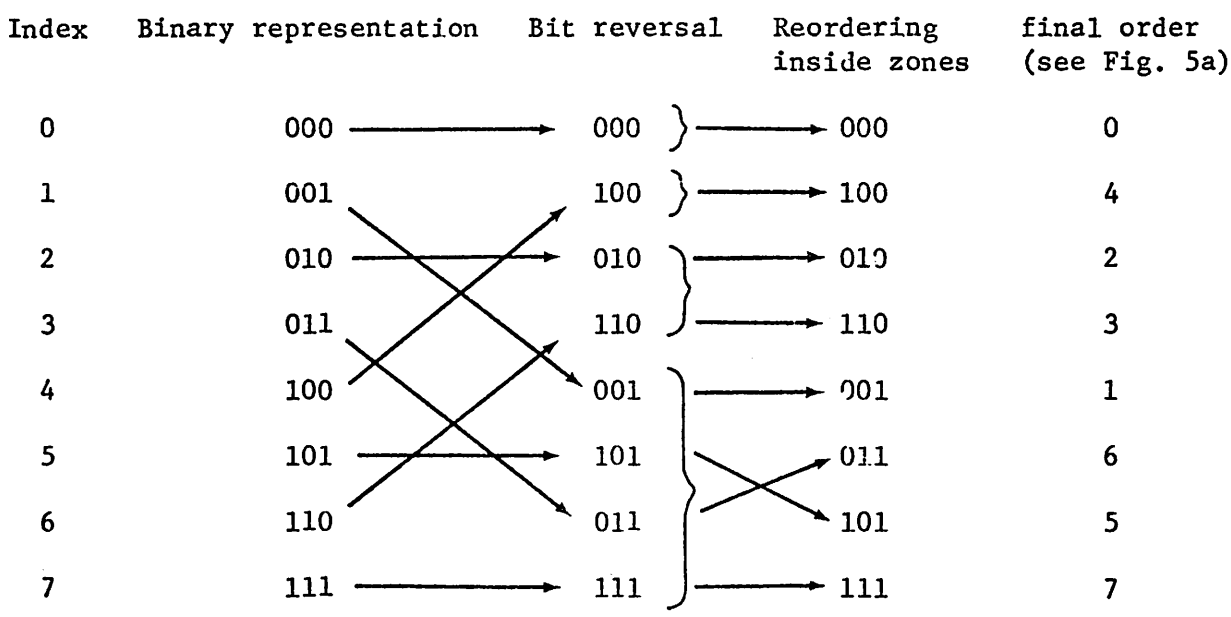
1) The Haar matrix of order 2^n is obtained from the Haar matrix of order 2^{n-1} by simple Kronecker product with $[I_2] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ followed by rotation of the rows 0 and 2^n by $[I_2]$. This is the process $\mathcal{H}$ of

chapter 2 in terms of generative rules.

2) The Haar matrices are recursively defined by the relation:

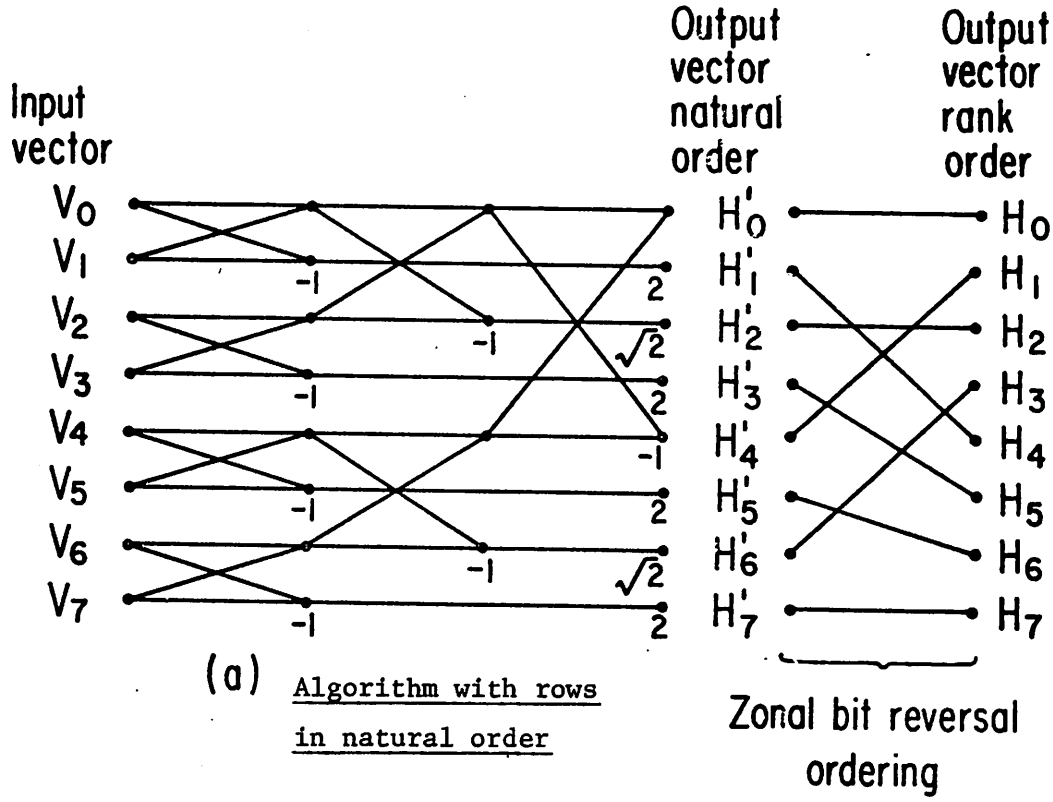
$$[H_{2^n} \text{ nat.}] \{[F_2], [I_2], \ldots., [I_2]\} \circledast [H_{2^{n-1}} \text{ nat.}] \qquad (19)$$

The rows are obtained in "natural" order. To reorder them by their ranks, we need a "zonal bit reversal" ordering. A zone as defined in chapter 2, is a set of coefficients with indexes between two successive powers of 2. A "zonal bit reversal" ordering is a bit-reversal followed by a reordering in the original order inside each zone. For 8 coefficients the zonal bit reversal ordering gives:

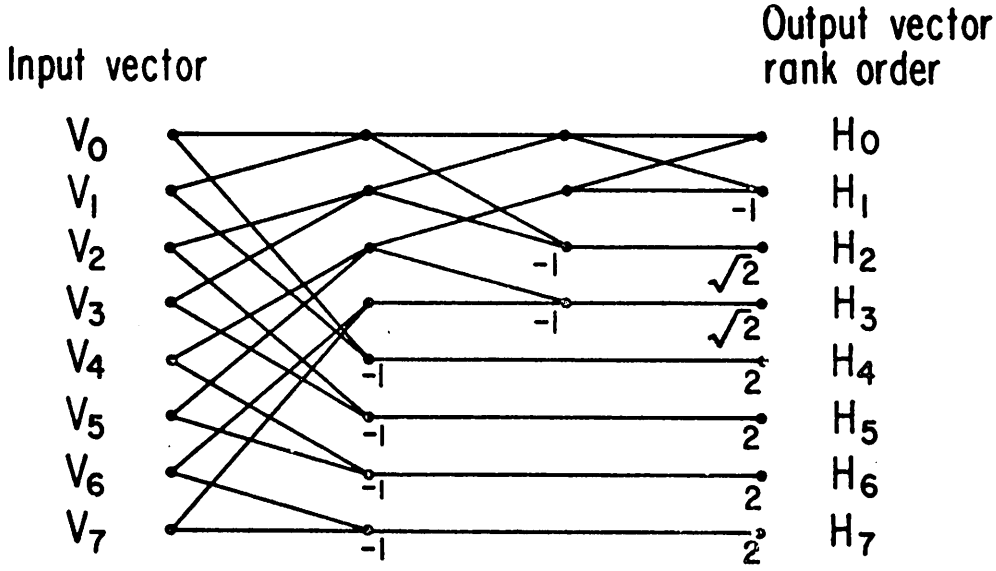


With both definitions, we obtain by recursive application of the diagram of Fig. 3-2 the algorithm of Fig. 3-5 a. This algorithm can be more conveniently organized as shown in Fig. 3-5 b and give the rows directly ordered by their rank.

By application of (4) we obtain the following recursive formula for the number of additions:



(a) Algorithm with rows in natural order



(b) Algorithm with rows in rank order

Fig. 3-5. Fast Haar transform of order 8

$\mathcal{A}_{2^n} = 2 \cdot \mathcal{A}_{2^{n-1}} + 2$. Hence $\mathcal{A}_{2^n} = 2(2^n-1)$ with $\mathcal{A}_2 = 2$. 2^{n-1} normalizations are also required. A modified Haar transform obtained from the Haar transform by permutation of its columns is related to the Fourier transform (see section 3-7-3): it is defined recursively by:

$$[MH_{2^n}] = [Z]\ [\{[F_2], [I_2], \ldots, [I_2]\} \otimes [MH_{2^{n-1}}]]\ [P]^t \qquad (20)$$

Globally the permutations [Z] perform a bit-reversal ordering inside each zone.

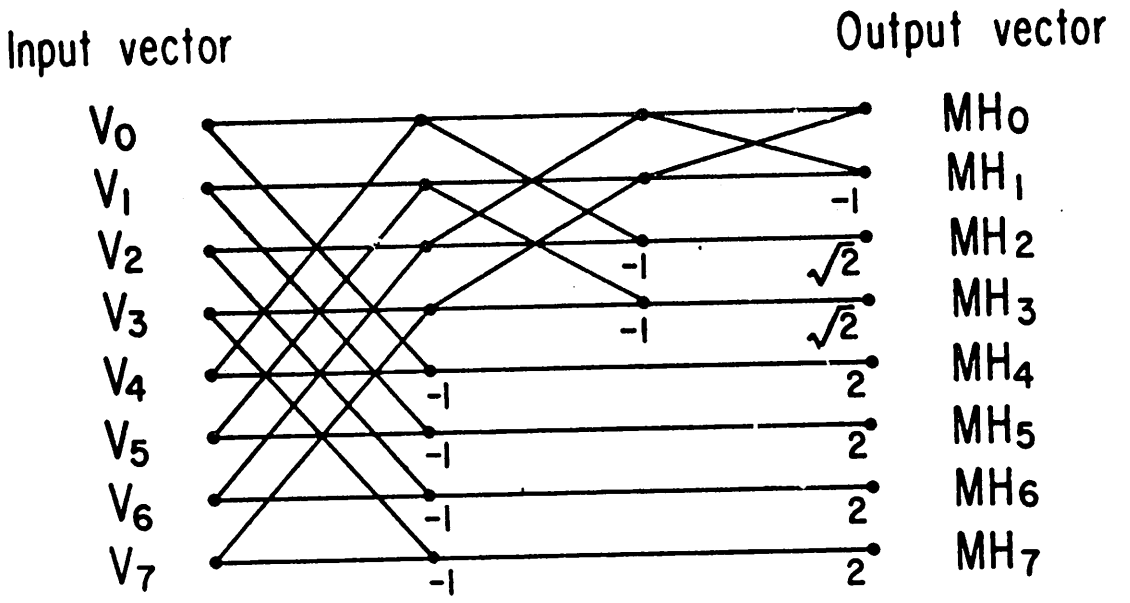
The modified Haar matrix of order 8 is as follows

$$[MH_8] = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} & 0 & +\sqrt{2} & 0 & -\sqrt{2} & 0 \\ 0 & \sqrt{2} & 0 & -\sqrt{2} & 0 & +\sqrt{2} & 0 & -\sqrt{2} \\ 2 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & -2 \end{bmatrix}$$

and its algorithm is given in Fig. 3-5 c.

3-5. Generalizations of the basic transforms

For the three basic transforms we have found recursive definitions with a matrix formula similar to (6). The direct comparison of these definitions suggests the generalization of the basic transforms to families between them. The simplest generalizations are two families between the



(c) Modified Haar transform

Fig. 3-5. Fast Haar transform of order 8

Fourier and W-H transforms, and between the W-H and Haar transforms. A larger generalization is the IC_2 family which includes all three basic transforms with parent matrices of order 2. A number of these generalized transforms have recently been discussed independently. We would like to show that they fall easily within the framework we have developed and that further generalizations are clearly possible.

3-5-1. Family between W-H and F:

If we compare the recursive generation of the Fourier matrices with radix 2 (14) and the W-H transform (17), we notice that they differ only by a set of factors. If we exclude the reordering of the rows, we see that we can easily generalize the W-H and Fourier transforms to a large family of unitary transforms given by the recursive formula

$$[GT_{2^n}] = \{[F_2(\theta_0),..,F_2(\theta_{2^{n-1}-1})]\} \otimes [GT_{2^{n-1}}]\,[P]^t$$

where $[F_2(\theta)] = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & \exp(-j\theta) \\ 1 & -\exp(-j\theta) \end{bmatrix}$

This family includes the W-H and Fourier transforms for the appropriate choices of the parameters $\theta_0 \ldots \theta_{2^{n-1}-1}$.

Two families have appeared in the literature for special choices of these parameters:

1) $\theta_k = \frac{2\pi\, kc}{2^n}$ where c is a real scalar varying from 0 (W-H transform) to 1 (Fourier transform). The corresponding transform has been called "general spectral analyzer" [14] [15].

2) $\theta_k = \frac{2\pi k}{2^n}$ if k mod $(2^{n-1-g}) = 0$

where g is an integer varying from 0 (W-H transform) to n-1 (Fourier transform), and $\theta_k = 0$ otherwise. The corresponding transform is the "Generalized Discrete Transform" [17]; we will denote it $[GT^g_{2^n}]$. Fig. 3-6 shows the matrix $[GT^2_{16}]$ and its fast algorithm. Many other choices for these factors are obviously possible and these two special choices do not seem to bear any exceptional importance.

As an example of use of our general formulas we compute now the required number of multiplications for the Generalized Discrete Transform. There are 2^g-1 factors different from 1.

$$\text{So } \mathcal{M}^g_{2^n} = 2\mathcal{M}^g_{2^{n-1}} + 2^g - 1 \qquad \text{for } n - 1 > g$$

$$\mathcal{M}^g_{2^n} = 2\mathcal{M}^g_{2^{n-1}} + 2^{n-1} - 1 \qquad \text{otherwise.}$$

$$\text{So that } \mathcal{M}^g_{2^n} = g\,2^{n-1} - 2^g + 1$$

If we do not count the multiplications by $\pm j$, we find similarly :

$$\mathcal{M}^g_{2^n} = (g-1)\,2^{n-1} - 2^g + 2 \qquad \text{Both results are new.}$$

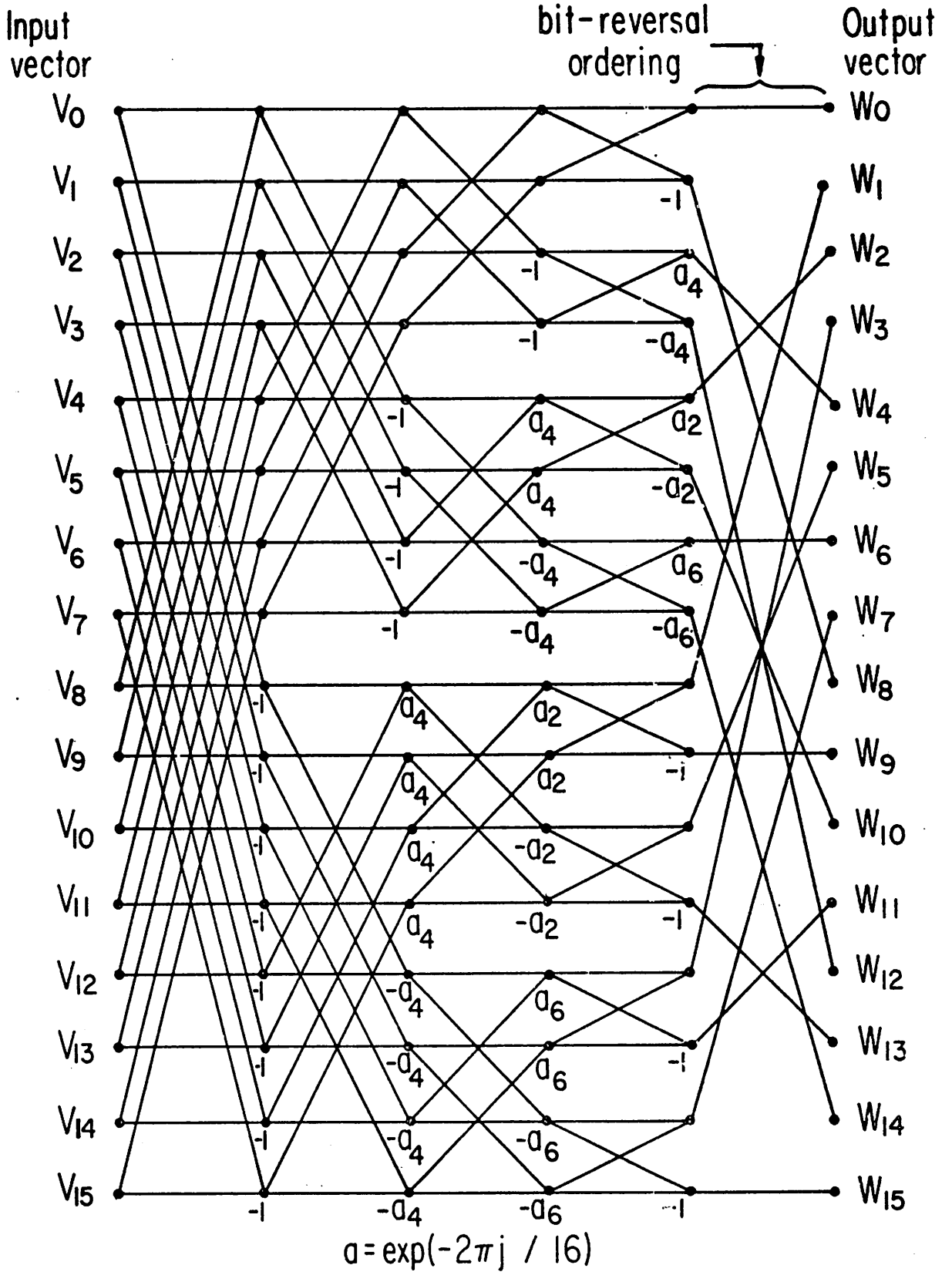
3-5-2 Family between Haar and W-H

In chapter 2, we have presented a family of transforms between the Haar and W-H transforms. This family was obtained by replacing W-H transforms of lower order by Haar transforms in the decomposition of the fast algorithm of a W-H transform. Now we have further decomposed the fast

$$\frac{1}{\sqrt{16}}\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & a^2 & a^2 & a^4 & a^4 & a^6 & a^6 & -1 & -1 & -a^2 & -a^2 & -a^4 & a^4 & -a^6 & -a^6 \\
1 & a^2 & a^4 & a^6 & -1 & -a^2 & -a^4 & -a^6 & 1 & a^2 & a^4 & a^6 & -1 & -a^2 & -a^4 & -a^6 \\
1 & 1 & a^6 & a^6 & -a^4 & -a^4 & a^2 & a^2 & -1 & -1 & -a^6 & -a^6 & a^4 & a^4 & -a^2 & -a^2 \\
1 & a^4 & -1 & -a^4 & 1 & a^4 & -1 & -a^4 & 1 & a^4 & -1 & -a^4 & 1 & a^4 & -1 & -a^4 \\
1 & 1 & -a^2 & -a^2 & a^4 & a^4 & -a^6 & -a^6 & -1 & -1 & a^2 & a^2 & -a^4 & -a^4 & a^6 & a^6 \\
1 & a^6 & -a^4 & a^2 & -1 & -a^6 & a^4 & -a^2 & 1 & a^6 & -a^4 & a^2 & -1 & -a^6 & a^4 & -a^2 \\
1 & 1 & -a^6 & -a^6 & -a^4 & -a^4 & -a^2 & -a^2 & -1 & -1 & a^6 & a^6 & a^4 & a^4 & a^2 & a^2 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & -1 & a^2 & -a^2 & a^4 & -a^4 & a^6 & -a^6 & -1 & 1 & -a^2 & a^2 & -a^4 & a^4 & -a^6 & a^6 \\
1 & -a^2 & a^4 & -a^6 & -1 & a^2 & -a^4 & a^6 & 1 & -a^2 & a^4 & -a^6 & -1 & a^2 & -a^4 & a^6 \\
1 & -1 & a^6 & -a^6 & -a^4 & a^4 & a^2 & -a^2 & -1 & 1 & -a^6 & a^6 & a^4 & -a^4 & -a^2 & a^2 \\
1 & -a^4 & -1 & a^4 & 1 & -a^4 & -1 & a^4 & 1 & -a^4 & -1 & a^4 & 1 & -a^4 & -1 & a^4 \\
1 & -1 & -a^2 & a^2 & a^4 & -a^4 & -a^6 & a^6 & -1 & +1 & a^2 & -a^2 & -a^4 & a^4 & a^6 & -a^6 \\
1 & -a^6 & -a^4 & -a^2 & -1 & a^6 & a^4 & a^2 & 1 & -a^6 & -a^4 & -a^2 & -1 & a^6 & a^4 & a^2 \\
1 & -1 & -a^6 & a^6 & -a^4 & a^4 & -a^2 & a^2 & -1 & 1 & a^6 & -a^6 & a^4 & -a^4 & a^2 & -a^2
\end{bmatrix}$$

a) Matrix

Fig. 3-6. Generalized Discrete transform $\left[GT_{16}^{2}\right]$



(b) Fast algorithm

Fig. 3-6. Generalized Discrete tranform $\left[GT_{16}^{2}\right]$

algorithms of the two transforms up to similar recursive formulas (16) and (19) or (17) and (20). Obviously, if we choose any of the 2^{n-1} parent matrices needed to generate the matrix of order 2^n to be either $[F_2]$ or $[I_2]$, we obtain a large family of unitary matrices which includes the Haar, W-H, and unity matrices. There are

$$2^{2^{n-1}} \cdot 2^{2^{n-2}} \cdot \ldots \cdot 2^1 = 2^{1+} \ldots + 2^{n-1} = 2^{2^n-1}$$

members of order 2^n in this family.

The number of additions is obviously twice the number of parent matrices equal to $[F_2]$. For the W-H transform we have $n2^{n-1}$ such matrices and therefore $n2^n$ additions. For the Haar transform we have $2^{n-1} + 2^{n-2} \ldots + 1 = 2^n-1$ such matrices $[F_2]$ and so $2(2^n-1)$ additions. The number of normalizations varies from 0(W-H) to 2^n-2 (the normalizing factors come by pairs and all pairs are different in the worst case). No multiplication is required during the computation. At the order 8, 2^7 matrices are in the family. We show in Fig. 3-7 one of them with its fast algorithm.

Assume as a particular case that we choose the parent matrices of the recursive formulas to be

$$[F^k] \quad k = 0, \ldots, 2^{p-1}-1$$

with

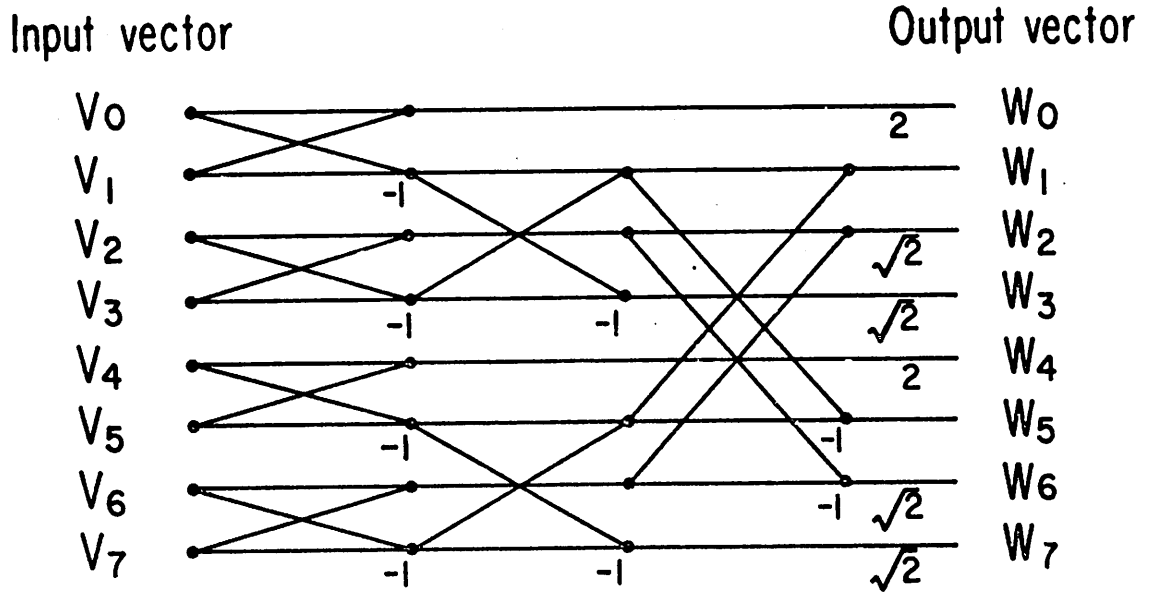
$$[F^k] = [F_2] \text{ for } k = 0 \bmod (2^{p+h-n}) \quad \text{or if } p \leq n - k$$

$$[F^k] = [I_2] \quad \text{otherwise}$$

where p is the stage of computation up to n when we generate a transform of

$$\frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \times 2 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \times \sqrt{2} \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & \times \sqrt{2} \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & \times 2 \\ 1 & -1 & 1 & -1 & -1 & +1 & -1 & +1 & \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 & -1 & \times \sqrt{2} \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & +1 & \times \sqrt{2} \end{bmatrix}$$

a) Matrix



(b) Fast algorithm

Fig. 3-7. Example of a transform of the family between the Haar and Walsh-Hadamard transforms

order 2^n and h an index lower than n.

Then, if the recursive formula used is similar to (16) and (19) we obtain a subclass of n transforms: for h = 0 we have the W-H transform and for h = n - 1 we have the Haar transform, both in natural order.

If the recursive formula is similar to (17) and (20) (with the permutation $[P]^t$ of the columns) we still obtain n transforms: for h = 0 we have the W-H transform in Paley's order and for h = n - 1 we have an unordered modified Haar transform. We denote these transforms $[WHH_2^k n]$. Fig. 3-8 shows $[WHH_{16}^2]$ and its fast algorithm.

3-5-3 IC_2 family:

To generate the family between W-H and P we have introduced a set of factors into the recursive formula for the W-H transform. To generate the family between W-H and Haar, we have replaced some parent matrices by the identity matrix $[I_2]$ in the same recursive definition of the W-H transform. If we allow simultaneously both operations we generate a larger family that we call IC_2.

More formally if $[T_{2^{n-1}}]$ is a member of IC_2 of order 2^{n-1}, a member of order 2^n is given by

$$[T_{2^n}] = [D_1]\ [\{[C_0], \ldots, [C_{2^{n-1}-1}]\} \otimes [T_{2^{n-1}}]]\ [D_2]$$

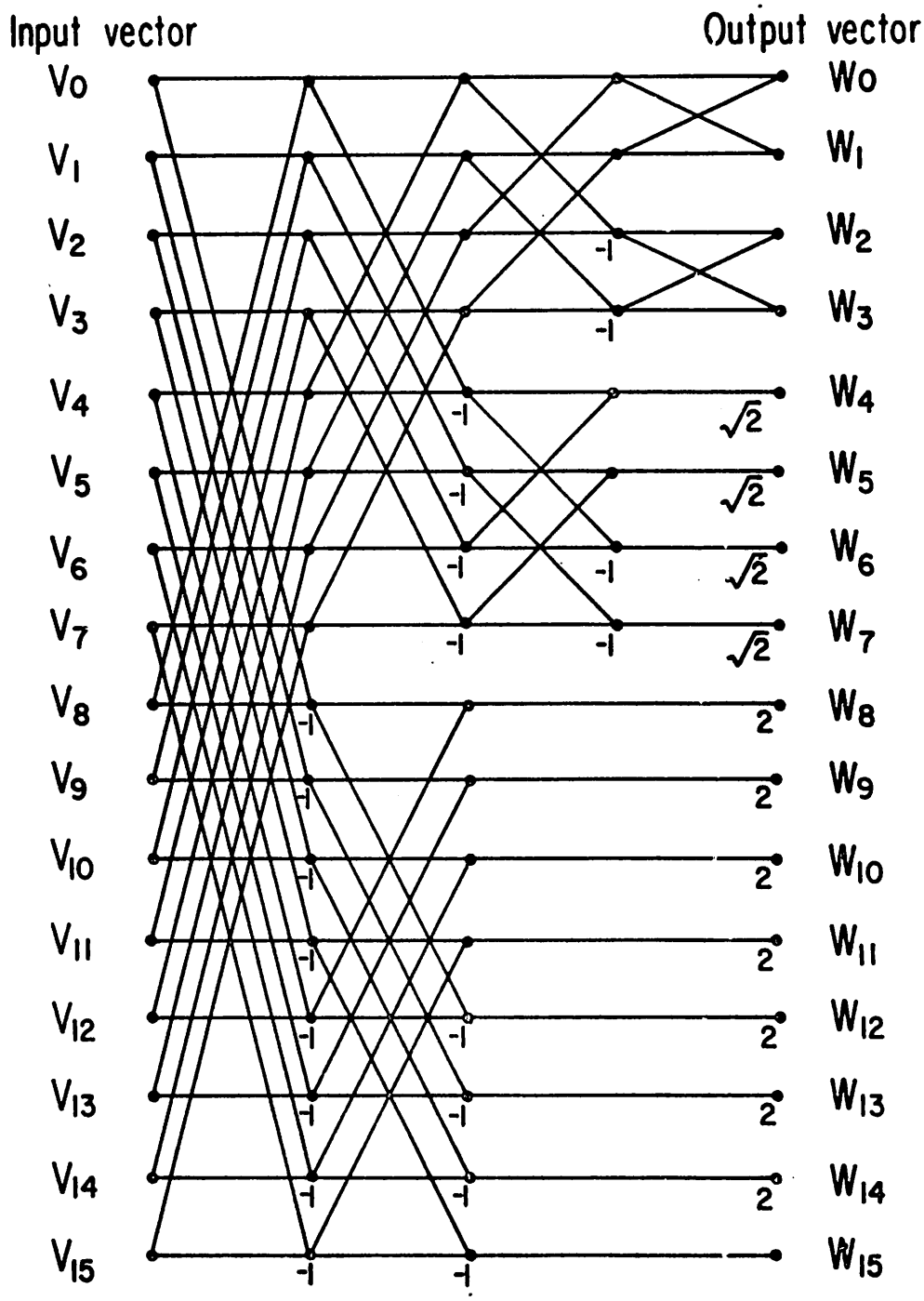
where $[D_1]$ and $[D_2]$ are permutation matrices and $C_0, \ldots, C_{2^{n-1}-1}$ are either

$$[I_2] \text{ or } [F_2(0)] = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & \exp(-j\theta) \\ 1 & -\exp(-j\theta) \end{bmatrix}$$

$$\frac{1}{\sqrt{16}}\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & +1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
\sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 \\
0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} \\
\sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 \\
0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} \\
2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 \\
2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2
\end{bmatrix}$$

a) matrix

Fig. 3-8. Generalized transform $\left[WHH_{16}^{2}\right]$



(b) Fast algorithm

Fig. 3-8. Generalized transform $\left[\mathrm{WHH}_{16}^{2}\right]$

Let us call this class of parent matrices C_2. For the order 2^n, $2^n-1(2^{n-1} + 2^{n-2} +\ldots+ 1 = 2^n-1)$ parent matrices have to be chosen independently in C_2: we say that the family IC_2 has 2^n-1 degrees of freedom over C_2 (see footnote 4).

The IC_2 family is very large and includes the families between W-H and F, W-H and Haar.

The number of required operations is given recursively by

additions: $\mathcal{A}_{2^n} = 2\ \mathcal{A}_{2^{n-1}} + 2A_n$

$$\text{Hence } \mathcal{A}_{2^n} = \sum_{k=1}^{n} 2^{n-k+1}\ A_k \tag{21}$$

multiplications: $\mathcal{M}_{2^n} = 2\mathcal{M}_{2^{n-1}} + L_n$

$$\text{hence } \mathcal{M}_{2^n} = \sum_{k=2}^{n} 2^{n-k}\ L_k \tag{22}$$

---

(4) This notion of degree of freedom is an extension of a concept introduced by Andrews and Caspari [16]. For them the degree of freedom of a class of matrices is the number of free parameters required to define this class. This definition is ambiguous when the constraints which define a class cannot be reduced to a set of free parameters. For example, the unitary matrices of order 2 are given 1 degree of freedom in [16] when in fact, on the real numbers, the most general matrix is

$$\begin{bmatrix} \cos\alpha & \sin\alpha \\ \varepsilon \sin\alpha & -\varepsilon\cos\alpha \end{bmatrix} \qquad \varepsilon = \pm 1 \qquad \alpha \in [0,2\pi]$$

and on the complex numbers the general solution depends on 4 angles $\in [0,2\pi]$ and 2 binary choices. Our approach is to track as far as possible the reduction to independent choices. If it can be reduced to a number of free parameters our degree of freedom will be the number of these parameters. Note that the relations (1) and (4) apply also to the recursive computation of the degree of freedom of a class. Note also that the degree of freedom has generally no relation with the computational complexity (which varies usually for the transforms of a class).

where A_k is the number of parent matrices different from $[I_2]$ at this k^{th} stage, and L_k the number of factors different from ± 1 (and maybe $\pm j$) at this stage.

For Haar $\quad A_k = 1$ for any k and $\mathcal{A}_{2^n} = 2(2^n-1)$

For W-H $\quad A_k = 2^{k-1}$ and $\mathcal{A}_{2^n} = n2^n$

For Fourier $\quad \mathcal{A}_{2^n} = n2^n$ and $L_k = 2^{k-1}$, $2^{k-1}-1$ or $2^{k-2}-2$, which yield the results of section 2.3 (radix 2).

We present now an example of interest in the IC_2 family: a class of transforms which make a discrete transition between the 3 basic transforms and which we call therefore the WFH class.

Each transform of this class is indexed by two positive integer parameters h and g such that $h + g < n$ when 2^n is the order of the transform and is denoted $[WFH^{g,h}_{2^n}]$.

$[WFH_{2^n}^{g,k}]$ is obtained recursively as the Fourier transform of radix 2 (formula 14) but with the parent matrices $[P_2^k]$ k = 0, .., $2^{p-1}-1$ such that

$$[P_2^k] = [F_2(2\pi k/2^p)] \text{ for } k = 0 \pmod{2^{p-g-1}}$$

$$[P_2^k] = [F_2] \quad \text{for } k = 0 \pmod{2^{p+h-n}} \quad \text{or if } p \leq n - h$$

$$k \neq 0 \pmod{2^{p-g-1}}$$

$$[P_2^k] = [I_2] \text{ otherwise}$$

where p is the level of computation up to n.

We can represent then WFH transforms on a (g, h) plane as shown in Fig. 3-9.With appropriate permutation matrices, for h = 0 we have the n Generalized Discrete Transforms (see section 5-1), for g = 0 the n WHH transforms (see section 5-2). $WFH^{0,0}$ is the W-H transform, $WHH^{0,\,n-1}$ the Modified Haar transform and $WHH^{n-1,0}$ the Fourier transform. For h + g = n - 1 we have a set of n transforms in between the Fourier and Haar transforms which have been called the Modified Generalized Discrete Transforms and defined after much work in [18].

3-6. Other IC transforms:

Except for the Fourier transform, we have restricted ourselves so far to IC transforms obtained from original core matrix $[F_2]$ and parent matrices of order 2. The generative rules have given a unified approach to the usual unitary transforms. We now consider some examples with a different original core matrix and parent matrices of higher orders.

The matrices of order 2 are of practical interest for the fast algorithm as long as we perform the required operations (specially additions) with only two operands at a time. If fast additions involving, let us say, f operands, become available, the transforms with parent matrices of order f may be of interest.

Most of the recursive structures of the transforms presented in the previcus sections can be applied to parent matrices of higher orders than 2. We now give some examples:

a) different original core matrix

In the definition of the W-H transform the original core matrix $[F_2]$ may be replaced by the core matrix $\begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}$ and we obtain a

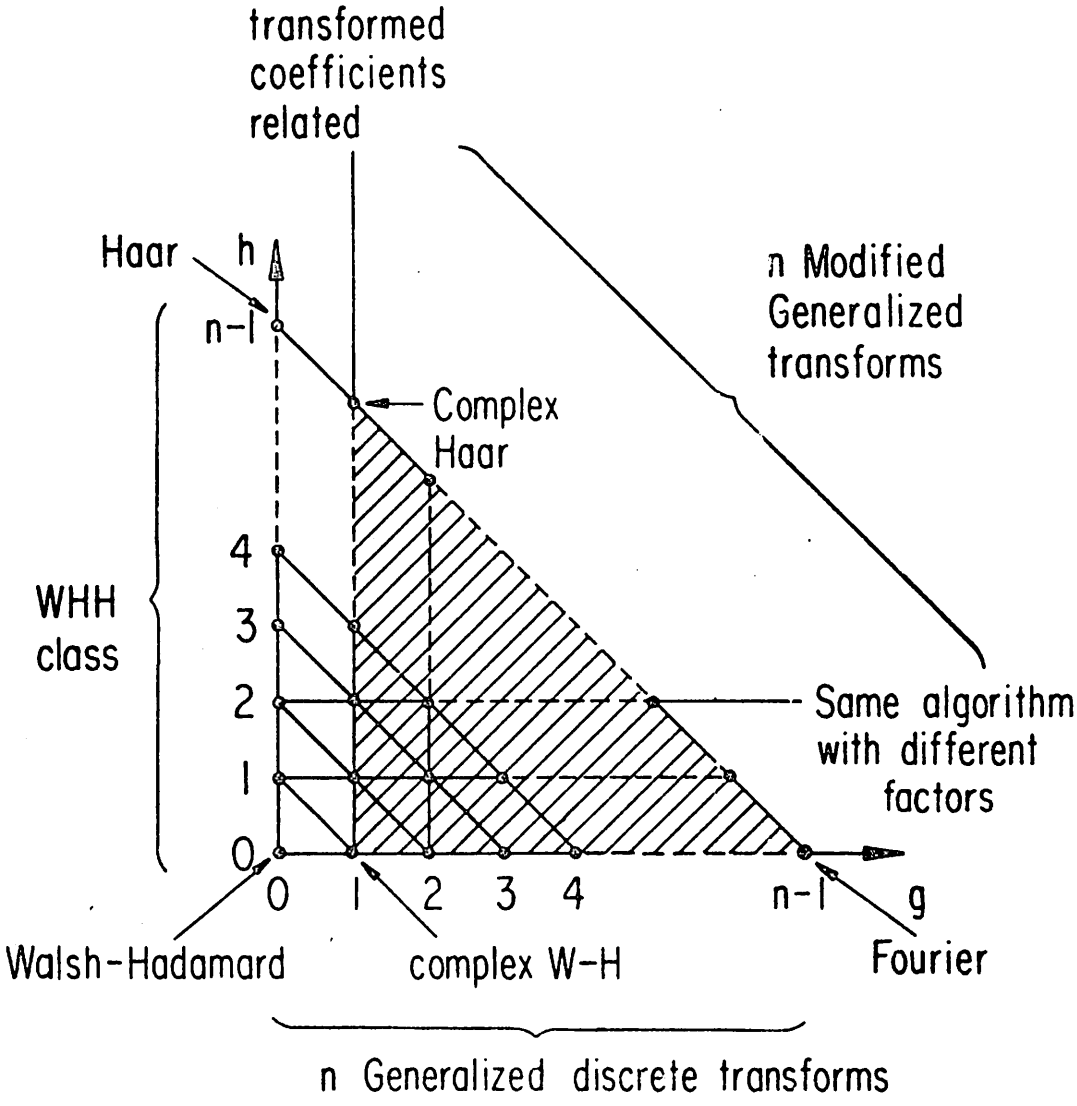


Fig. 3-9. WFH family of fast unitary transforms

transform considered by Andrews et al. [15]. This original core matrix can be used for all the recursive definitions considered.

b) Generalized 2 and 3 valued transforms:

In the definition of the W-H transform the role of $[F_2]$ as original core matrix and parent matrix can be performed by any unitary matrix [U] of order f. If [U] is an Hadamard matrix (its entries are $\pm 1/\sqrt{f}$) the generated matrix of order f^n will also be a Hadamard matrix. These matrices have been called "generalized 2-valued transforms" [19]. Similarly we can replace $[F_2]$ in the definition of the Haar transform by the same matrix [U] and we will generate a unitary matrix with entries 0 or $\pm 1/C_i$ where C_i is the normalizing factor of the i^{th} row: these matrices are the "generalized 3-valued transforms" [19]. More generally [U] can replace $[F_2]$ in the definition of the family of transforms between W-H and Haar.

c) IC_f family:

The IC_2 family was based on the set $\mathcal{C}_2$ for the parent matrices. We can define similarly the IC_f family based on the class $\mathcal{C}_f$ of parent matrices of order f which contains $[I_f]$ and $[F_f(\theta_1, \ldots, \theta_{f-1})]$ where k^{th} column of $[F_f(\theta_1, \ldots, \theta_{f-1})]$ = k^{th} column of $[F_f]$ x $\exp(e^{-j\theta k})$ ($[F_f]$ is the Fourier matrix of order f). The family IC_f has $\frac{f^n-1}{f-1}$ independent parent matrices chosen in $\mathcal{C}_f$: we say that IC_f has $\frac{f^n-1}{f-1}$ degrees of freedom over $\mathcal{C}_f$. The required number of additions and multiplication is computed recursively as done for (21) and (22):

$$\text{additions: } \mathcal{A}_{f^n} = f \cdot \mathcal{A}_{f^{n-1}} + \mathcal{A}_f A_n$$

$$\Rightarrow \mathcal{A}_{f^n} = \mathcal{A}_f \sum_{k=1}^{n} f^{n-k} A_k \qquad (23)$$

multiplications: $\mathcal{M}_{f^n} = f\,\mathcal{M}_{f^{n-1}} + \mathcal{M}_f\, A_n + L_n$

$$\mathcal{M}_{f^n} = \mathcal{M}_f \sum_{k=1}^{n} f^{n-k} A_k + \sum_{k-2}^{n} f^{n-k} L_k \quad (24)$$

where A_k is the number of parent matrices different from $[I_f]$ at the k^{th} stage and L_k the number of column multiplications with factors different from ± 1 (and maybe $\pm j$) at this k^{th} stage.

d) $WFH^{g,h}_{f^n}$ subfamily of IC_f

By analogy to the $WFH^{g,h}_{2^n}$ subfamily of IC_2 we can define the subfamily $[WFH^{g,h}_{f^n}]$ of IC_f as follows:

$[WFH^{g,h}_{f^n}]$ is obtained by successive generalized Kronecker products with the sets $\{[M^0], \ldots [M^k] \ldots, [M^{f^{p-1}-1}]\}$ of parent matrices such that

$[M^k] = [F_f^k]$ with column i of $[F_f^k]$ = column i of $[F_f]$ x $e^{\frac{-2\pi jki}{f^n}}$

for $k = 0 \bmod (f^{p-g-1})$

$[M^k] = [F_f]$ for $k = 0 \bmod (f^{p+h-n})$ and

$k \neq 0 \bmod (f^{p-g-1})$

$[M^k] = [I_f]$ otherwise

and at each level the permutation matrix $[P]^t$, $P_{st} = \delta_{uz}\,\delta_{kw}$ with $s = uf + k$, $t = wf^{p-1} + z$, is applied to reorder the columns.

It is easy to see that $[WFH^{n-1,\,0}_{f^n}]$ is the usual Fourier transform of order f^n; the matrices $[WFH^{0,0}_{f^n}]$ and $[WFH^{0,n-1}_{f^n}]$ have been introduced in the

literature respectively by Chrestenson [27] and Watari [28]. For these 2 matrices (23) and (24) reduce to the same recursive formulas and denoting by $\mathcal{P}$ the number of additions or multiplications:

for $WFH_{f^n}^{0,0}$ $\qquad \mathcal{P}_{f^n} = n \; \mathcal{P}_f \, f^{n-1}$

for $WFH_{f^n}^{0,n-1}$ $\qquad \mathcal{P}_{f^n} = \mathcal{P}_f \, \frac{f^n-1}{f-1}$

(This last result corrects the result given in [15], page 20.)
The other matrices of the family can be represented in the g-h diagram of Fig. 3-9.

3-7. Slant transform:

The Slant transform has been proposed by Enomoto et al. [29] for the order 8. Pratt et al. [30] have generalized this transform to any order 2^n and compared its performance with other transforms [31]. In this section we want to express the recursive generation of the Slant transform with our generative rules and compute the number of elementary operations required by its fast algorithm.

The Slant transforms of orders 4 and 8, $[S_4]$ and $[S_8]$, are as follows (in "natural" order).

$$[S_4] = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 & \\ 1 & -3 & 3 & -1 & \times 1/\sqrt{5} \\ 3 & 1 & -1 & -3 & \times 1/\sqrt{5} \\ 1 & -1 & -1 & 1 & \end{bmatrix} \qquad \begin{matrix} \underline{\text{Zequencies}} \\ 0 \\ 3 \\ 1 \\ 2 \end{matrix}$$

$$[S_8] = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & 0 \\ 1 & -3 & 3 & -1 & 1 & -3 & 3 & -1 & \times 1/\sqrt{5} & 7 \\ 7 & -1 & -9 & -17 & 17 & 9 & 1 & -7 & \times 1/\sqrt{5\times21} & 3 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & & 4 \\ 7 & 5 & 3 & 1 & -1 & -3 & -5 & -7 & \times 1/\sqrt{21} & 1 \\ 1 & -3 & 3 & -1 & -1 & 3 & -3 & 1 & \times 1/\sqrt{5} & 6 \\ 3 & 1 & -1 & -3 & -3 & -1 & 1 & 3 & \times 1/\sqrt{5} & 2 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & & 5 \end{bmatrix} \quad \text{Zequencies}$$

The rows can be reordered by zequencies with the same permutation as the W-H transform in natural order.

The Slant transform of order 2^n in natural order is obtained from the Slant transform of order 2^{n-1} in natural order by simple Kronecker product with $[F_2]$ followed by rotation of the rows 2^{n-2} and 2^{n-1} by the matrix

$$\begin{bmatrix} \text{Sin } \theta_n & \text{Cos } \theta_n \\ \text{Cos } \theta_n & -\text{Sin } \theta_n \end{bmatrix}$$

with $$\text{Sin } \theta_n = \sqrt{\frac{2^{2n-2}-1}{2^{2n}-1}}$$

and $$\text{Cos } \theta_n = \frac{2^{n-1}}{\sqrt{\frac{2^{2n}-1}{3}}}$$

This choice of θ_n introduces in the Slant matrix $[S_{2^n}]$ the Slant vector $\vec{S}$

with components linearly decreasing:

$$S_i = \frac{(2^n - 1) - 2i}{\sqrt{\frac{2^n(2^{2n} - 1)}{3}}}$$

But some normalizations can be delayed to the last stage of computation and the rows 2^{n-2} and 2^{n-1} are rotated by the matrix

$$\begin{bmatrix} 2^{n-1} & -\frac{(2^{2n-2}-1)}{3} \\ 1 & 2^{n-1} \end{bmatrix}$$

requiring 2 shifts, 2 additions, 1 multiplication. The corresponding algorithm is shown in Fig. 3-10 a.

Number of elementary operations:

Formulas (1) and (5) give:

for the number of additions:

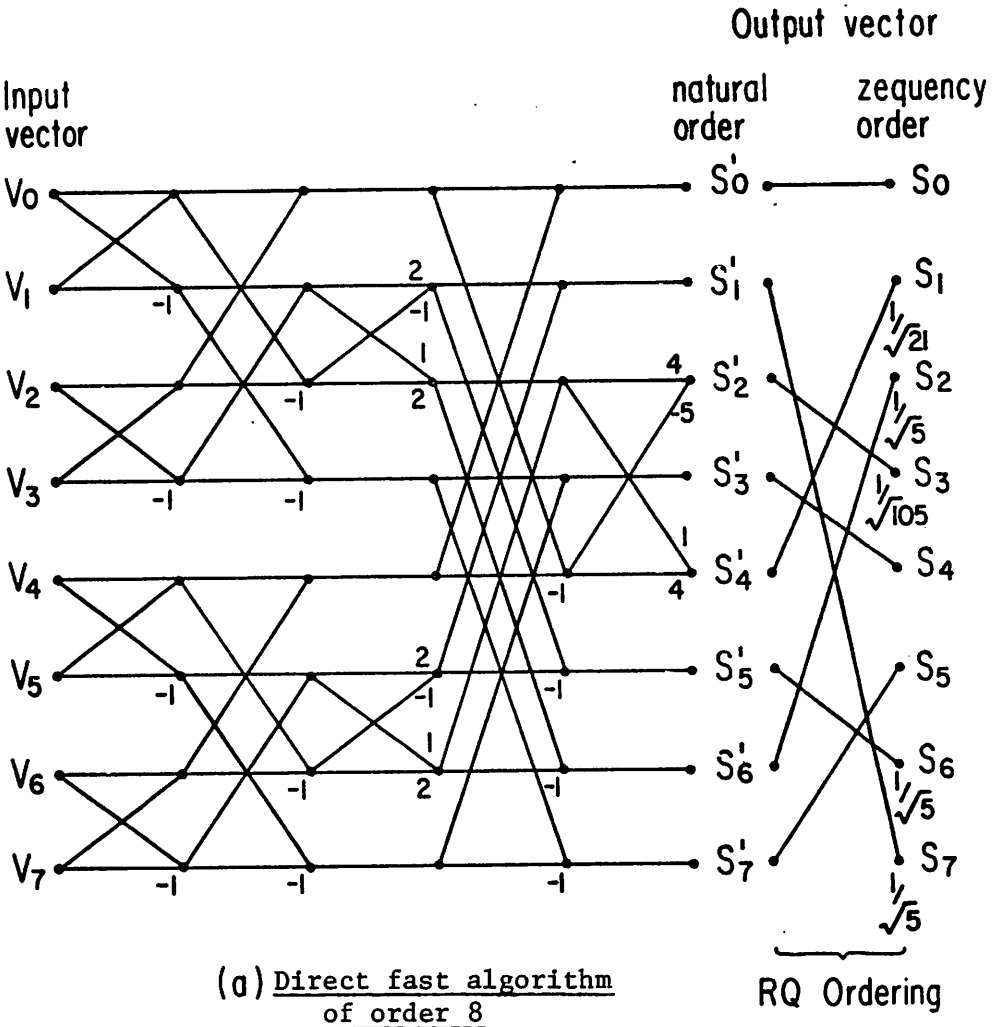
$$\mathcal{A}_{2^n} = 2\,\mathcal{A}_{2^{n-1}} + 2^{n-1} \cdot 2 + 2 \text{ with } \mathcal{A}_2 = 2$$

$$\text{hence } \mathcal{A}_{2^n} = (n+1)\,2^n - 2$$

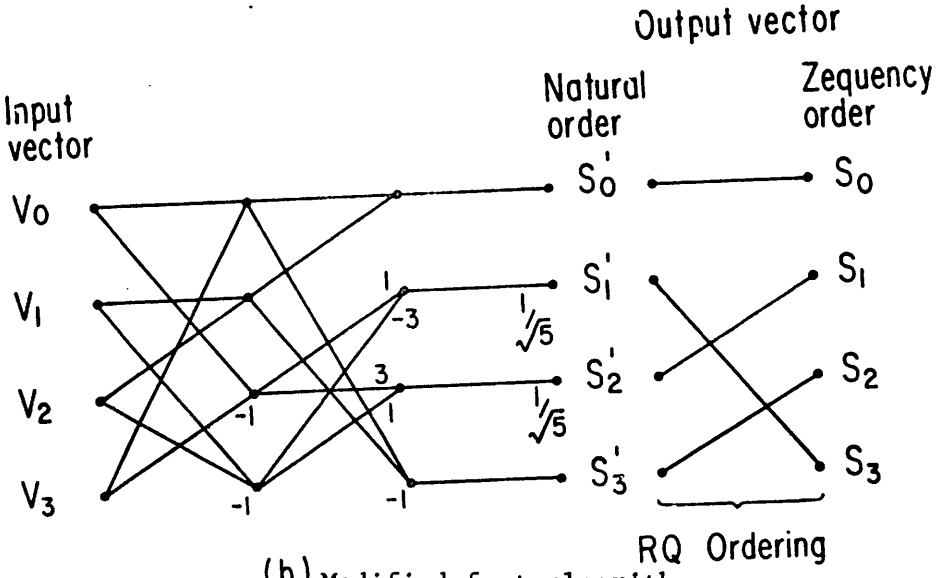
for the number of shifts:

$$\mathcal{S}_{2^n} = 2 \cdot \mathcal{S}_{2^{n-1}} + 2 \quad \text{with } \mathcal{S}_2 = 0$$

$$\text{hence } \mathcal{S}_{2^n} = 2^n - 2$$



(a) Direct fast algorithm of order 8



(b) Modified fast algorithm of order 4

Fig. 3-10. Slant transform

for the number of multiplications:

$$\mathcal{M}_{2^n} = 2\,\mathcal{M}_{2^{n-1}} + 1 \quad \text{with } \mathcal{M}_4 = 0$$

$$\text{hence } \mathcal{M}_{2^n} = 2^{n-2}-1$$

Finally $2^n - 2^{n-2}-1$ normalizations are required at the last stage of computation.

However the algorithm at the order 4 can be performed with 8 additions, 2 multiplications as shown in Fig. 3-10 b [30] instead of 10 additions and 2 shifts. The formulas (1) and (5) give then:

additions:

$$\mathcal{A}'_{2^n} = 2\,\mathcal{A}'_{2^{n-1}} + 2^{n-1} \cdot 2 + 2 \qquad \text{with } \mathcal{A}'_4 = 8$$

$$\text{hence } \mathcal{A}'_{2^n} = (2n+1)\,2^{n-1}-2 = \mathcal{A}_{2^n}-2^{n-1}$$

shifts:

$$\mathcal{S}'_{2^n} = 2 \cdot \mathcal{S}'_{2^{n-1}} + 2 \text{ with } \mathcal{S}'_4 = 0$$

$$\text{hence } \mathcal{S}'_{2^n} = 2^{n-1}-2 = \mathcal{S}_{2^n} - 2^{n-1}$$

multiplications:

$$\mathcal{M}'_{2^n} = 2\,\mathcal{M}'_{2^{n-1}} + 1 \text{ with } \mathcal{M}'_4 = 2$$

$$\text{hence } \mathcal{M}'_{2^n} = 32^{n-2}-1 = \mathcal{M}_{2^n} + 2^{n-1}$$

and as before $2^n - 2^{n-2} - 1$ normalizations.

3-8. Additional properties and generalizations of unitary transforms:

In this section we discuss briefly the complex extension of a real transform. We also point out some additional relations between transforms suggested by the unified framework presented.

3-8-1 Complex extension of a real transform:

From a real unitary matrix [RT] with rows $RT_0, \ldots, RT_{N-1}$, we construct a complex extension noted [CT] with rows $CT_0, \ldots, CT_N$ by creating two complex rows CT_p and CT_q from two real rows RT_m RT_n as follows

$$CT_p = \frac{1}{\sqrt{2}} (RT_m - j\ RT_n)$$

$$CT_q = \frac{1}{\sqrt{2}} (RT_m + j\ RT_n) \qquad (25)$$

Then the complex transform $\mathcal{V} = \mathcal{R} + j\mathcal{I}$ of a complex input vector V = R + j I is expressed uniquely from the real transforms of $\mathcal{R}$ and $\mathcal{I}$ denoted $\widetilde{\mathcal{R}}$ and $\widetilde{\mathcal{I}}$:

$$\mathcal{V}_p = CT_p \cdot V = \frac{1}{\sqrt{2}}(RT_m - RT_n)\ (R + j\ I) \text{ or}$$

$$\mathcal{V}_p = \frac{1}{\sqrt{2}}(\widetilde{\mathcal{R}}_m + \widetilde{\mathcal{I}}_n) + j(\widetilde{\mathcal{I}}_m - \widetilde{\mathcal{R}}_n) \qquad \text{and similarly}$$

$$\mathcal{V}_q = \frac{1}{\sqrt{2}}(\widetilde{\mathcal{R}}_m - \widetilde{\mathcal{I}}_n) + j(\widetilde{\mathcal{I}}_m + \widetilde{\mathcal{R}}_n)$$

With these relations the properties of complex transforms can be deduced from those of the real transform. In the literature, besides the real and complex Fourier transforms, the complex W-H transfoms (also called Complex BIFORE transform) [32][33] complex Haar transform (also called Complex Modified BIFORE transform) [34] have been defined.

Note that the complex W-H transform obtained by relations (25) would have entries $\pm 1 \pm j$; commonly the rows are then rotated by $\frac{1+j}{\sqrt{2}}$ to give a transform with $\sqrt{2}$ entries ± 1 and $\pm j$. The rows of the complex W-H transform can be ordered according to a generalized frequency defined as the number of clockwise rotations around the origin when following cyclically the entries of a row.

3-8-2. Multidimensional transforms

The techniques presented for the one dimensional transforms extend to multidimensional separable transforms. Let us denote an input array of p dimensions by $A_{i_q, \ldots, i_p}$ and the p-dimensional separable transform by $T_{u_1, \ldots, u_p, i_1, \ldots, i_p} = T^1_{u_1 i_1} T^2_{u_2 i_2} \cdots T^p_{u_p i_p}$. Then the transformed array

$$B_{u_1, \ldots, u_p} = \sum_{i_1} \sum_{i_2} \cdots \sum_{i_p} A_{i_1, \ldots, i_p} T_{u_1, \ldots, u_p, i_1, \ldots, i_p}$$

can be written

$$B_{u_1, \ldots, u_p} = \sum_{i_p} T^p_{u_p i_p} \sum_{i_{p-1}} \cdots \sum_{i_1} A_{i_1, \ldots i_p} T^1_{u_1 i_1}.$$

If we express both arrays as 1 dimensional vectors A and B , for which indexes are obtained by lexicographic ordering of the indexes $(i_1, \ldots, i_p)$ and $(u_1, \ldots, u_p)$, the multidimensional transform can be expressed as a 1-dimensional transform:

$$A = [[T^1] \otimes [T^2] \ldots \otimes [T^P]]\ B$$

$$A = [T]\ B$$

The multidimensional transform has been reduced to a 1 dimensional transform. This expression now allows the evalution of the number of elementary operations and other generalizations discussed previously.

3-8-3. Relations between transforms

Two transforms with similar structures will often be related by matrix relations or energy invariants between the two sets of transformed coefficients.

a) matrix relations between transforms of same order:

In chapter 2, matrix relations between the Haar and W-H transforms were proved. More generally, for WFH families, similar relations hold for all transforms lying on the same vertical line in the g-h graph of Fig. 3-9. These transforms only differ by the number of parent matrices $[F_2]$ they include. Therefore a multiplication by all the missing $[F_2]$ matrices will generate one transform from the other. Note that these relations only involve computations in zones as defined in 3.3 or subzones (zonal divisions of a zone).

b) energy invariants:

By Parseval's theorem the total energy of the transform coefficients of a same vector with different transforms is preserved. However, it may happen that the energy of a subset of coefficients is the same for some transforms: we say then there is an energy invariant between these transforms. Energy invariants are most likely when the transforms have an identical structure with different factors. For example, by direct comparison of the

algorithms for the Fourier, W-H and modified Haar transforms (Fig. 3-3b, 4c and 5c), it is clear that the transformed coefficients before respective reorderings have identical energies in the zones defined in 3.3. This leads to the following energy invariants for the order 8.

Zone	Fourier (frequencies)	W-H (zequencies)	Mod. Haar (rank)
0	0	0	0
1	4	7	1
2	2,-2	3,4	2,3
3	1,3,-1,-3	1,2,5,6	4,5,6,7

For the WFH families, the transforms with same sets of invariants form nested triangles as shown in Fig. 9: the introduction of additional factors leads to additional smaller subsets of coefficients of a same subzone over which energy is invariant; the relations between transforms which exist along vertical lines of the diagram of Fig. 3-9 preserves the energy invariance in zones. The invariants between the Generalized Discrete Transforms and the Modified Discrete Transforms have been studied by Rao et al. [18].

3-9. Conclusions:

In this chapter, we have presenter a unified treatment of unitary transforms having a fast algorithm. The use of recursive rules to describe unitary transforms allows a systematic way to view known transforms, to generate new transforms and provide a general approach to the evaluation of the computational complexity of transform algorithms. Among transforms

which are clearly related, we have studied the IC_f families and the WFH subfamilies which include most of the transforms considered in the literature.

In addition to allowing the introduction of new transforms with properties of interest, the framework provided can be used in several other studies and applications of unitary transforms. In particular an error analysis of unitary transforms is now presented in the following chapters.

C H A P T E R I V

ERROR ANALYSIS IN FIXED-POINT COMPUTATION

4-1. Introduction:

In chapter 3 we presented a common framework which defines unitary transforms with a fast algorithm by using simple generative rules. In practical implementations of algorithms, numbers are represented with finite length registers. This will lead to errors in the representation of coefficients and to round-off errors in computations. In this chapter and the following one we consider an analysis of round-off errors and comment only briefly on representation errors. Round-off errors depend on (1) the mode chosen to represent numbers: in this chapter we consider the fixed-point and block-scaling (also called block floating-point) modes while in chapter 5 we consider the floating-point mode. and (2) the computational procedure : truncation or rounding (and for rounding we have to specify how the midrange point is handled).

Our objective is to estimate the round-off error of each output coefficient or at least to estimate the average mean square error over the set of output coefficients. We shall also consider the output error-to-signal ratio (ratio of total output mean square error to total output signal energy). However round-off errors are usually data dependent: in floating-point mode we shall see that each error is data dependent while in fixed-point mode the algorithm is usually data dependent. In order to carry out an analysis, we have two options:

-to compute bounds (worst or best case analysis)

-to assume a statistical model for the input coefficients and the

simplest such model is a white signal (statistically independent input coefficients with equal variances).

Previous contributions to the round-off error analysis for fixed-point computation consider only the Fast Fourier Transform (FFT) of radix 2 . Welch [1] has considered the best and worst cases and derived bounds. Oppenheim & Weinstein [2] , Weinstein [3] have considered a white signal model and presented experimental results for rounding and truncation. To our knowledge there has not been any theoretical approach for the case of truncation nor any study of the block-scaling mode previous to our work. In the following we evaluate the round-off errors with a general framework applicable to any fast unitary transform defined in chapter 3. We use the models and assumptions of previous works but our systematic approach allows us a more complete and accurate study. We reestablish most of their results as particular cases and, even for the well known FFT, we obtain some new results.

In the following, we discuss first the error models for computation round-off and scaling with rounding and truncation. Then we use the framework of chapter 3: fast unitary transforms are defined by combination of "parent matrices" according to recursive generative rules. We recall our notations for the generalized Kronecker product of two sets of parent matrices, $\{\mathcal{A}\} = \{ [A^0], \ldots\ldots, [A^{m-1}] \}$ and $\{\mathcal{B}\} = \{ [B^0], \ldots$ $\ldots, [B^{n-1}] \}$. $[C] = \{\mathcal{A}\} \otimes \{\mathcal{B}\}$ and we have shown that

$$[C] = [P] \left[\mathrm{Diag}\{\mathcal{A}\} \right] [P^t] \left[\mathrm{Diag}\{\mathcal{B}\} \right]$$

The backbone of our approach is to exploit the recursive use of the generalized Kronecker product in the definitions of fast unitary transforms in order to derive recursive relations between round-off errors at successive stages of computation. Therefore our analysis

will have two steps:

(1) generation of round-off errors in parent matrices $[A^0],\ldots,[A^{m-1}]$ and $[B^0],\ldots,[B^{m-1}]$

(2) propagation of these errors in a generalized Kronecker product.

We will take this approach successively for the different conditions of scaling and computational procedures considered in this chapter for fixed-point computation and again in chapter 5 for floating-point computation.

4-2. Error models in fixed-point and block-scaling computations:

In a fixed-point representation of numbers there are two sources of error : the round-off errors introduced in a multiplication and the scaling errors introduced to avoid overflow. We consider them successively in the cases of rounding and truncation.

a) round-off errors in multiplications:

From Wilkinson [4•page 4•] the fixed-point representation of the product a . a' ,denoted fi(aa'), is such that

$$fi(aa') = aa' + \varepsilon$$

where ε is the round-off error such that

for rounding : $|\varepsilon| \leq \frac{1}{2} 2^{-b}$

for truncation : $0 \leqslant \sigma \triangleq |\varepsilon| = -\text{sign}(aa')\,\varepsilon < 2^{-b}$

when the register has b bits and a sign bit.

If we consider that ε and σ are random variables uniformly distributed in their respective intervals, we have

for rounding : ε has zero mean and variance $\Delta^2 = 2^{-2b}/12$

for truncation : σ has mean $\mu = \frac{1}{2} 2^{-b}$ and variance Δ^2.

We note that a complex multiplication involves in general 4 real

multiplications and, assuming that the errors they introduce are independent, the variance of a complex multiplication error is $4\,\Delta^2$. For rounding the error has zero mean while for truncation the mean depends on the signs of the real and imaginary parts. If the operands a and a' have been previously scaled by S bits, μ is multiplied by 2^S and Δ^2 by 2^{2S}.

b) scaling error:

Let us assume that a cumulative scaling by (S-q) bits has already been performed at previous stages of computation and that we scale now the number a by q bits. We choose to write the error ε' as :

for rounding : $|\varepsilon'| \leq \frac{1}{2}\, 2^{-b+q}.\, 2^{S-q}$

for truncation : $0 \leq \delta' = -\text{sign}(a)\ \varepsilon' \leq 2^{-b+q}.\, 2^{S-q}$

Assuming that ε' and δ' are uniformely distributed random variables, we have:

for rounding : ε' has zero mean and variance $\Delta'^2\ 2^{2(S-q)}$

with $\Delta'^2 = 2^{-2(b-q)} / 12$

for truncation : δ' has mean $\mu'\, 2^{S-q}$ with $\mu' = 2^{-b}/4$

and variance $2^{2(S-q)}\ \Delta'^2$

For $q = 1$ however, the scaled bit is either 1 or 0, making the uniform distribution erroneous. We have then directly :

$$\Delta'^2 = \tfrac{1}{2}\, 2^{-2b} = 6\,\Delta^2 \qquad \text{and}$$

$$\mu' = \tfrac{1}{2}\, 2^{-b}$$

c) scaling methods:

In the next section we analyse the combination of round-off and scaling errors in the case of rounding while the case of truncation will be studied in section 4-4. However there are many ways to perform scaling, depending on the test of overflow. Following Welch [1] and Weinstein [3]

we first consider two extreme cases:

- no scaling is ever performed
- a fixed scaling is performed at each stage of computation.

In these two cases the errors are independent of the data and we obtain directly simple recursive relations for rounding and for truncation. Then we consider the more common scheme of block-scaling in which we impose a scaling of all intermediate results of a stage of computation if any overflow occurs : this scheme is data dependent and we shall need a statistical model for the probability of overflow.

4-3. Error analysis for fixed-point computation with rounding:

We consider successively the two extreme cases of no-scaling and step-by-step scaling.

4-3-1. No-scaling:

This case with rounding is the simplest case of error analysis we consider in this dissertation. We give a detailed treatment in this section in order to stress the successive steps of our approach and also because subsequent analyses will depend heavily on this section.

We consider first a parent matrix $[T]$ of order N which transforms an input vector $\vec{V}$ into $\vec{W}$ so that

$$\vec{W} = [T] \vec{V} \quad \text{or}$$

$$W_k = \sum_{\ell=0}^{N-1} T_{k\ell} V_\ell$$

Separating real and imaginary parts, we have :

$$\begin{aligned} Re(W_k) &= \sum_{\ell=0}^{N-1} \left[Re(T_{k\ell}) \; Re(V_\ell) - Im(T_{k\ell}) \; Im(V_\ell) \right] \\ Im(W_k) &= \sum_{\ell=0}^{N-1} \left[Im(T_{k\ell}) \; Re(V_\ell) + Re(T_{k\ell}) \; Im(V_\ell) \right] \end{aligned} \tag{1}$$

We denote by $\mathcal{E}_r(W_k)$ and $\mathcal{E}_i(W_k)$ the round-off errors on the real and imaginary parts of W_k , $\varepsilon_{...}$ the round-off errors caused by real multiplications. $\varepsilon_{...}$ is null if the corresponding multiplication is a multiplication by ± 1 . We assume the individual round-off errors to be independent and that there is no error in the representation of $T_{k\ell}$. Then :

$$\begin{aligned}
\mathcal{E}_r(W_k) &= \sum_{\ell=0}^{N-1} \left[\mathrm{Re}(T_{k\ell})\ \mathcal{E}_r(V_\ell) + \varepsilon_{4\ell,k} - \mathrm{Im}(T_{k\ell})\ \mathcal{E}_i(V_\ell) - \varepsilon_{4\ell+1,k} \right] \\
\mathcal{E}_i(W_k) &= \sum_{\ell=0}^{N-1} \left[\mathrm{Im}(T_{k\ell})\ \mathcal{E}_r(V_\ell) + \varepsilon_{4\ell+2,k} + \mathrm{Re}(T_{k\ell})\ \mathcal{E}_i(V_\ell) + \varepsilon_{4\ell+3,k} \right]
\end{aligned} \tag{2}$$

In case of rounding, the mean square error of W_k , denoted $E_W(k)$, is given by the error variance. Assuming that all errors in the input vector are independent of each other and of the new round-off errors, (2) gives :

$$E_W(k) = \sum_{\ell=0}^{N-1} \left\| T_{k\ell} \right\|^2 E_V(\ell) + \mu_k \Delta^2 \tag{3}$$

where μ_k is the number of real multiplications performed to compute W_k from $\vec{V}$.

For a transform generated by a generalized Kronecke_ product (rule 3), by permutation and multiplication by roots of the unity of the rows (rule 2 b and c) and columns (rule 1 a and b), the previous assumption of independence of the input errors $\mathcal{E}(V_\ell)$ holds [1] .

---

[1] Note however that the output errors are then correlated.

For a transform generated by the general rule 2, the errors in the input vector of a parent matrix are not independent and their correlations should be considered in (3).

Error propagation :

Let us now use (3) and focus attention on the combination of errors in a generalized Kronecker product. Following the notations of Figure 3-2 we denote by $b_j^k \Delta^2$ the variance of the error on the j th coefficient of the output vector of the matrix $[B^k]$ (of order m) and α_i^j the number of real multiplications in a dot product with the i th row of matrix $[A^j]$ (of order n). Then the variance of the error on the (im + j) th coefficient of the output vector , denoted E(im + j), is obtained from (3) :

$$E(im + j) = \sum_{k=0}^{n-1} \underbrace{b_j^k \left\| A_{ik}^j \right\|^2}_{\substack{\text{sum of variances of} \\ \text{independent error} \\ \text{random variables} \\ \text{propagated from} \\ \text{input vector}}} \Delta^2 + \underbrace{\alpha_i^j \Delta^2}_{\substack{\text{sum of variances} \\ \text{of new errors}}} \qquad (4)$$

Summing the error variances of all the output coefficients, we obtain the total error variance, denoted V_C , ($[C] = \{A\} \otimes \{B\}$) :

$$V_C = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} E(im + j)$$

$$V_C = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} b_j^k \left\| A_{ik}^j \right\|^2 \Delta^2 + \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \alpha_i^j \Delta^2 \qquad (5)$$

Several cases of interest give simplified expressions :

1) $\left\| A_{ik}^j \right\| = 1$

This happens for the WFH family of transforms for example (see chapter 3 section 4-3) when the normalizations are performed after the last

stage of computation. Then (4) and (5) become :

$$E(im + j) = \left(\sum_{k=0}^{n-1} b_j^k + \alpha_i^j \right) \Delta^2 \qquad (6)$$

$$V_C = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} b_j^k \Delta^2 + \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \alpha_i^j \Delta^2$$

$$\text{or} \quad V_C = n \sum_{k=0}^{n-1} V_{B^k} + \sum_{j=0}^{m-1} V_{A^j} \qquad (7)$$

where V_{B^k} (V_{A^j}) denote the total error variance of the output error vector after multiplication by matrix $[B^k]$ ($[A^j]$) , when the input vector is error free.

2) $b_j^k = b_j$ for any k

This happens for example for all IC transforms (see chapter 3). Then , using the orthonormality of the matrices $[A^j]$, (4) and (5) become :

$$E(im + j) = n\, b_j \Delta^2 + \alpha_i^j \Delta^2 \qquad (8)$$

$$V_C = n^2 \sum_{j=0}^{m-1} b_j \Delta^2 + \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \alpha_i^j \Delta^2$$

$$\text{or} \quad V_C = n^2 V_B + \sum_{j=0}^{m-1} V_{A^j} \qquad (9)$$

Note the similarity between the relations (7) or (9) and the relation (4) of chapter 3 . This is not surprising since all errors have the same weight and thus the total error variance is proportional to the number of real multiplications performed. A factor n appears however due to the fact that the normalizations are carried out only after the last stage of computation.

Now for each fast transform defined in terms of parent matrices

and rules 1, 2 b and c, 3 of chapter 3, we can compute, using the recursive relations between errors at successive stages presented in this section, the mean square error of each coefficient and the total output mean square error. We shall consider as an example the FFT algorithms in section 4-3-3.

4-3-2. Step-by-step scaling:

We consider now the other extreme case where a fixed scaling is performed at every stage of computation. An expression similar to (2) but with the scaling error is obtained :

$$\mathcal{E}_r(W_k) = \sum_{\ell=0}^{N-1} \left\{ \mathrm{Re}(T_{k\ell}) \left[\mathcal{E}_r(V_\ell) + 2^{S-q}\, \mathcal{E}'_{2\ell,k} \right] + \mathcal{E}_{4\ell,k}\, 2^S \right. \\ \left. - \mathrm{Im}(T_{k\ell}) \left[\mathcal{E}_i(V_\ell) + 2^{S-q}\, \mathcal{E}'_{2\ell+1,k} \right] - \mathcal{E}_{4\ell+1,k}\, 2^S \right\} \tag{10}$$

where S is the number of scaling bits up to the present stage of computation (S-q at the previous stage) and $\mathcal{E}'_{\ldots}$ the scaling error. As in (3) the round-off errors are zero mean and the mean square of each output coefficient is obtained from (10) and the similar expression for $\mathcal{E}_i(W_k)$:

$$E_W(k) = \sum_{\ell=0}^{N-1} \|T_{K\ell}\|^2 \left[E_V(\ell) + 2 \cdot 2^{2S-2q}\, \Delta'^2 \right] + \mu_k\, 2^{2S}\, \Delta^2 \tag{11}$$

Error propagation:

To analyse the error propagation in a generalized Kronecker product, we assume that we scale the intermediate results before each stage of computation. However the scaling errors introduce a dissymmetry between the roles of the two sets of matrices $\{\mathcal{A}\}$ and $\{\mathcal{B}\}$ of a generalized Kronecker product. As we have seen in chapter 3, fast transforms may

be defined recursively with the previous stage making the set $\{\mathcal{A}\}$ (e. g. FFT with Cooley-Tukey algorithm) or the set $\{\mathcal{B}\}$ (e. g. FFT with Sande-Tukey algorithm). We assume first that scaling is performed after the matrices of the set $\{\mathcal{A}\}$. Then, using the notations of 4-2-1, the mean square error on the (im + j) th coefficient of the output vector is obtained from (11) :

$$E(im + j) = \sum_{k=0}^{n-i} \left[\underbrace{b_j^{\,k} \Delta^2}_{\text{errors from previous stage}} + \underbrace{2 \Delta'^2 2^{2S-2q}}_{\text{scaling errors}} \right] \left\| A_{ik}^{j} \right\|^2 + \underbrace{\alpha_i^{\,j} \Delta^2 2^{2S}}_{\text{new round-off errors}} \tag{12}$$

Using the orthonormality property we can write:

$$V_c = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} b_j^{\,k} \left\| A_{ik}^{j} \right\|^2 \Delta^2 + \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \alpha_i^{\,j} \Delta^2 (2^S)^2 + 2\, m\, n^2\, \Delta'^2\, (2^{S-q})^2 \tag{13}$$

Again we consider two particular cases of interest which simplify (12) and (13) :

1) $\left\| A_{ik}^{j} \right\| = 1$

Then (12) and (13) become :

$$E(im + j) = \sum_{k=0}^{n-1} b_j^{\,k} \Delta^2 + \alpha_i^{\,j} \Delta^2 (2^S)^2 + 2\, n\, \Delta'^2 (2^{S-q})^2 \tag{14}$$

$$V_C = n \sum_{k=0}^{n-1} V_{B^k} + \sum_{j=0}^{m-1} V_{A^j} (2^S)^2 + 2\, m\, n^2\, \Delta'^2 (2^{S-q})^2 \tag{15}$$

2) $b_j^{\,k} = b_j$ for any k:

Then, with the orthomormality of $[A^j]$, (12) and (13) give :

$$E(im + j) = n\, b_j\, \Delta^2 + \alpha_i^{\,j} \Delta^2 (2^S)^2 + 2\, n\, \Delta'^2 (2^{S-q})^2 \tag{16}$$

$$V_C = n^2 V_B + \sum_{j=0}^{m-1} V_{A^j} (2^S)^2 + 2\, m\, n^2\, \Delta'^2 (2^{S-q})^2 \tag{17}$$

Whithout details we give now the similar relations obtained when scaling is performed before the matrices of the set $\{\mathcal{B}\}$. In this case we obtain for the error variances :

$$E(im + j) = \sum_{k=0}^{n-1} \beta_i^k \, \Delta^2 \, (2^q)^2 \, \| A_{ik}^j \|^2 + a_i^j \, \Delta^2 \, (2^q)^2 + 2 \, m \, n \, \Delta'^2 \qquad (18)$$

$$V_C = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} E(im + j)$$

where β_j^k denotes the number of real multiplications in the dot product with the j th row of matrix $[B^k]$, $a_i^j \, \Delta^2$ denotes the variance of the error in the j th coefficient of the output vector of matrix $[A^j]$. In the particular case where $\| A_{ik}^j \| = 1$ we obtain:

$$V_C = n \sum_{k=0}^{n-1} V_{B^k} (2^q)^2 + \sum_{j=0}^{m-1} V_{A^j} (2^q)^2 + 2 \, (m \, n)^2 \, \Delta'^2 \qquad (19)$$

4-3-3. Application to FFT:

The previous formulas allow the fixed-point error analysis for a large variety of transforms. The FFT with radix 2 is certainly the most widely used and the precision obtained is an important problem. We now apply the results we have obtained to the Cooley-Tukey and Sande-Tukey algorithms. Both algorithms have been defined with generalized Kronecker products in chapter 3. We first recall these expressions:

Cooley-Tukey algorithm: $$\left[F_{2^n}\right] = \left[\left\{ \left[F_2^k\right] \right\} \otimes \left[F_{2^{n-1}}\right] \right] \left[P^t\right]$$

$$\text{with} \quad \left[F_2^k\right] = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & \exp(-2\pi jk/2^n) \\ 1 & -\exp(-2\pi jk/2^n) \end{bmatrix}$$

Sande-Tukey algorithm: $$\left[F_{2^n}\right] = \left[\left[F_{2^{n-1}}\right] \otimes \left\{ \left[F_2'^k\right] \right\} \right] \left[P\right]$$

$$\text{with} \quad \left[F_2'^k\right] = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ \exp(-2\pi jk/2^n) & -\exp(2\pi jk/2^n) \end{bmatrix}$$

a) Cooley-Tukey algorithm:

The application of (8) and (16) leads to equal mean square errors on coefficients of a same zone , as defined in chapter 2. Here, before reordering according to the frequency, the zones appear as the sets of coefficients with indexes λ such that $\lambda \equiv 2^{\ell} \pmod{2^{\ell+1}}$ $\ell = 0,..,n-1$ Then it can be shown that the mean square errors are :

no-scaling:

$$\begin{aligned} E(\lambda) &= (2^{n-\ell-2}-1)\ 4\ \Delta^2 \quad \text{for} \quad \lambda \equiv 2^{\ell} \pmod{2^{\ell+1}} \\ &\qquad\qquad \ell = 0,\ldots,n-2 \\ &= 0 \quad \text{for} \quad \lambda = 0,\ 2^{n-1} \end{aligned} \tag{20}$$

step-by-step scaling :

$$\begin{aligned} E(\lambda) &= (2^{2n+1} - 2^{n+\ell+3})\ 4\ \Delta^2 + 2^{n+1}(2^n-1)\ \Delta'^2 \\ &\qquad\qquad \text{for} \quad \lambda \equiv 2^{\ell} \pmod{2^{\ell+1}} \\ &= 2^{n+1}(2^n-1)\ \Delta'^2 \quad \text{for} \quad \lambda = 0, 2^{n-1} \end{aligned} \tag{21}$$

The corresponding total error variance is given by the following recursive formulas :

no-scaling:

From (9), we obtain :

$$V_{2^n} = 4\ V_{2^{n-1}} + 2\ \mu_n\ 4\ \Delta^2 \tag{22a}$$

where μ_n is the number of factors (each introduces two round-off errors in a parent matrix hence $2\ \mu_n$ in the relation). Then for $\mu_n = 2^{n-1} - 2$ as given in 3-3-1 a, we obtain, solving for V_{2^n} :

$$V_{2^n} = (\ 2^{2n-1}/3 - 2^n + 4/3)\ 4\ \Delta^2 \tag{22}$$

which is, with more accuracy, the result obtained by Welch[2] [1] for the round-off error alone (we obtain his result by taking $\mu_n = 2^{n-1}$ for $n \geq 3$ and $\mu_n = 2^{n-1} - 2$ for $n < 3$ in the above recursive equation). Weinstein [2] [3] has an approximate result by taking $\mu_n = 2^{n-1}$ for all n. In fact (22) is a direct consequence of (20) , by averaging over ℓ :

$$V_{2^n} = 2 \sum_{\ell=0}^{n-2} 2^{n-1-\ell} \left(2^{n-\ell-2} - 1 \right) 2 \Delta^2$$

step-by-step scaling :

From (11) we obtain:

$$V_{2^n} = 4 V_{2^{n-1}} + 2 \mu_n 4 \Delta^2 (2^n)^2 + 2^n 4 \Delta'^2 (2^{n-1})^2$$

With $\mu_n = 2^{n-1} - 2$ we obtain

$$V_{2^n} = 4^{n+1} (2^{n-1} - n) 4 \Delta^2 + 4^n (2^{n+1} - 2) \Delta'^2 \qquad (23a)$$

result which could also be obtained by averaging (21) over ℓ :

$$V_{2^n} = \sum_{\ell=0}^{n-2} E(\lambda) 2^{n-\ell-1} + 2 . 2^n . 2^n (2^n - 1) \Delta'^2 \qquad (23)$$

On Figure 4-1 we have plotted the experimental results reported by Weinstein [3] with white imput signal uniformely distributed in $[-1 , +1]$ (real and imaginary parts). We have plotted the error estimation with his approximation and finally the curve obtained from (23). We find a perfect fit of our theoretical results with his experimental measures.

---

[2] Welch assumes also that rounding is performed only after the additions of the results of the two multiplications of each term of the dot product of (2). This procedure reduces the factor of Δ^2 by half.

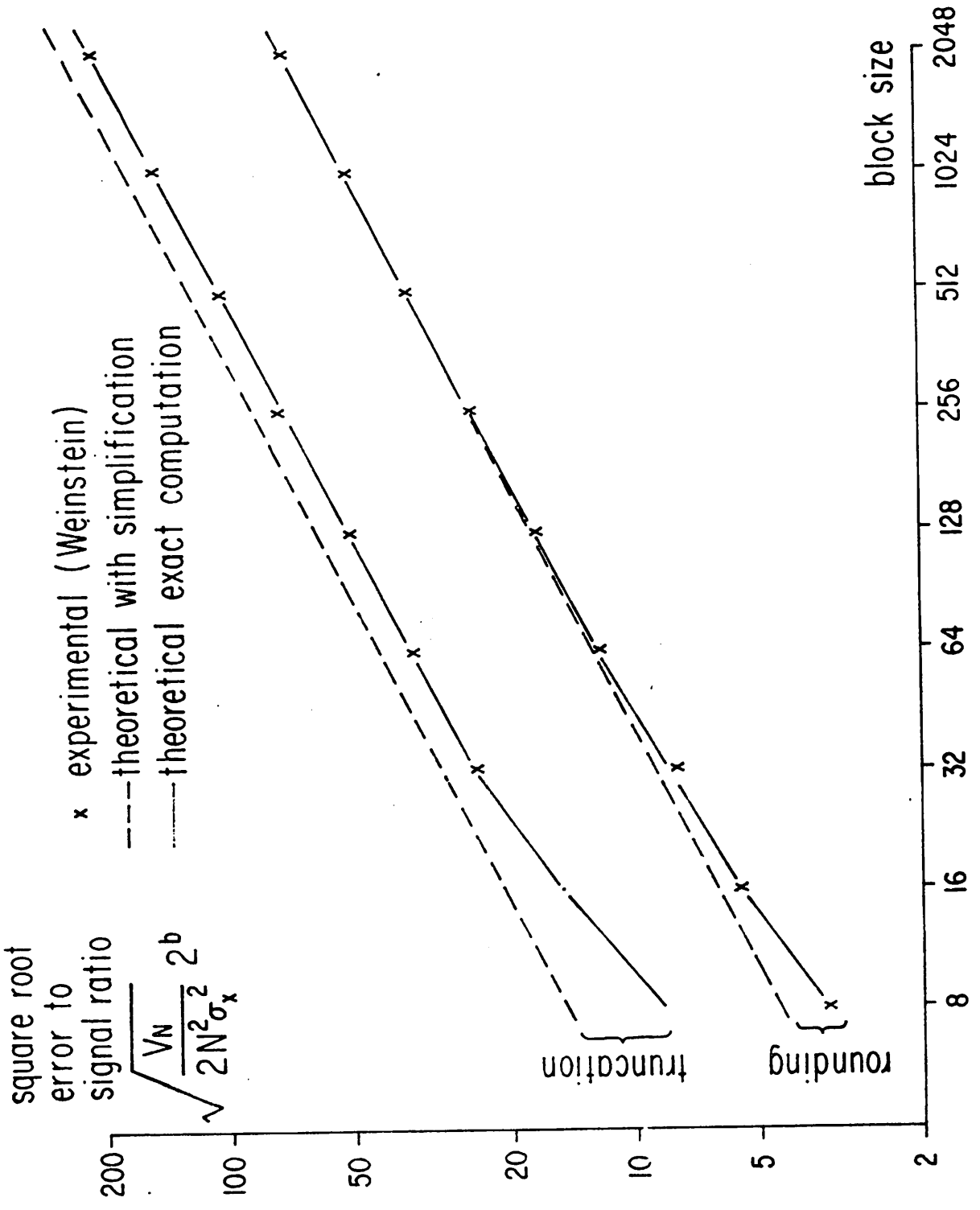


Fig. 4-1. FFT fixed-point error analysis with step-by-step scaling

b) Sande-Tukey algorithm:

In a very similar way we can compute the individual and total mean square errors for the Sande-Tukey algorithm. We obtain:

individual mean square errors: From (8) and (18), we have

$$\text{no-scaling:} \quad E(\lambda) = \sum_{i=1}^{n-1} (2^i - 2) \quad p_{n-i} \quad 4 \quad \Delta^2$$

$$\text{with} \quad \lambda = p_0 + 2p_1 + \ldots\ldots + 2^{n-1} p_{n-1}$$

(binary expression of λ)

step-by-step scaling:

$$E(\lambda) = \sum_{i=1}^{n-1} (2^i - 2) \; 4^{n-i} p_{n-i} 4 \; \Delta^2 + 2^{n+1} \; (2^n - 1) \; \Delta'^2$$

total mean square error: From (7) and (19) or by summation of the individual errors, we have:

$$\text{no-scaling:} \quad V_{2^n} = 2^{n-1} \mu_n \; 4 \quad \Delta^2 \quad + 2 \; V_{2^{n-1}}$$

$$\text{Hence} \quad V_{2^n} = 2^n \; (\; 2^{n-1} - n) \quad 4 \quad \Delta^2 \qquad (24)$$

step-by-step scaling:

$$V_{2^n} = 4 \; 2^{n-1} \mu_n \; 4 \; \Delta^2 + 2 \quad 4 \quad V_{2^{n-1}} + 2 \; 2^{2n} \quad \Delta'^2$$

$$\text{Hence} \quad V_{2^n} = (2^{3n+1}/3 - 2^{2n} + 2^{n+2}/3) \; 4 \; \Delta^2 + 2^{2n+1} \; (2^n - 1) \; \Delta'^2 \qquad (25)$$

c) Comparisons:

The Cooley-Tukey algorithm has the advantage that multiplication errors in parent matrix operations occur in later stages of computation as compared to the Sande-Tukey algorithm (see Figures 3-3 b and c). On the other hand each mutiplicative round-off generates two errors which propagate independently while each Sande-Tukey factor introduces only

one error. These two effects oppose themselves so that no-scaling and step-by-step bounds for the Sande-Tukey algorithm lie inside the corresponding bounds for the Cooley-Tukey algorithm. However, if we use one of these extreme cases, the Sande-Tukey algorithm has to be prefered for a step-by-step scaling while the Cooley-Tukey algorithm has to be prefered for a no-scaling scheme. For an intermediate scheme our only conclusion is that the Sande-Tukey algorithm has a smaller range of possible errors.

The step-by-step scheme, for both algorithms, has a much higher error ; however the input signal acceptable with this scheme has a much higher magnitude. To compare the two schemes it is therefore interesting to compute the error-to-signal ratio, denoted E/S. The error-to-signal ratio is the ratio of the total mean square error to the total output signal variance. If σ^2 is the average input signal variance, the total output signal variance is $N^2 \sigma^2$ so that

$$E/S = V_N / N^2 \sigma^2$$

In the case of no-scaling, we have roughly from (22) and (24)

$$V_N \sim K N^2 \quad (K \text{ is a constant}).$$

To insure that no overflow occurs in the computation, the input coefficients must have a modulus smaller than 1/N. We further assume, following Oppenheim & Weinstein[2] that the real and imaginary parts of input coefficients are independent random variables uniformely distributed in $\left[-1/\sqrt{2}N , 1/\sqrt{2}N \right]$ so that $\sigma^2 = 1/ 3 N^2$ and

$$E/S \sim 3 K N^2$$

Thus, in the case of no-scaling E/S varies as N^2 .

In the case of step-by-step scaling, we have roughly from (23)

and (25): $$V_N \sim K' N^3$$

But assuming again a uniform distribution of the input variables, now in $[-1 , +1]$ we have $\sigma^2 = 2/3$ and therefore

$$E/S \sim \frac{3}{2} K' N$$

Thus, in the case of step-by-step scaling, E/S has a linear variation with N. This shows, as expected, that the step-by-step scheme is in general much better.

4-4. Error analysis for fixed-point computation with truncation:

This analysis is very similar in its organization to the analysis for the case of rounding done in the previous section. However, new terms appear in the derivation due to the bias of truncation errors and also due to the correlations between errors and signs. We consider again successively the extreme cases of no-scaling and step-by-step scaling.

4-4-1. No-scaling:

Relations (1) and (2) are still valid but with $\varepsilon_{4\ell,k}$, $\varepsilon_{4\ell+1,k}$, $\varepsilon_{4\ell+2,k}$ and $\varepsilon_{4\ell+3,k}$ respectively replaced in (2) by $-sr_\ell \, sr_{k\ell} \, \delta_{4\ell,k}$, $-si_\ell \, si_{k\ell} \, \delta_{4\ell+1,k}$, $-sr_\ell \, si_{k\ell} \, \delta_{4\ell+2,k}$ and $-si_\ell \, sr_{k\ell} \, \delta_{4\ell+3,k}$ where sr_ℓ, si_ℓ, $sr_{k\ell}$ and $si_{k\ell}$ are the respective signs of $Re(V_\ell)$, $Im(V_\ell)$, $Re(T_{k\ell})$ and $Im(T_{k\ell})$. Then, with the same assumptions of statistical independence as in the case of rounding, we obtain for the mean square error $E_W(k)$ of W_k :

$$E_W(k) = \sum_{\ell=0}^{N-1} \|T_{k\ell}\|^2 \; E_V(\ell) + \mu_k (\Delta^2 + \mu^2) + E'_W(k) \qquad (26)$$

where the two first terms are similar to those of (3). The complementary term $E'_W(k)$, which comes from the crossmultiplied terms in $\mathcal{E}_r(W_k)$ and $\mathcal{E}_i(W_k)$, results from the correlations between the signs of the input coefficients and the errors on these coefficients. Denoting $n_{2\ell} = E(\, sr_\ell \;\; \mathcal{E}_r(V_\ell)\,)$ and $n_{2\ell+1} = E(\, si_\ell \;\; \mathcal{E}_i(V_\ell)\,)$, we have

$$E'_W(k) = -2\,\mu\left\{\sum_{\ell=0}^{N-1}(n_{2\ell}+n_{2\ell+1})\;\left|Re(T_{k\ell})\right| + \sum_{\ell=0}^{N-1}(n_{2\ell}+n_{2\ell+1})\;\left|Im(T_{k\ell})\right|\right\} \tag{27}$$

This expression simplifies if all input coefficients of any parent matrix have similar error statistics, i. e. $n_{2\ell} = n_{2\ell+1} = n$ independent of ℓ. This happens for all the IC transforms of chapter 3 and so for the usual transforms. Then, with

$$A_k = \sum_{\substack{\ell=0 \\ |Re(T_{k\ell})|\neq 1}}^{N-1} Re(T_{k\ell}) + \sum_{\substack{\ell=0 \\ |Im(T_{k\ell})|\neq 1}}^{N-1} Im(T_{k\ell})$$

we have:

$$E'_W(k) = -4\,\mu\, n\, A_k \tag{28}$$

As $\mathcal{E}_r(V_\ell)$ and $Re(V_\ell)$ are only correlated by their signs, we have:

$$n_{2\ell} = \frac{E(\,\mathcal{E}_r(V_\ell)\;\; Re(V_\ell)\,)}{E\left|\,Re(V_\ell)\right|}$$

and also

$$n_{2\ell+1} = \frac{E(\,\mathcal{E}_i(V_\ell)\;\; Im(V_\ell)\,)}{E\left|\,Im(V_\ell)\right|}$$

We study now a recursive relation for $n_{2\ell}$ and $n_{2\ell+1}$. From (1) and (26) we have :

$$\begin{aligned} Re(W_k)\;\mathcal{E}_r(W_k) = \sum_{\ell=0}^{N-1}\Big(\, & Re^2(T_{k\ell})\;Re(V_\ell)\;\mathcal{E}_r(V_\ell) + Im^2(T_{k\ell})\;Im(V_\ell)\,\mathcal{E}_i(V_\ell) \\ & - \left|Re(T_{k\ell})\right|\left|Re(V_\ell)\right|\delta_{4\ell,k} - \left|Im(T_{k\ell})\right|\;\left|Im(V_\ell)\right|\delta_{4\ell+1,k} \\ & + \text{terms with random signs}\;\Big) \end{aligned} \tag{29}$$

For simplicity let us denote $N_{2k} = E(\, sr_k \;\; \mathcal{E}_r(W_k))$,

$N_{2k+1} = E(\, si_k \;\; \mathcal{E}_i\,(W_k))$, $P_{2k} = E\left|\, Re(W_k)\right|$, $P_{2k+1} = E\left|\, Im(W_k)\right|$,

$p_{2\ell} = E\left|\, Re(V_\ell\,)\right|$ and $p_{2\ell+1} = E\left|\, Im(V_\ell\,)\right|$. Then taking the expectation of (29), we obtain:

$$N_{2k}\;P_{2k} = \sum_{\ell=0}^{N-1} \left(Re^2(T_{k\ell}\,)\;\; n_{2\ell}\; p_{2\ell} + Im^2\,(T_{k\ell}\,)\; n_{2\ell+1}\; p_{2\ell+1}\,\right) - \mu \sum_{\substack{\ell=0 \\ |Re(T_{k\ell})|\neq 1}}^{N-1} \left|Re(T_{k\ell}\,)\right| p_{2\ell} - \mu \sum_{\substack{=0 \\ |Im(T_{k\ell})|\neq 1}}^{N-1} \left|Im(T_{k\ell}\,)\right|\; p_{2\ell+1} \tag{30}$$

$N_{2k+1}\;\; P_{2k+1}$ gives a similar expression.

If $n_{2\ell} = n_{2\ell+1} = n$ and simutaneously $p_{2\ell} = p_{2\ell+1} = p$,

then (30) simplifies to

$$N_{2k+1}\;\; P_{2k+1} = N_{2k}\;\; P_{2k} = n\; p\; N - p\; A_k \tag{31}$$

In (31) , the computation of $p = E\left|Re(V_\ell\,)\right|$ requires the knowledge of the statistics of the input coefficients. For simplicity and also because the intermediate results tend to have a normal distribution (law of large numbers) we assume that the input coefficients have a zero-mean normal distribution with variance σ_V^2 . Then, from (1), we see that the output coefficients have also a normal distribution with variance $\sigma_W^2 = N\,\sigma_V^2$. Now we can compute $P = P_{2k} = P_{2k+1}$:

$$P = \frac{2}{\sqrt{2\pi\,\sigma_W^2}} \int_0^\infty x \exp(-\,x^2/\,2\,\sigma_W^2)\; dx = 2\,\sigma_W/\sqrt{2\pi}$$

Similarly $p = 2\,\sigma_V/\sqrt{2\pi}$ so that $P = \sqrt{N}\;\; p$ (32)

Then, from (31) and (32) , we have

$$N_{2k} = N_{2k+1} = \sqrt{N}\;\; n - \frac{\mu}{\sqrt{N}}\, A_k \tag{33}$$

To sum up this development, the truncation round-off errors are

obtained in general from the recursive relations (26), (27) , (30) and from the statistics of the intermediate results. In the particular case of importance where the input coefficients of any parent matrix have similar statistics such that $n_{2\ell} = n_{2\ell+1} = n$ and $p_{2\ell} = p_{2\ell+1} = p$, then (26) (28) and (33) give a simpler way to compute the round-off errors.

The propagation of the round-off errors in a generalized Kronecker product can be carried out as done for the case of rounding. We shall consider as an example only the FFT with Cooley-Tukey algorithm in section 4-4-3.

4-3-2. Step-by-step scaling:

The basic relation giving the round-off error on $\mathrm{Re}(W_k)$ is still given by (10) with the variables $\varepsilon_{4\ell,k}$, $\varepsilon_{4\ell+1,k}$, $\varepsilon_{4\ell+2,k}$ and $\varepsilon_{4\ell+3,k}$ modified as done in the previous section. The variables $\varepsilon'_{2\ell,k}$ and $\varepsilon'_{2\ell+1,k}$ have also to be replaced respectively by $-sr_\ell \ \varepsilon'_{2\ell,k}$ and $-si_\ell \ \varepsilon'_{2\ell+1,k}$. Then we obtain a relation for the mean square error of each coefficient similar to (26) and (27) :

$$
\begin{aligned}
E_W(k) = \sum_{\ell=0}^{N-1} \|T_{k\ell}\|^2 \left[E_V(\ell) + 2^{2S-2q}(\Delta'^2 + \mu'^2) \right] + \mu_k (\Delta^2 + \mu^2)\, 2^{2S} \\
+ 4\, 2^{2S-q}\, \mu\, \mu'\, A_k - 2\mu' \sum_{\ell=0}^{N-1} \|T_{k\ell}\|^2\, 2^{S-q}\, (n_{2\ell} + n_{2\ell+1}) \\
- 2\mu \left\{ \sum_{\substack{\ell=0 \\ |\mathrm{Re}(T_{k\ell})| \neq 1}}^{N-1} (n_{2\ell} + n_{2\ell+1}) \ |\mathrm{Re}(T_{k\ell})| + \sum_{\substack{\ell=0 \\ |\mathrm{Im}(T_{k\ell})| \neq 1}}^{N-1} (n_{2\ell} + n_{2\ell+1}) \ |\mathrm{Im}(T_{k\ell})| \right\}
\end{aligned}
\tag{34}
$$

where we have two new terms corresponding to the correlations between the scaling and truncation errors and between the signs of the input coefficients and the scaling errors. As for (27), (34) can simplify

when $n_{2\ell} = n_{2\ell+1} = n$. Then we have :

$$E_W(k) = \sum_{\ell=0}^{N-1} \| T_{k\ell} \|^2 \left[E_V(\ell) + 2^{2S-2q} (\Delta'^2 + \mu'^2) \right] + \mu_k (\Delta^2 + \mu^2) \, 2^{2S} + 4 \, 2^{2S-q} \, \mu \mu' A_k - 4 n (2^{S-q} N \mu' + 2^S \mu A_k) \tag{35}$$

The relation corresponding to (29) is now :

$$\begin{aligned} \mathrm{Re}(W_k) \, \mathcal{E}_r(W_k) = \sum_{\ell=0}^{N-1} \Big[& \mathrm{Re}^2(T_{k\ell}) \, (\mathrm{Re}(V_\ell) \, \mathcal{E}_r(V_\ell) - |\mathrm{Re}(V_\ell)| \, 2^{S-q} \, \sigma'_{2\ell,k}) \\ & + \mathrm{Im}^2(T_k) \, (\mathrm{Im}(V_\ell) \, \mathcal{E}_i(V_\ell) - |\mathrm{Im}(V_\ell)| \, 2^{S-q} \, \sigma'_{2\ell+1,k}) \\ & - |\mathrm{Re}(T_{k\ell})| \, |\mathrm{Re}(V_\ell)| \, \sigma_{4\ell,k} \, 2^S \\ & - |\mathrm{Im}(T_{k\ell})| \, \mathrm{Im}(V_\ell) \, \sigma_{4\ell+1,k} 2^S \\ & + \text{terms of random signs} \Big] \end{aligned}$$

giving similarly to (30) :

$$\begin{aligned} N_{2k} \, P_{2k} = \sum_{\ell=0}^{N-1} \Big(& \mathrm{Re}^2(T_{k\ell}) \, p_{2\ell} \, (n_{2\ell} - 2^{S-q} \, \mu') + \mathrm{Im}^2(T_{k\ell}) \, p_{2\ell+1} (n_{2\ell+1} - 2^{S-q} \, \mu') \Big) \\ & - \mu \sum_{\substack{\ell=0 \\ |\mathrm{Re}(T_{k\ell}) \neq 1}}^{N-1} \mathrm{Re}(T_{k\ell}) \, p_{2\ell} \, 2^S - \mu \sum_{\substack{\ell=0 \\ |\mathrm{Im}(T_{k\ell})| \neq 1}}^{N-1} \mathrm{Im}^2(T_{k\ell}) \, p_{2\ell+1} \, 2^S \end{aligned} \tag{36}$$

which, if $n_{2\ell} = n_{2\ell+1} = n$ and $p_{2\ell} = p_{2\ell+1} = p$, reduces to

$$N_{2k} \, P_{2k} = N \, n \, p - 2^{S-q} \, \mu' \, N \, p - \mu \, p \, A_k \tag{37}$$

And finally making use of (32),which is unchanged, and (37) we have:

$$N_{2k} = N_{2k+1} = \sqrt{N} \, n - \frac{1}{\sqrt{N}} \, (N \, 2^{S-q} \, \mu' + \mu \, 2^S \, A_k) \tag{38}$$

The relations (34), (36) or (35) and (37) when the simplifications apply, provide a set of recursive equations to compute the truncation errors in this case of step-by-step scaling. In the next section we consider, as an example, the FFT with Cooley-Tukey algorithm.

4-4-3. Application to FFT Cooley-Tukey algorithm:

As an example, we consider the round-off error for the FFT with Cooley-Tukey algorithm in the case of truncation. In order to obtain closed form results we shall introduce some approximations in our computation ; they are not however strictly necessary.

We first compute for a Fourier parent matrix the coefficient A_k :

$$A_k = |\text{Cos } \theta_k| + |\text{Sin } \theta_k| \qquad \theta \neq 0 \text{ or } \pi/2$$

At the n th stage of computation the average value of A_k , denoted A^n is

$$A^n = \frac{8}{2^n} \sum_{k=1}^{2^{n-2}-1} \text{Cos } (2\pi k/ 2^n) = \frac{8}{2^n} \left(\frac{\text{Sin}(2\pi/2^n)}{2 - \text{Cos}(2\pi/2^n)} - \tfrac{1}{2}\right)$$

For n large , A^n tends to a limit $A^{\#} = \frac{8}{2\pi} \sim 1.27$

For simplicity, we shall replace A_k by the constant $A^{\#}$ in the following. We consider now the cases of no-scaling and step-by-step scaling.

a) No-scaling:

We first solve the recursive relation (33) which is now

$$N_n = 2\, N_{n-1} - \frac{2^{-b} A^{\#}}{2\sqrt{2}} \qquad \text{giving} \qquad N_n = - \frac{2^{-b} A^{\#} (\sqrt{2})^{n-2}}{2 \sqrt{2} (\sqrt{2} -1)}$$

Then the recursive use of (26) and (28) gives the following recursive relation for the total mean square error:

$$V_{2^n} = 4\, V_{2^{n-1}} + (2^n -4)\, 2^{-2b} (1/3 +1) + 2^{-2b} \frac{A^{\#2}\, 2^{\frac{3n-2}{2}}}{(\sqrt{2} -1)\sqrt{2}} \qquad (39a)$$

giving approximatly $V_{2^n} = 2^{2n}\, 2^{-b}\, (1.5)$

about 10 times the rounding error but with the same variation on N^2.

b) Step-by-step scaling:

The recursive relation (38) gives now:

$$N_n = \sqrt{2}\, N_{n-1} - \frac{1}{\sqrt{2}}\, 2^{n-1}\, 2^{-b}\, (1 + A^{\#})$$

Hence

$$N_n = \frac{2^{-b}\, 2^n\, (1 + A^{\#})}{4\,(\sqrt{2} - 1)}$$

Then, from (35), we obtain the following recursive relation:

$$V_{2^n} = 4\, V_{2^{n-1}} + (2^n - 4)\, 4^n\, 2^{-2b}\, (1/3 + 1) + 2^{3n}\, 2^{-2b} \left(\tfrac{1}{2} + \frac{A^{\#}}{2} + \frac{(1 + A^{\#})^2}{2\,(\sqrt{2} - 1)}\right) \qquad (39b)$$

Thus

$$V_{2^n} \simeq 2^{3n+1}\, 2^{-2b} \quad (8.7) \qquad (39)$$

which is also about 10 times the result obtained for rounding.

On Figure 4-1 we have plotted the experimental results obtained by Weinstein [3] with white signals and the approximate theoretical results of (39). Our approximation in the value of A_k is responsible for the slight discrepancy between the results. The exact theoretical error estimation done by computer computation gives the results also plotted on Figure 4-1 and which match perfectly the experimental results.

Our result suggests that the output mean square error should be drastically reduced if the correlations between signal signs and errors could be suppressed, for example by leaving random the last bit of each truncated number.

4-5. Block-scaling with rounding or truncation:

In the previous sections, we have considered the two extremes cases of no-scaling and step-by-step scaling. In both cases the algorithm is independent of the data and we have derived bounds valid for any input vector. These extreme cases do not correspond however to usual situations. A commonly used fixed-point algorithm is block-scaling: at every multiplication overflow is checked and, whenever it occurs, all intermediate results at this stage of computation are equally scaled. The cummulative number of shifts is counted and gives the scaling factor common to all output coefficients. For a known scaling sequence, the recursive relations for the mean square error in a block-scaling computation are easely deduced from the results of the previous sections. However the scaling sequence for a given set of data is difficult to know a priori. Weinstein [3] has measured the probability of each possible sequence with 500 sets of 512 independent complex numbers with real and imaginary parts uniformely distributed in $[-1 , +1]$ for the FFT with Cooley-Tukey algorithm. Such approach to determine the probability of each scaling sequence requires repeated computations for each transform and block size of interest. We propose a more general approach.

In order to obtain the probability of overflow at each stage, we assume that each input coefficient V_ℓ has independent normal real and imaginary parts with variance σ^2. Any intermediate result at the k th stage of computation is a weighted sum of N_k input coefficients and the weights $\gamma_{i\ell}$ are normalized to the same factor N_k:

$$\sum_{\ell=0}^{N_k-1} \|\gamma_{i\ell}\|^2 = N_k$$

It is easy to show that the intermediate results, denoted Y_i, are independent and normally distributed with variances

$$\mathrm{Var}(\mathrm{Re}(Y_i)) = \mathrm{Var}(\mathrm{Im}(Y_i)) = \sigma^2 N_k$$

The probability that one overflow occurs for one of these intermediate results when S scalings have already been performed at previous stages is $P_0 = \mathrm{erf}\left(\frac{2^S}{\sqrt{2 N_k \sigma^2}}\right)$ (see footnote 3) and so the probability $P_S(k)$ that no new scaling is necessary during the k th stage of computation when S have been performed is :

$$P_S(k) = (P_0)^{P_k} = \left[\mathrm{erf}\left(\frac{2^S}{\sqrt{2 N_k \sigma^2}}\right)\right]^{P_k} \qquad (40)$$

where P_k is the number of intermediate results at this stage.
It is interesting to note that, with this model, the scaling factor is known with a high probability (see Weinstein [3]) and also that the number of scaling sequences with a significant probability is low. One might expect therefore good approximations when averaging over the possible scaling sequences.

Then for a transform defined with the recursive generative rules of Chapter 3, we can compute an expected mean square error for each output coefficient and an expected total mean square error. This result will apply also to all transforms giving the same recursive relations for the errors in case of no-scaling and step-by-step scaling. For each scaling sequence we compute the corresponding mean square errors. Then, using (40), we take an average of all the mean square errors with their corresponding probabilities. As an example , we apply this method to

---

3

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\, dt$$

the FFT with Cooley-Tukey algorithm for rounding and truncation.

a) Rounding:

The mean square errors at each stage are given recursively by (4) and (5) if no scaling is performed, (12) and(13) if a scaling is performed. In particular for the FFT with Cooley-Tukey algorithm we obtain the total mean square error from (22 a) and (23a). We have plotted on Figure 4-2 the theoretical points we obtain along with the experimental results (for white uniformely distributed input coefficients) reported by Weinstein [3] . For N sufficiently large our results are quite accurate.

b) Truncation:

The mean square errors are now computed from (26) (28)(or (26) (28) when the simplifications apply) when no scaling is performed and from (34) or (35) when a scaling is performed. However we need also to compute the random variables $N_{2\ell}$ and $N_{2\ell+1}$ from the relations (30) or (33) with no scaling and (36) or (37) with a scaling. In particular for the FFT with Cooley-Tukey algorithm, we have an approximate computation by using the relations(39 a) and (39 b). We have plotted the corresponding points on Figure 4-2 along with the experimental results by Weinstein [3] . Again we find a good agreement for N sufficiently large.

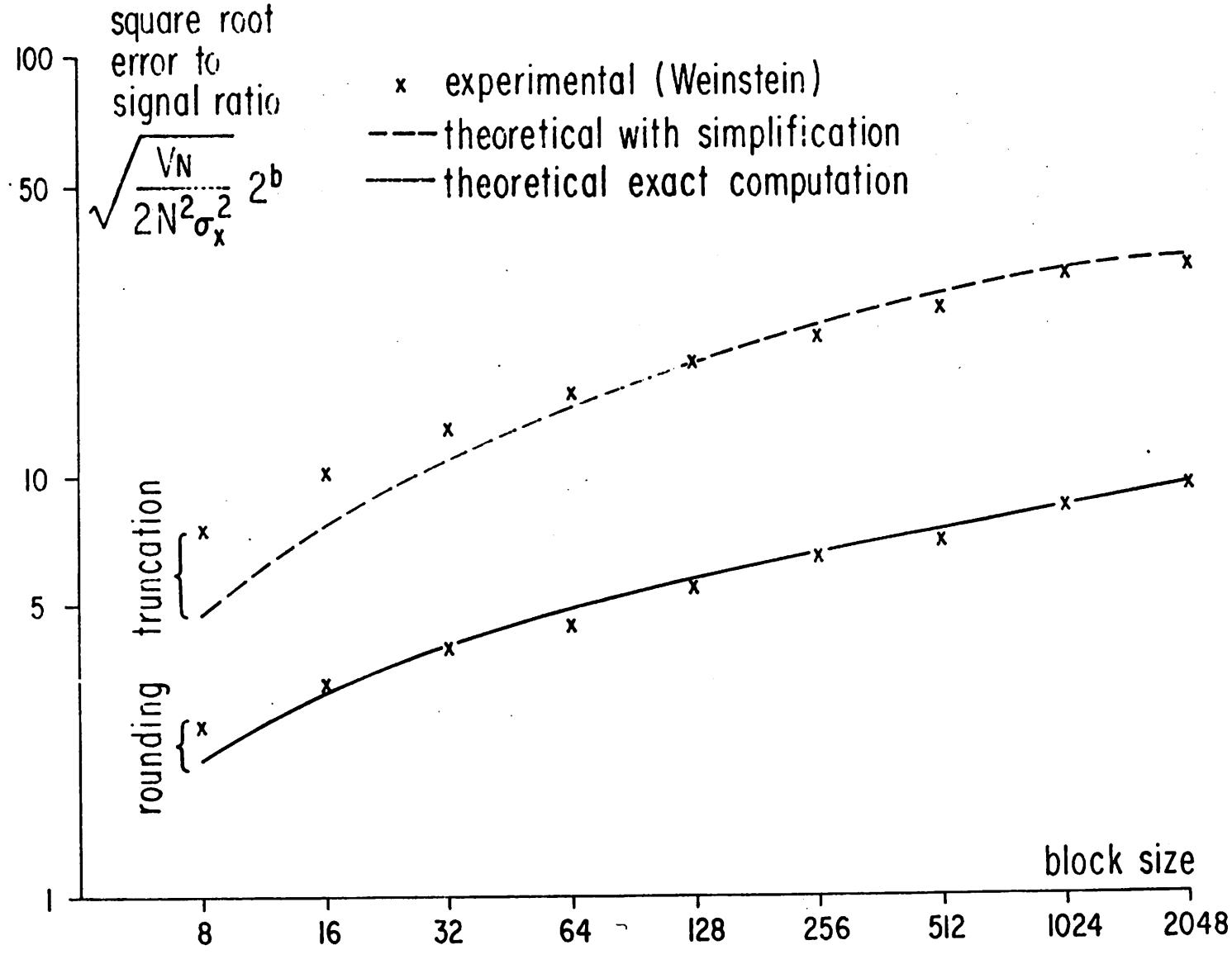


Fig. 4-2. FFT fixed-point error analysis with block scaling

4-6. Conclusions:

In conclusion we review the results derived in this chapter and comment upon their interest in practical situations. For transforms which give independent errors on the input coefficients of every parent matrix, we have developed a general method to compute the individual and total mean square errors in the case of rounded computation. This restriction on the transforms does not seem very stringent : the previous condition is verified for the very large family of IC transforms and for other transforms it is possible that the correlations between error terms do not introduce a significant output error. In case of truncation our derivation required also a model for the intermediate results ; to obtain simple recursive relations we have further assumed some similar statistical properties for the errors on intermediate results and also for these intermediate results.

Our method is a direct application of the recursive generation of fast unitary transforms, developed in chapter 3, to the error models for fixed-point computations. In the following chapter, we use the same method but with floating-point error models.

We have applied our method to the FFT algorithms and our results fit closely the experimental results reported for white uniformely distributed data. We feel confident that for other transforms and other data this method of error analysis will provide meaningful results. However it should be noted that specific type of data may not verify the error model and one should be careful in using our results or method.

C H A P T E R V

ERROR ANALYSIS IN FLOATING-POINT COMPUTATION

5-1. Introduction:

In chapter 4, the error analysis of fast algorithms in fixed-point computation is based on the error models and the recursive definition of fast transforms developed in chapter 3 . The analysis was successful because the error models used were consistent with the recursive relations, in the sense that the errors also satisfied recursive relations. In this chapter, we derive similar results for error models valid in floating-point computations. The main difference is the dependence of every round-off error on the represented data. Thus in this chapter, we shall rely heavily on bounds and on statistical models for the data.

Previous work has been mainly concerned with FFT algorithms. Kaneko and Liu [1] were able to carry out an exact study of the FFT Sande-Tukey algorithm for rounding as well as for truncation errors in computation. Weinstein [2] [3] ,Oppenheim and Weinstein [4] have modelled the input vector as a white signal and studied the total mean square error due to rounding for the FFT Cooley-Tukey algorithm. Weinstein reported also some experimental results for errors due to truncation. Chan and Jury [5] extended some of these results to the Generalized discrete transforms (see chapter 3) which include the Walsh-Hadamard transform. Gentleman and Sande [6] and also Ramos [7] have derived bounds on errors for the FFT algorithms. Ramos [7] ,

Chan and Jury [5] have also extended some of these results to multidimensional transforms.

In this chapter, we follow the same steps as for the fixed-point computation: we first develop the error model used and also discuss the importance of the organization of a dot product, the basic operation in fast algorithms. Then, we study the generation of errors in parent matrices and finally how these errors combine in the generative rules of chapter 3 . We study successively the cases of rounding and truncation errors.

To conclude the error analysis, we comment briefly on the effect of errors in the representation of the transform entries or factors of the computation. These errors may not be negligeable for some floating-point computations.

5-2. Error models in floating-point computations and organization of dot products:

5-2-1. Error models:

Errors in floating-point computations appear whenever a result is obtained in an arithmetic operation. Let fl(a) denote the floating-point approximation of the operand a. Wilkinson [8] has shown that[1]:

$$fl(a) = a\,(1 + \alpha) \quad \text{with}$$

$$\alpha \in [-2^{-b},\ 2^{-b}] \quad \text{for rounding}$$

$$\alpha \in [-2^{-b+1},\ 0] \quad \text{for truncation}$$

In the following, the analysis requires only the mean and variance

[1] We assume here a base 2 representation as done in most computers. For other bases, our results will apply if the error model is still valid.

of the random variable α . With the assumption of a uniform distribution of α in its interval of variation, we find the following values:

rounding: zero-mean, variance $\varepsilon^2 = 2^{-2b}/3$

truncation: mean $\mu = -2^{-b}$, variance ε^2.

Unfortunately, a uniform distribution of errors is not a good model since usually the input numbers have their values concentrated in a small interval of the total range of the floating-point numbers. Additions and multiplications by coefficients also within a small interval of variation results in higher probability of occurence of some numbers. An exact study of the distribution of α is quite complex [9] [10] and no simple model valid in most circumstances seems feasible. In the numerical applications of our results, we shall use the experimental values obtained by Weinstein [3] for the variance and Liu & Kaneko [11] for the truncation mean:

rounding: zero-mean, variance $\varepsilon^2 = 0.23\ 2^{-2b}$

truncation: mean $\mu = -\ 0.26\ 2^{-b}$, variance ε^2.

Experiments on the FFT algorithms reported by Gentleman & Sande [6] and Weinstein [3] show a discrepancy with their theoretical analyses when usual rounding is used. Weinstein [2] [3] showed that a randomized rounding of the midway[2] point yields experimental results in agreement with the previous error estimates but no theoretical model has explained the usual rounding situation. Here we use the experimental observation by Liu and Kaneko [11] that the midway point has a higher probability of occurence in additions. Our theoretical work, based on this model, gives error estimates in good agreement with experimental results.

---

[2] Value half way between quantization values.

5-2-2. Floating-point dot products computations:

Consider a parent matrix vector multiplication

$$\vec{W} = [T] \quad \vec{V} \qquad \text{or} \tag{1}$$

$$\begin{cases} Re(W_k) = \sum_{\ell=0}^{N-1} \left(Re(T_{k\ell}) \; Re(V_\ell) - Im(T_{k\ell}) \; Im(V_\ell) \right) & (1a) \\ Im(W_k) = \sum_{\ell=0}^{N-1} \left(Im(T_{k\ell}) \; Re(V_\ell) + Re(T_{k\ell}) \; Im(V_\ell) \right) & (1b) \end{cases}$$

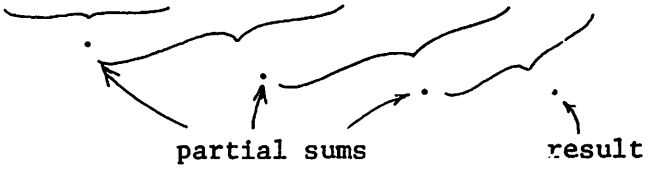
The computation of a dot product is the basic operation : there are many different ways to compute a dot product each giving a different error.

a) We may round-off after each multiplication such as $Re(T_{k\ell})\; Re(V_\ell)$ in (1a) and (1b), and then perform the additions in full precision rounding or truncating only to store the result of the dot product. We introduce then an error α such that:

$$fl(\text{dot product}) = (\text{sum of approximate products}) \; (1 + \alpha)$$

b) Assume that we have adders with f operands (usually 2) to add the p terms of the dot product addition. Many choices are open for the organization of this addition. The usual way uses only one accumulator to store the partial sum and the computation is performed as follows (we take f = 3 and p = 8 in the following diagram):

terms to sum : $a_1 \; a_2 \; a_3 \; a_4 \; a_5 \; a_6 \; a_7 \; a_8$



If we have several accumulators (or temporary storage register locations), we can organize the computation in a tree-like manner as follows:

5-2-2. Floating-point dot products computations:

Consider a parent matrix vector multiplication

$$\vec{W} = [T] \quad \vec{V} \qquad \text{or} \tag{1}$$

$$\begin{cases} Re(W_k) = \sum_{\ell=0}^{N-1} \left(Re(T_{k\ell}) \; Re(V_\ell) - Im(T_{k\ell}) \; Im(V_\ell) \right) & (1a) \\ Im(W_k) = \sum_{\ell=0}^{N-1} \left(Im(T_{k\ell}) \; Re(V_\ell) + Re(T_{k\ell}) \; Im(V_\ell) \right) & (1b) \end{cases}$$

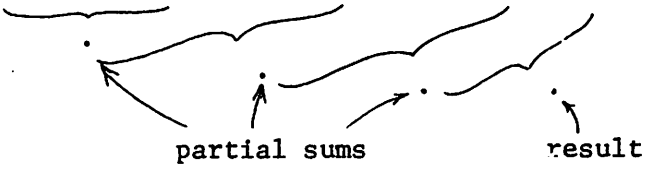
The computation of a dot product is the basic operation : there are many different ways to compute a dot product each giving a different error.

a) We may round-off after each multiplication such as $Re(T_{k\ell})\; Re(V_\ell)$ in (1a) and (1b), and then perform the additions in full precision rounding or truncating only to store the result of the dot product. We introduce then an error α such that:

$$fl(\text{dot product}) = (\text{sum of approximate products}) \; (1 + \alpha)$$

b) Assume that we have adders with f operands (usually 2) to add the p terms of the dot product addition. Many choices are open for the organization of this addition. The usual way uses only one accumulator to store the partial sum and the computation is performed as follows (we take f = 3 and p = 8 in the following diagram):

terms to sum : $a_1 \; a_2 \; a_3 \; a_4 \; a_5 \; a_6 \; a_7 \; a_8$



If we have several accumulators (or temporary storage register locations), we can organize the computation in a tree-like manner as follows:

terms to sum: a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8

partial sums

result

The number of elementary operations is usually the same but the error expression is different. Assume that an elementary operation introduces an error δ such that :

$$fl(a_1 + \ldots\ldots + a_f) = (a_1 + \ldots\ldots + a_f)(1 + \delta) \quad \text{with}$$
$$\delta \in [-2^{-b}, 2^{-b}]$$

In the first case, we have:

$$fl(a_1+\ldots+a_8) = fl(fl(fl(fl(a_1+a_2+a_3)+a_4+a_5)+a_6+a_7)+a_8)$$

$$=((((a_1+a_2+a_3)(1+\delta_1)+a_4+a_5)(1+\delta_2)+a_6+a_7)(1+\delta_3)+a_8)(1+\delta_4)$$

$$= (a_1+a_2+a_3)(1+\delta_1)(1+\delta_2)(1+\delta_3)(1+\delta_4)$$
$$+(a_4+a_5)(1+\delta_2)(1+\delta_3)(1+\delta_4)$$
$$+(a_6+a_7)(1+\delta_3)(1+\delta_4) + a_8(1+\delta_4)$$

In the second case, we have:

$$fl(a_1+\ldots\ldots+a_8) = fl(fl(a_1+a_2+a_3) + fl(a_4+a_5+a_6) + fl(a_7+a_8))$$

$$= (a_1+a_2+a_3)(1+\delta_1)(1+\delta_4) + (a_4+a_5+a_6)(1+\delta_2)(1+\delta_4)$$
$$+ (a_7+a_8)(1+\delta_3)(1+\delta_4)$$

Except for the common parent matrices of order 2, for which there is no choice to organize the dot product, the error analysis of a matrix vector multiplication and from there the error analysis of a fast transform, depends heavily on the type of adders and the organization

of dot products as shown on the previous example.

In the following, we consider only the case of adders with two operands and of dot products performed step-by-step. We also assume that rounding is performed just after any addition or multiplication. Other cases of organization of the computation can be studied as in the following sections but they will yield quite different results.

5-3. Floating-point computations with rounding: analysis of errors in parent matrix operations:

Following the approaches of previous workers mentioned in 5-1 but with the steps of the previous chapter, we express in this section the mean square error of each output coefficient or the total mean square error of the output vector, first in a parent matrix operation.

5-3-1. Error analysis in parent matrices:

a) Direct norm bounds:

Let us consider the matrix relation between real vectors

$$\vec{Y} = [T] \quad \vec{X}$$

where $[T]$ is a real unitary transform. Wilkinson [8] has established the following norm relations when the input vector is error free:

$$\| Y \| \leqslant \| T \| \; \| X \| \tag{2}$$

and

$$\| E_Y \| \leqslant 1.06 \; N \; \varepsilon \; \| T \| \; \| X \| \tag{3}$$

where $\| X \| = \left(\sum_i X_i^2 \right)^{\frac{1}{2}}$ $\quad \| T \| = \left(\sum_i \sum_j T_{ij}^2 \right)^{\frac{1}{2}}$

$\vec{E}_Y = \hat{\vec{Y}} - \vec{Y}$ ($\hat{\vec{Y}}$ computed value of $\vec{Y}$)

ε^2 = rounding error variance

As $[T]$ is unitary, $\|T\| = \sqrt{N}$ so that:

$$\|E_Y\| \leq 1.06\ \varepsilon\ N^{3/2}\ \|X\| \tag{4}$$

In case of a complex vector relation $\vec{W} = [T]\ \vec{V}$, we write:

$$\begin{pmatrix} \mathrm{Re}(W) \\ \mathrm{Im}(W) \end{pmatrix} = \begin{bmatrix} \mathrm{Re}(T) & -\mathrm{Im}(T) \\ \mathrm{Im}(T) & \mathrm{Re}(T) \end{bmatrix} \begin{pmatrix} \mathrm{Re}(V) \\ \mathrm{Im}(V) \end{pmatrix}$$

so that $$\|E_W\| \leq 1.06\ \varepsilon\ 2\ N\ \sqrt{2N}\ \|V\| \tag{5}$$

This result was used by Gentleman and Sande [6]. If the matrix $[T]$ is not normalized by $1/\sqrt{N}$, then (5) becomes:

$$\|E_W\| \leq 1.06\ 2\sqrt{2}\ \varepsilon\ N^2\ \|V\| \tag{6}$$

This bound is independent of the parent matrix and of the operations involved in the computation: thus, the bounds obtained by combination of parent matrices will depend only on the orders of these parent matrices. However the gain in generality is paid for in the looseness of the bound. We note finally that relation (3) has been applied above to the matrix relation corresponding to the rotation by a parent matrix. It can be used for any matrix relation and later in this section, we shall use it again.

b) Expression of the error:

Here, instead of direct norm bounds, we evaluate exactly the error at each output coefficient. Then, we simplify the expression of the error with norm bounds or with a statistical model for the input coefficients. If round-off errors occur in the computation of dot products, then relations (1a) and (1b) become, with $\hat{}$ denoting approximate values (we have assumed that there is no error in the expression of $\vec{V}$) :

$$\widehat{Re(W_k)} = \prod_{j=1}^{N-1}(1+\delta_j)\left\{\left[Re(T_{k1})Re(V_1)(1+\alpha_1) - Im(T_{k1})Im(V_1)(1+\beta_1)\right](1+\gamma_1)\right\}$$
$$+ \sum_{i=2}^{N}\prod_{j=i-1}^{N-1}(1+\delta_j)\left\{\left[Re(T_{ki})Re(V_i)(1+\alpha_i) - Im(T_{ki})Im(V_i)(1+\beta_i)\right](1+\gamma_i)\right\}$$

(7)

$$\widehat{Im(W_k)} = \prod_{j=1}^{N-1}(1+\delta'_j)\left\{\left[Im(T_{k1})Re(V_1)(1+\alpha'_1) + Re(T_{k1})Im(V_1)(1+\beta'_1)\right](1+\gamma'_1)\right\}$$
$$+ \sum_{i=2}^{N}\prod_{j=i-1}^{N-1}(1+\delta'_j)\left\{\left[Im(T_{ki})Re(V_i)(1+\alpha'_i) + Re(T_{ki})Im(V_i)(1+\beta'_i)\right](1+\gamma'_i)\right\}$$

where α_i, β'_i (α'_i, β_i) are multiplication rounding errors which are zero if $Re(T_{ki}) = \pm 1$ ($Im(T_{ki}) = \pm 1$).

γ_i, γ'_i are addition **rounding** errors in the complex multiplications and are zero if T_{ki} is real or purely imaginary.

δ_j, δ'_j are also addition rounding errors in the dot product computation , organized with step-by-step additions of two operands. We assume here that T_{k1} has non null real or imaginary part to obviate unnecessarly complicated expressions. δ_j (δ'_j) are zero if $Re(T_{k\ell}) = 0$ ($Im(T_{k\ell}) = 0$).

Relation (7) can be written :

$$\begin{pmatrix} \widehat{Re(W)} \\ \widehat{Im(W)} \end{pmatrix} = \left[[F] \odot [\mathcal{C}]\right] \begin{pmatrix} Re(V) \\ Im(V) \end{pmatrix} \qquad (8)$$

in which $\odot$ denotes the direct product of matrices (each entry is multiplied the corresponding entry).

$[F]$ is the $2N \times 2N$ matrix such that :

$$F_{k1} = \prod_{j=1}^{N-1}(1+\delta_j)(1+\gamma_1)(1+\alpha_1) \qquad k = 1,\ldots,N$$

$$F_{ki} = \prod_{j=i-1}^{N-1}(1+\delta_j)(1+\gamma_i)(1+\alpha_i) \qquad i = 2,\ldots,N$$

$$
\begin{aligned}
F_{k,2^n+1} &= \prod_{j=1}^{N-1} (1+\delta_j)(1+\gamma_1)(1+\beta_1) \\
F_{k,2^n+i} &= \prod_{j=i-1}^{N-1} (1+\delta_j)(1+\gamma_i)(1+\beta_i) \\
F_{2^n+k,1} &= \prod_{j=1}^{N-1} (1+\delta'_j)(1+\gamma'_1)(1+\alpha'_1) \\
F_{2^n+k,i} &= \prod_{j=i-1}^{N-1} (1+\delta'_j)(1+\gamma'_i)(1+\alpha'_i) \\
F_{2^n+k,2^n+1} &= \prod_{j=1}^{N-1} (1+\delta'_j)(1+\gamma'_1)(1+\beta'_1) \\
F_{2^n+k,2^n+i} &= \prod_{j=i-1}^{N-1} (1+\delta'_j)(1+\gamma'_i)(1+\beta'_i)
\end{aligned}
\qquad (9)
$$

and

$$
[\mathcal{C}] = \begin{bmatrix} \mathrm{Re}([T]) & -\mathrm{Im}([T]) \\ \mathrm{Im}([T]) & \mathrm{Re}([T]) \end{bmatrix}
$$

Then (1) can be written:

$$
\begin{pmatrix} \mathrm{Re}(W) \\ \mathrm{Im}(W) \end{pmatrix} = [\mathcal{C}] \begin{pmatrix} \mathrm{Re}(V) \\ \mathrm{Im}(V) \end{pmatrix} \qquad (10)
$$

and, from (8) and (10), we obtain:

$$
\begin{pmatrix} \mathrm{Re}(E_W) \\ \mathrm{Im}(E_W) \end{pmatrix} = \begin{pmatrix} \mathrm{Re}(\hat{W} - W) \\ \mathrm{Im}(W - W) \end{pmatrix} = \underbrace{\left[[F] - [1]\right]}_{[\mathcal{E}]} \odot [\mathcal{C}] \begin{pmatrix} \mathrm{Re}(V) \\ \mathrm{Im}(V) \end{pmatrix} \qquad (11)
$$

where $[1]$ is the $2N \times 2N$ matrix with all entries equal to 1. The error matrix $[\mathcal{E}]$ of (11) takes a simple expression if we neglect in (9) the second order errors:

$$
\mathcal{E}_{k,1} = (\alpha_1 + \gamma_1 + \sum_{j=1}^{N-1} \delta_j) \quad \mathrm{Re}(T_{k1}) \qquad \text{N+1 error terms at most}
$$

$$E^{K,2n+1} = \prod_{j=1}^{N-1} (1+\rho^j)(1+\lambda^1)(1+\beta^1)$$

$$E^{K,2n+i} = \prod_{j=i-1}^{N-1} (1+\rho^j)(1+\lambda^1)(1+\beta^1)$$

$$E^{2n+K,1} = \prod_{j=1}^{N-1} (1+\rho_i^j)(1+\lambda_i^1)(1+\alpha_i^1) \qquad (9)$$

$$E^{2n+K,i} = \prod_{j=i-1}^{N-1} (1+\rho_i^j)(1+\lambda_i^1)(1+\alpha_i^1)$$

$$E^{2n+K,2n+1} = \prod_{j=1}^{N-1} (1+\rho_i^j)(1+\lambda_i^1)(1+\beta_i^1)$$

$$E^{2n+K,2n+i} = \prod_{j=i-1}^{N-1} (1+\rho_i^j)(1+\lambda_i^1)(1+\beta_i^1)$$

and

$$[G] = \begin{bmatrix} Re([T]) & -Im([T]) \\ Im([T]) & Re([T]) \end{bmatrix}$$

Then (1) can be written:

$$\begin{pmatrix} Re(W) \\ Im(W) \end{pmatrix} = [G] \begin{pmatrix} Re(V) \\ Im(V) \end{pmatrix} \qquad (10)$$

and, from (8) and (10), we ob . .

$$\begin{pmatrix} Re(E^W) \\ Im(E^W) \end{pmatrix} = \begin{pmatrix} Re(W - \hat{W}) \\ Im(W - W) \end{pmatrix} - \underbrace{\left[[E] - [1] \right]}_{[\varepsilon]} \circ [G] \begin{pmatrix} Re(V) \\ Im(V) \end{pmatrix} \qquad (11)$$

where $[1]$ is the 2N x 2N matrix with all entries equal to 1. The error matrix $[\varepsilon]$ of (11) takes a simple expression if we neglect in (9) the second order errors:

$$\varepsilon^{K,1} = (\alpha^1 + \lambda^1 + \sum_{j=1}^{N-1} \rho^j) \quad Re(T^{K1}) \qquad \text{N+1 error terms at most}$$

$$
\begin{aligned}
\mathcal{E}_{k,i} &= \left(\alpha_i + \gamma_i + \sum_{j=i-1}^{N-1} \sigma_j\right) \mathrm{Re}(T_{ki}) && N+2-i \text{ error terms at most} \\
\mathcal{E}_{k,2^n+1} &= \left(\beta_1 + \gamma_1 + \sum_{j=1}^{N-1} \sigma_j\right) \left[-\mathrm{Im}(T_{k1})\right] && N+1 \quad " \\
\mathcal{E}_{k,2^n+i} &= \left(\beta_i + \gamma_i + \sum_{j=i-1}^{N-1} \sigma_j\right) \left[-\mathrm{Im}(T_{ki})\right] && N+2-i \quad " \\
\mathcal{E}_{2^n+k,1} &= \left(\alpha'_1 + \gamma'_1 + \sum_{j=1}^{N-1} \sigma'_j\right) \mathrm{Im}(T_{k1}) && N+1 \quad " \\
\mathcal{E}_{2^n+k,i} &= \left(\alpha'_i + \gamma'_i + \sum_{j=i-1}^{N-1} \sigma'_j\right) \mathrm{Im}(T_{ki}) && N+2-i \quad " \\
\mathcal{E}_{2^n+k,2^n+1} &= \left(\beta'_1 + \gamma'_1 + \sum_{j=1}^{N-1} \sigma'_j\right) \mathrm{Re}(T_{k1}) && N+1 \quad " \\
\mathcal{E}_{2^n+k,2^n+i} &= \left(\beta'_i + \gamma'_i + \sum_{j=i-1}^{N-1} \sigma'_j\right) \mathrm{Re}(T_{ki}) && N+2-i \quad "
\end{aligned}
\tag{12}
$$

The general expression (11) with the simplifications of (12) lead to expressions for the output errors which depend on the values of the input coefficients. For later use in recursive relations, we wish to obtain expressions depending only on the magnitude of this input vector. To reach this result, we consider two approaches:

- computing bounds
- assuming a statistical model.

c) Rounding mean square error:

All the random variables $\alpha, \beta, \gamma, \sigma$ appearing in the expression of $[\mathcal{E}]$ are then zero mean, so that, from (11) and (12),

$$E(\vec{E_W}) = 0$$

To estimate the mean square error, we first apply the norm relation (2)

to relations (11) and (12) . We obtain a bound for $\| E_W \|$ which is function of the random variables α_i, β_i.....etc. Taking the expectation value over this bound and considering that no error variable is null, we obtain an upper bound for the total mean square error of the output vector. The computation of the matrix norm $\| \mathcal{E} \|$ involves only the number of error terms appearing in (12):

$$E \| E_W \|^2 < \underbrace{\mathcal{E}^2}_{\text{error variance}} 4 N \underbrace{\left\{ (N+1)^2 + (N+1)^2 + N^2 + \ldots\ldots + 4 \right\}}_{\text{matrix norm}} \underbrace{\| V \|^2}_{\text{input vector norm}} \qquad (13)$$

Thus $$E \| E_W \| < 2 \mathcal{E} N^2 \| V \|$$

This bound was obtained by Ramos [7] with a slightly different computation.

We may also express the error variance of each coefficient from relations (11) and (12), considering now the null terms among the error variables :

$$\begin{aligned} E \| E_{W_k} \|^2 = \mathcal{E}^2 \Bigg\{ & \sum_{\substack{i \\ |Re(T_{ki})| \neq 1}} Re^2(T_{ki}) \| V_i \|^2 + \sum_{\substack{i \\ |Im(T_{ki})| \neq 1}} Im^2(T_{ki}) \| V_i \|^2 \\ & + \sum_{i \in Q_k} \| T_{ki} \; V_i \|^2 \\ & + \sum_{\substack{i=1 \\ T_{ki} \neq 0}}^{N-1} \left\| \sum_{\ell=1}^{i+1} (T_{k\ell} \; V_\ell) \right\|^2 \Bigg\} \end{aligned} \qquad (14)$$

where Q_k is the set of indexes i such that T_{ki} is not real or purely imaginary. The terms of (14) correspond respectively to the error variables α and β' , α' and β , γ and γ' , δ and δ'. In (14), $\vec{V}$ will appear only through its magnitude in the few particular cases where $\| V \|^2$ can be factored out in the two last terms. One

such case of interest is the FFT of radix 2 performed by the Sande-Tukey algorithm. Since the rows of $[T]$ are then:

$$\text{either} \quad \begin{cases} 1 & \pm 1 \\ j & \pm j \end{cases} \qquad \text{giving} \quad E\,\|E_{W_k}\|^2 = 2\,\epsilon^2\,\|V\|^2 = \epsilon^2\,\|W\|^2$$

$$\text{or} \quad e^{-j} \quad \pm e^{-j} \qquad \text{giving} \quad E\,\|E_{W_k}\|^2 = \epsilon^2\,(2\,\|V\|^2+2\|V\|^2+2\|V\|^2) = 3\,\epsilon^2\,\|W\|^2 \tag{15}$$

This result makes possible the work by Kaneko & Liu [1] for this particular transform. It is also valid for the W-H transform as shown by Chan & Jury [5] and for a limited number of transforms such as the WHH family of chapter 3. However there are many transforms of interest, for example the FFT Cooley-Tukey algorithm, which do not have this simplification. For them the following statistical approach will yield the desired property. We note finally that, by taking summations in (14) over all indexes including those which do not give any error, we obtain the norm bound of the previous section.

The second approach is to assume that the input signal is white with variance σ^2. Then, we have :

$$E(V) = 0$$

$$E(Re^2(V_i) = E(Im^2(V_i) = \sigma^2/2$$

$$E(Re(V_i)\ Im(V_i)) = 0 \text{ for all i and j}$$

These relations imply similar relations for $\vec{W}$ showing that $\vec{W}$ is also white with variance σ^2 if $[T]$ is normalized, $N\sigma^2$ otherwise. Then, taking the expectation value of (14) over the random variables V_i, we obtain:

$$E\left\|E_{W_k}\right\|^2 = \varepsilon^2\,\sigma^2\left\{\sum_i \mathrm{Re}^2(T_{ki}) + \sum_i \mathrm{Im}^2(T_{ki}) + \sum_{i\in Q_k}\left\|T_{ki}\right\|^2 + \sum_{i=1}^{N-1}\left\|\sum_{\ell=1}^{i+1} T_{k\ell}\right\|^2\right\} \tag{16}$$

$$|\mathrm{Re}(T_{ki})|\neq 1 \qquad |\mathrm{Im}(T_{ki})|\neq 1 \qquad T_{ki}\neq 0$$

which can be written $E\left\|E_{W_k}\right\|^2 = \varepsilon^2\,\sigma^2\,b_k$ (16a)

where b_k depends only on the k th row of $[T]$. We notice that (16a) gives as expected (15) for the FFT (Sande-Tukey algorithm). From (16), we can also compute the total error variance by summation over k and we find:

$$V_W = \varepsilon^2\,\sigma^2\sum_{k}^{N-1} b_k \tag{17}$$

5-3-2. Transmission of errors from input vector:

We have assumed so far that the input vector $\vec{V}$ of the parent matrix rotation $[T]$ was error free. We assume now that the input vector has an error $\vec{E_V}$, so that the input signal is in fact $\vec{V} + \vec{E_V}$. Due to the linearity of the transform, the exact output vector would be:

$$\vec{W} = [T]\,\vec{V} + [T]\,\vec{E_V} \tag{18}$$

We have studied in the previous sections the error introduced by the rotation $[T]\,\vec{V}$. We neglect the secondary errors introduced in the computation of $[T]\,\vec{E_V}$ and therefore consider that the error vector $\vec{E_V}$ is exactly transformed. Then, it is clear from (18) that the transmission of the input errors depends on the transform as well as the input error vector. However, we know by Parseval's theorem that the energy of a vector is transmitted exactly through a unitary tranform. Then:

$$\text{total MSE(W)} = \text{total MSE(V)} + \underbrace{V_W}_{\substack{\text{total rounding MSE in}\\ \text{the computation } \vec{W} = [T]\,\vec{V}}} \tag{19}$$

If $[T]$ is not normalized, (19) becomes:

$$\text{total MSE}(W) = N \text{ total MSE}(V) + V_W \qquad (19a)$$

5-4. Floating-point computations with rounding : error propagation

In the previous section, we have considered how the different approaches for floating-point error analysis apply to parent matrix multiplications. We now use these results and the framework developed in chapter 3 to derive, using the recursive generative rules, the rounding errors of fast unitary transforms. We consider successively the computation of norm bounds and exact computations with a statistical model.

5-4-1. Norm bounds:

Refering to the three generative rules of fast unitary transforms, we observe that permutations do not change vector norms. Thus we have to consider the combination of error bounds for parent matrix computations in the cases of generalized Kronecker product and rotations of rows.

1) For a generalized Kronecker product of two sets of matrices, $\{\mathcal{A}\}$ and $\{\mathcal{B}\}$ such as defined in chapter 3, we have (from (3) of chapter 3):

$$\vec{W} = [P^t] \left[\text{Diag}\{\mathcal{A}\}\right] [P] \left[\text{Diag}\{\mathcal{B}\}\right] \vec{V}$$

As the matrix operations are performed from right to left, we have:

$$\|E_W\| = \left\| fl\left(\left[\text{Diag}\{\mathcal{A}\}\right] fl\left(\left[\text{Diag}\{\mathcal{B}\}\right] \vec{V} \right) \right) - \left[\text{Diag}\{\mathcal{A}\}\right]\left[\text{Diag}\{\mathcal{B}\}\right] \vec{V} \right\|$$

$$= \left\| fl\left(\left[\text{Diag}\{\mathcal{A}\}\right] fl\left(\left[\text{Diag}\{\mathcal{B}\}\right] \vec{V} \right) \right) - \left[\text{Diag}\{\mathcal{A}\}\right] fl\left(\left[\text{Diag}\{\mathcal{B}\}\right]\vec{V}\right) \right.$$
$$\left. + \left[\text{Diag}\{\mathcal{A}\}\right. fl\left(\left[\text{Diag}\{\mathcal{B}\}\right] \vec{V}\right) - \left[\text{Diag}\{\mathcal{B}\}\right] \vec{V} \right\|$$

We decompose the right hand side of the last expression into:

$$E_1 = fl\left(\left[Diag\{\mathcal{A}\}\right]\ fl\left(\left[Diag\{\mathcal{B}\}\right]\ \vec{V}\right)\right) - \left[Diag\{\mathcal{A}\}\right]\ fl\left(Diag\{\mathcal{B}\}\right)\vec{V}$$

$$E_2 = \left[Diag\{\mathcal{A}\}\right]\left[\ fl\left(\left[Diag\{\mathcal{B}\}\right]\right)\vec{V} - Diag\{\mathcal{B}\}\ \vec{V}\right] \qquad (20)$$

Then, by the triangle inequality, we have:

$$\| E_W \| \leqslant \| E_1 \| + \| E_2 \|$$

We first consider $\| E_2 \|$:

since $\left[Diag\{\mathcal{A}\}\right]$ is unitary $\| E_2 \|$ reduces to:

$$E_2 = fl\left(\left[Diag\{\mathcal{B}\}\right]\right)\vec{V} - \left[Diag\{\mathcal{B}\}\right]\ \vec{V}$$

Assume that for the parent matrices $[A^k]$ of the set $\{\mathcal{A}\}$, we have found norm bounds similar to (3) or (6), and which we denote

$$\| E_w \| \leqslant E_{A^k}\ \varepsilon\ \| v \| \qquad \text{for the} \qquad (21)$$

vector rotation $\vec{w} = [A^k]\ \vec{v}$

Assume also that we have similar bounds and notations for the matrices $[B^j]$ of the set $\{\mathcal{B}\}$. Then, decomposing $\vec{V}$ in subvectors on which operate the matrices $[B^j]$, we obtain by applying (21) to each matrix rotation:

$$\| E_2 \|^2 \leqslant \varepsilon^2 \sum_j (E_{B^j})^2\ \| V \|^2 \qquad (22)$$

It follows from (22) that:

$$\left\| fl\left(\left[Diag\{\mathcal{B}\}\right]\right)\vec{V} \right\| = \left\| \left[Diag\{\mathcal{B}\}\right]\vec{V} \right\| + \mathcal{O}(\varepsilon)$$

$$= \| V \| + \mathcal{O}(\varepsilon) \qquad (23)$$

The computation of $\| E_1 \|$ follows the steps of the computation of $\| E_2 \|$ and using (23) into (20) gives:

$$\| E_1 \|^2 = \varepsilon^2 \sum_k (E_{A^k})^2\ \| V \|^2 \qquad (24)$$

Then, from (22) and (24), we have:

$$E_W \leqslant \varepsilon\ \left(\sqrt{\sum_k (E_{A^k})^2} + \sqrt{\sum_j (E_{B^j})^2}\right)\ \| V \| \qquad (25)$$

2) For a rotation of rows (rule 2) by a unitary matrix $[U]$, after applying the transform $[T]$, a similar derivation would give with E_T and E_U specifying the norm bounds obtained for $[T]$ and $[U]$:

$$\| E_W \| \leq \varepsilon \ (E_T + E_U) \ \| V \| \qquad (26)$$

The relations (25) and (26) allow the computation of error bounds for any transform defined within the framework of chapter 3. We apply now (25) to the case of fast transforms of composite order.

Assume that $N = \prod_i n_i$ and that the error bound for the parent matrices of order n_i are given by:

$E_i = K n_i^2$ as in (6). Then (25) becomes:

$$\| E_W \| \leq \varepsilon \sum_i \left(\sqrt{\frac{N}{n_i} K^2 n_i^4} \right) \| V \|$$

or $$\| E_W \| \leq \varepsilon \ \| V \| K \sqrt{N} \sum_i (n_i)^{3/2} \qquad (27)$$

a result given by Gentleman & Sande [6] for $K = 1.06\ 2\ \sqrt{2}$ and Ramos [7] for $K = 2$, both for the FFT.

5-4-2. Statistical model for input vector:

In our analysis of floating-point rounding errors, we have in the previous section used the norm bounds derived for parent matrix rotations. In this section we use the results obtained with a white statistical model for the input vector.

Whenever we can express the mean square error of the output coefficients of a parent matrix as proportional to the norm of the input vector, directly such as (15) or with the assumption of a white signal (16a) , we can obtain a recursive relation for the error by neglecting

the secondary errors (errors occuring on error terms coming from previous stages of computation). Then, if the input errors for any parent matrix are independent, we derive by (19) a total error variance of the input vector.

- for a rotation of the rows (rule 2), with independent input errors and denoting by V_T, V_U, $V_{T'}$ the total error variances of the vectors transformed by $[T]$, $[U]$, and $[T']$ (notations of chapter 3), we have:

$$V_{T'} = V_T + V_U \tag{28}$$

- for a generalized Kronecker product (rule 3), we have with the notations of the previous section:

$$V_C = \sum_{k=0} V_{A^k} \quad \sum_{j=0} V_{B^j} \tag{29}$$

We note that relations (28) and (29) are very similar to the relations (1) and (4) of chapter 3 for the expression of the number of elementary operations. This is not surprising since all error sources are considered to be of equal importance for a white signal model.

5-4-3. Application to FFT and other transforms:

We now apply the general results of 5-4-1 and 5-4-2 to the usual transforms, mainly the FFT; we restablish simply some known results and we find some new results.

a) norm relations:

We have already noted that the general result of relation (25) applies in particular to FFT of composite orders. However for the particular parent matrices of the FFT algorithm with a given radix,

we can derive tighter bounds before applying (5) and (26). First, we may take the exact value of K as given by (13) or compute directly a bound from (11) and (12). For example, we have seen that for the FFT with the Sande-Tukey algorithm, we have an exact value for the error given in (15). So, refering to the notations of relation (15) of chapter 3, we have:

$$E_{F^k} = 2 \quad \text{for } k=0 \text{ or } 2^{n-2} \quad (n \geqslant 2)$$
$$= 2\sqrt{2} \quad \text{otherwise.}$$

Applying (25), we have then:

$$\| E_W{}^n \| \leqslant \|V\| \varepsilon (\quad 2 \, (E_W{}^{n-1})^2 + \quad (2^{n-1}-2) \quad 8 + 8 \quad)$$

Thus

$$\| E_W{}^n \| \leqslant \|V\| \varepsilon \left(2\sqrt{2^{n-1}} + 2 \sum_{j=1}^{n-1} \sqrt{2^n - 2^j} \right)$$

$$\| E_W{}^n \| \leqslant \|V\| \varepsilon \sqrt{N} \, (2n + \sqrt{2} - 2)$$

This result is given as an example of application of the norm approach but the exact computation of the total mean square error is possible in this case.

Another way to obtain a tighter bound for the FFT is to consider the definition of the parent matrices as given by (8) or (9) of chapter 3 as the product of a diagonal matrix $[D]$ and a Fourier matrix. The error norm, $\|E_1\|$, for this matrix $[D]$ of order r is such that:

$$\|E_1\| \leqslant \varepsilon \, \|V\| \, E_D \quad \text{with} \quad E_D = 2\sqrt{2^r}$$

by direct application of (14). Then, if we have a tight bound for the Fourier matrix of order r, radix of the FFT, and denoted E_{F_r}, we have from (26) a bound for the parent matrix of the FFT of radix r :

$$\|E_W\| \leqslant \varepsilon \, \|V\| \, (E_{F_r} + 2\sqrt{2r})$$

For radix 2, $E_{F_2} = 2$, so : $\|E_W\| \leqslant 6 \, \varepsilon \, \|V\|$

For radix 4, we obtain E_{F_4} from (11) and (12) directly and find

$$\|E_{F_4}\| \leq 2\sqrt{23} \qquad \text{so} \qquad \|E_W\| \leq \varepsilon \|V\| (2\sqrt{23} + 4\sqrt{2})$$

These bounds correspond to the bounds proposed by Ramos [7] with a similar approach. They show the flexibility allowed by our derivation and its generality.

The experimental results obtained by Ramos [7] for the FFT show how far these bounds may be from the errors encountered in practice. The statistical approach we appy now will give more realistic estimates of the errors. However the merit of the norm bounds is to provide an error estimate which does not depend on the input coefficients as for the statistical estimate.

b) Statistical analysis:

We show now how the previous relations simplify greatly the computation of error-to-signal ratios.

1) For the FFT of radix 2, it easy to show that in relation (16a) the coefficient b_k is:

b_k = 2 (number of complex multiplications) + (number of additions)

a relation valid for both FFT algorithms. Thus, the error-to-signal ratio for the FFT of order $2^n=N$ is given from (17) and (29) with the notations of chapter 3-3 :

$$E/S = \frac{N \varepsilon^2 \sigma^2}{N^2 \sigma^2} (2 \mathcal{M}_{2^n} + \mathcal{A}_{2^n}) \qquad (30)$$

If we include in $\mathcal{M}_{2^n}$ all factors ±1 and ±j except for two stages of computation (the two first ones for the Cooley-Tukey algorithm, the two last ones for the Sande-Tukey algorithm) we have $(n-2)\ 2^{n-1}$ multiplications and $n\ 2^n$ additions, giving:

$$E/S = \varepsilon^2 \; 2\,(n-1) \qquad (31)$$

a result given by Weinstein [2] [3] for the Cooley-Tukey algorithm and Kaneko & Liu [1] for the Sande-Tukey algorithm. If we exclude all factors ±1 and ±j we find, using the column $\mathcal{M}^3_{r^n}$ of the table given in -3-1 :

$$E/S = (n - 3/2 + (\tfrac{1}{2})^{n-1})\; 2\; \varepsilon^2 \qquad (32)$$

a result given by Weinstein [2] [3] for the Cooley-Tukey algorithm and Chan & Jury [5] for the Sande-Tukey algorithm.

2) For the FFT of higher radices, the relation (30) is still valid and the column $\mathcal{M}^3_{r^n}$ gives the most accurate estimates of the error-to-signal ratio . In order to give a comparison of the FFT with radices 2, 4 , 8 and 16 we express the error-to signal ratio as a function of n such that $N = 2^n = r^{n/\log_2 r}$:

Radix	$E/S \times 1/2\,\sigma^2$
2	$n-3/2 +2(\tfrac{1}{2})^n$
4	$\frac{1.75}{2}$ n -13/12 + 4/3$(\tfrac{1}{2})^n$
8	$\frac{1.66\ n}{2}$ -57/56 + 8/7 $(\tfrac{1}{2})^n$
16	$\frac{1.656\ n}{2}$ - 241/240 +16/15 $(\tfrac{1}{2})^n$

3) For the W-H transform, we find directly:

$$E\ S = n\ \varepsilon^2$$

a result obtained by Chan & Jury [5].

4) For the Generalized Discrete transforms, we apply again (30) and the result of 3-4-1 to (17):

$$E/S = \varepsilon^2\,(n + g - 1) - 2^{g+1-n} + 2^{2-n}$$

a result obtained through long computations by Chan & Jury [5] .

5) multidimensional transforms :

We have seen that multidimensional transforms reduce to a one-dimensional transforms obtained by sucessive simple Kronecker products (see 3-7-2). The error-to-signal ratio is then easily obtained by recursive use of (29). For an L-dimensional Fourier transform of composite order $N = \prod_{i=1}^{L} n_i$ or $N = 2^m = 2^{\left(\sum_{i=1}^{L} m_i\right)}$

the error-to-signal ratio derived from (31) is:

$$E/S = 2\ \varepsilon^2 \sum_{i=1}^{L} (m_i - 1) = 2\ \varepsilon^2\ (m - L)$$

and , with more accuracy, the error-to-signal ratio derived from (32) is:

$$E/S = 2\ \varepsilon^2 \sum_{i=1}^{L} \left[m_i - 3/2 + (\tfrac{1}{2})^{m_i - 1} \right]$$

$$= 2\ \varepsilon^2 \left[m - 3L/2 + \sum_{i=1}^{L} (\tfrac{1}{2})^{m_i - 1} \right]$$

Both results are given by Chan & Jury [5].

The previous examples have been considered because they show the efficiency of our approach compared to other works.

5-5. Floating-point computations with truncation ; application to FFT:

The derivations obtained previouly in the case or rounding are still valid for truncation but they no longer give the output mean square errors, only the error variances: there is now a bias vector. Each operation introduces a new bias and since the truncation errors have the same sign, the biases always add.

Denote by the superscript ^{R} (^{T}) the errors in the output coefficients for rounding (truncation). From (10), we have directly:

$$\begin{pmatrix} \mathrm{Re}(E_W^T) \\ \mathrm{Im}(E_W^T) \end{pmatrix} = E([\mathcal{E}]) \begin{pmatrix} \mathrm{Re}(V) \\ \mathrm{Im}(V) \end{pmatrix} + \begin{pmatrix} \mathrm{Re}(E_W^R) \\ \mathrm{Im}(E_W^R) \end{pmatrix} \qquad (33)$$

where $E([\mathcal{E}])$ can be written, using (12), as a direct matrix product:

$$E([\mathcal{E}]) = \mu\,[M] \otimes [\mathcal{C}] \qquad (34)$$

M_{ij} is the number of non zero error terms in $\mathcal{E}_{ij}$. In a generalized Kronecker product, the mean error vector in the input vector is transformed and new error biases given by (33) are added. In general, we cannot pursue further our analysis without specifying the transform and the original data. However, some particular cases of interest allow to express the bias vector proportionally to the intermediate results of the computation : then, knowing how the signal is propagated, we can estimate the contribution of the bias vector to the output mean square error. We consider now two cases of simplification.

a) At each stage, the bias vector is proportional to the intermediate output vector:

This situation is approximately verified for the large family of IC transforms (see chapter 3). Two conditions are necessary, one concerning the generation of the bias vector in parent matrices, and the other condition concerning the propagation of this bias errors.

1) First we need $M_{ij} = M_i$ for all j and all parent matrices : then, from (34) $E(\mathcal{E}_{ij}) = M_i\,\mu\,\mathcal{C}_{ij}$, and from (33) we see that the bias introduced in a parent matrix rotation, denoted $\vec{B}_p$, is such that:

$$(B_p)_i = \mu\, M_i\, W_i$$

Therefore the new contributions to the bias vector are proportional

to the intermediate output vector.

2) Secondly, we require that the bias vector at the input of any parent matrix be proportional to the input vector. This condition is verified for the IC transforms in particular.

If both conditions hold, the final bias vector $\vec{B}$ is related to the output vector $\vec{W}$ (see footnote 3) :

$$B_k = \mu\left(\sum_j M_j\right) W_k \tag{35}$$

where the summation is extended to all indexes j of parent matrices appearing in the computation of W_k. Then, the contribution of the bias vector to the output mean square error is $\mu^2\left(\sum_j M_j\right)^2 \|W_k\|^2$

so that $$E\|E_{W_k}{}^T\|^2 = \mu^2\left(\sum_j M_j\right)^2 \|W_k\|^2 + E\|E_{W_k}{}^R\|^2 \tag{36}$$

Summing (36) over k gives the total mean square error:

$$V_W{}^T = \mu^2 \sum_k \left(\sum_j M_j\right)^2 \|W_k\|^2 + V_W{}^R$$

If $\sum_j M_j = C$ and does not depend on k, we have:

$$V_W{}^T = \mu^2 C^2 N^2 \sigma^2 + V_W{}^R \tag{37}$$

For the FFT, Sande-Tukey algorithm, the relation $M_{ij} = M_i$ is approximately satisfied (it is exactly verified if the factors ±1 and ±j are taken to introduce round-off errors) and the above relations yield the approximate truncation error analysis of this algorithm, even for correlated input coefficients, as done by Kaneko & Liu [1] . For independent input coefficients, this leads

---

3 We assume here that secondary errors are negligeable.

to the following error-to-signal ratio:

$$(E/S)_n = \mu^2(4n^2-7n+2) + \varepsilon^2(2n-2) \qquad (38)$$

where the multiplications by ±1 and ±j are considered to introduce truncation errors except at the two last stages of computation.

b) At each stage, the bias vector corresponds to an input bias vector proportional to the intermediate input vector:

This case of simplification is symmetric to the previous one : we propagate a fictitious bias vector towards the input vector, i.e. we determine which bias vector added to the input vector would produce the same bias vector at the output through the normal error free transform. If it happens also that these fictitious error vectors are proportional to the signal output vectors of any of the parent matrices, then we can obtain an additive input vector which will account for the bias vector. To compute the error-to-signal ratio, we need the magnitude of the output bias vector which, by Parseval's theorem, is obtained from the fictitious input bias vector as well.

In particular, this approach gives an approximate error analysis for the Cooley-Tukey algorithm of the FFT and it easy to show that (38) applies also to the Cooley-Tukey algorithm. On Figure 5-1, we have plotted the experimental points obtained by Weinstein [3] and the theoretical curve given by (38). For the theoretical curve, we have taken the experimental values of μ and ε^2 presented in 5-2-1. We find a good agreement between the curves.

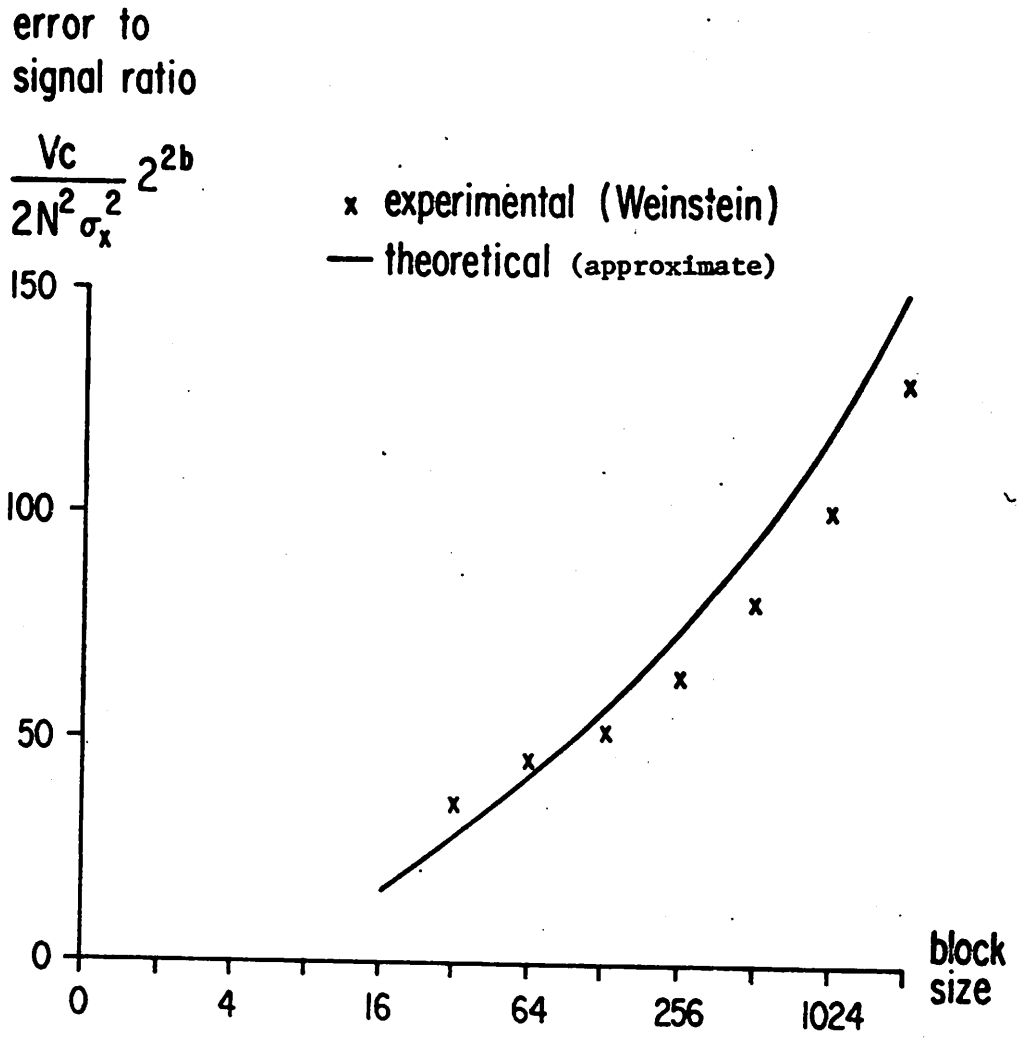


Fig. 5-1. FFT error analysis for floting-point and truncation

5-6. Floating-point computations with non randomized rounding of midway point:

The results derived in 5-4 have been compared with experimental error measures [1] [2] [3] [5] and for transforms of low orders a good agreement was found. Weinstein [2] [3] has shown that they descibe in fact with good accuracy the case of a perfect rounding situation when the midway point is randomly rounded up or down. However common rounding schemes are not so sophisticated and systematically round up or down this midway point, thus introducing a bias similar , for computation purposes, to the truncation bias. Therefore, our analysis of the truncation errors can also explain the strange experimental results obtained with common rounding schemes.

Weinstein has shown that additions are responsible for an additional error when common rounding is used and that this error becomes rapidly predominant as the block size increases. Let us assume that each addition introduces an error with mean μ' and variance ε^2. We assume, as previously, that the multiplications introduce an error with zero mean (no bias) and variance ε^2. Then, from (36) we obtain in this case and for the FFT a new error-to-signal ratio:

$$E/S = \mu'^2 \, (2.25 \, n^2 - 2.75 \, n + 0.5) + (E/S)^R \qquad (39)$$

where $(E/S)^R$ is given by (31) or (32).

We have plotted on Figure 5-2 the experimental results reported by Weinstein [3]. For each experimental point we compute, from (39) the corresponding value of μ' ; for all points, except maybe n=11, we obtain roughtly $\mu'^2 = 0.028$. This value corresponds to a probability of 0.33 of occurence of the midway point in an addition. This result

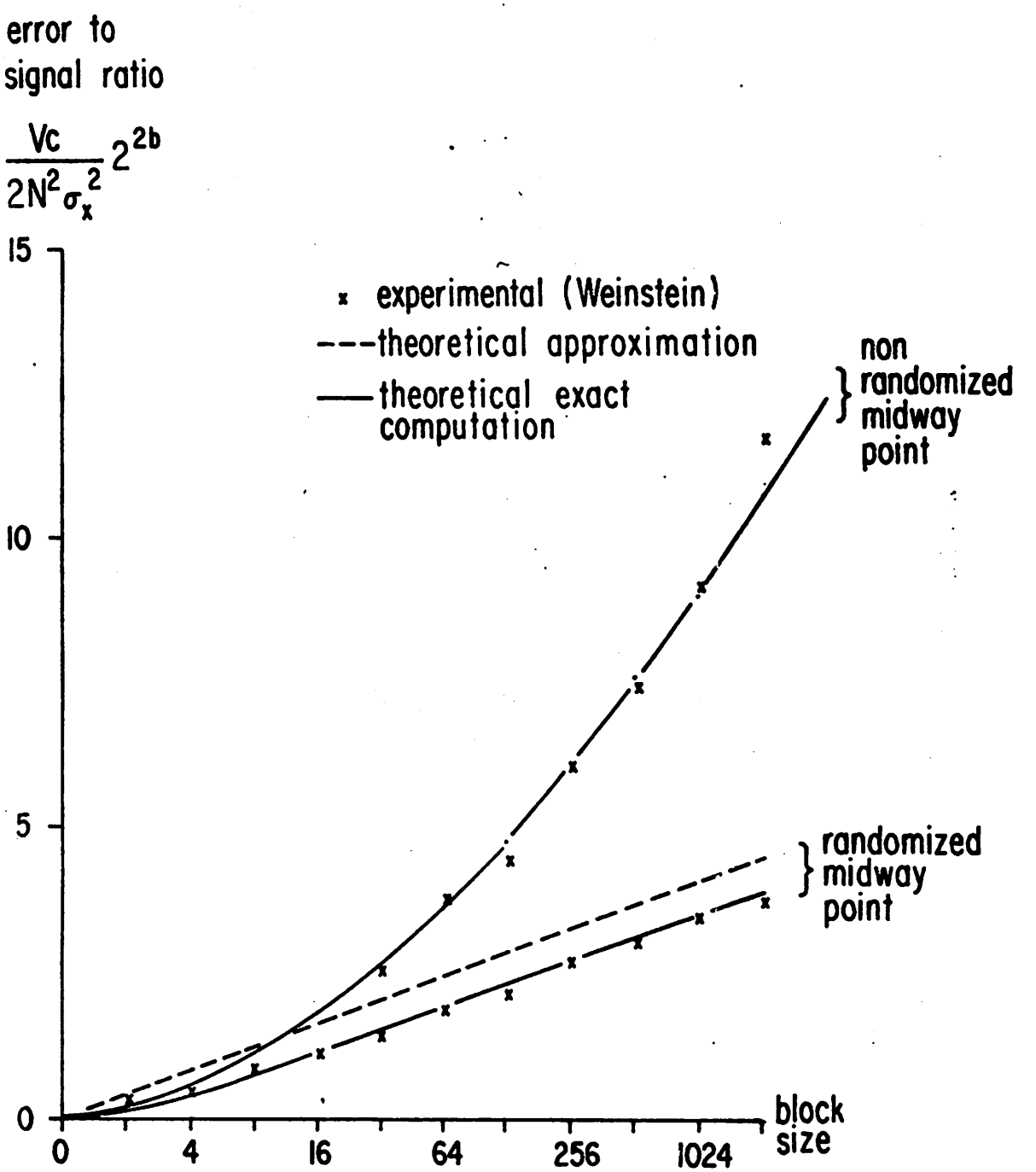


Fig. 5.2 FFT error analysis for floating-point with rounding-effect of non-randomized midway point.

has an experimental support in a work by Liu and Kaneko [11] . In Figure 5-2, we have plotted the curve given by (39) for $\mu'^2 = 0.028$. Again we see a good agreement with the experimental results.

A natural conclusion to this section is that the rounding error model is quite imperfect to describe the common rounding situations. The general problem of rounding errors in floating-point computations seems quite complex [9] [10] but a better understanding of the floating-point representation errors is necessary to obtain an accurate analysis of rounding errors in fast transform algorithms. We have only considered the case of the FFT in this section but our approach is obviously valid for other transforms, provided that the error model still applies.

5-7. Errors in the representation of transform entries or factors:

So far, we have considered only the round-off errors introduced in the computation. The parent matrix coefficients and the factors of the algorithm are also represented with finite length registers and therefore with approximations. In the experimental results obtained by Weinstein and used in the previous sections to support our analyses, these errors do not appear since both the "exact" computation and the rounded or truncated computations used the same entries or factors.

These entries or factors are usually stored in a table and it is reasonable to store them with a rounding scheme since this operation is done once. However, the same coefficient is usualy used several times in a transform computation and also the coefficients are correlated. We believe that with our recursive approach, it is

possible to express the output errors introduced by each coefficient. But it is certainly much simpler to assume that these representation errors are independent. Then, we may use the results of chapter 4 for fixed-point computations and chapter 5 for floating-point computations. Relation (3) of chapter 4 and relation (7) of chapter 5 would then have additional terms which would modify the following derivations adding new error terms to our results. However, for both cases of number representation, the contribution of the enties or factors approximations to the output error-to-signal ratio is a term proportional to the number of stages of computation n .In most cases, this term is rapidly negligeable. Experimental evidence for the FFT by Weinstein [3] and Kaneko & Liu [1] confirm that the new error terms are of second order [4]. Therefore, we shall not pursue further the analysis of the errors due to the representation of entries or factors.

5-8. Errors in transform domain approximations:

The transform encoding methods introduce coding errors in the transform coefficients of a vector (quantization, selection of transmitted coefficients). After inverse transform of the approximated transform vector, we obtain an approximation of the original vector. In chapter 7, we shall present with more detail these encoding

[4] Ramos [7] finds however with a different model (an absolute error model similar to the model we have used for fixed-point computations) an additional error term of same order of magnitude . We think that his model overemphasises the errors coming from the representation of entries and factors.

methods. In this section, we give some qualitative comments on the introduced errors.

By Parseval's theorem, we know that the mean square error can be computed on the original vector or on the transformed coefficients. However, a question of fundamental interest is to know the distribution of these errors for a given encoding scheme. It is known that a low-pass filter for the FFT will produce a Gibbs phenomenon on each signal discontinuity(either inside the input sequence or at its extremities). For the W-H transform of pictorial data, it has been reported that the errors accumulate on the edges of small blocks [12]. Still with the W-H transform, suppression of the discontinuities between extrimeties of the sequence yields a clear improvement [13]. The Haar transform, finally, shows a net improvement over the W-H transform when threshold encoding is used [14] . But both W-H and Haar transform let false contours appear in the shape of grids.

A general theoretical treatment of these effects does not seem easy; we should like to state some qualitative results which conclude our error analysis of fast unitary transforms and motivate the search for other transforms such as the generalized slant transforms of next chapter. With smoothly varying basis vectors , a transform will badly represent dicontinuities which will be rejected as transform coefficients of low magnitudes (high frequencies for the FFT). The errors then on the reconstructed data concentrate on dicontinuities, i. e. enhance the <u>true</u> contours. With discontinuous basis vectors, some discontinuities of the data will be reproduced or even enhanced while

some false discontinuities will appear. With locally defined basis vectors, the false contours will remain close to the real ones and so will be less objectionable.

These remarks lead to the need of a compromise transform between smooth and discontinuous basis vectors (the smoother versus the faster) and also with as much local properties as possible. Some generalized transforms verify this compromise.

5-9. Conclusions:

In this chapter, we have presented an error analysis of fast unitary transforms in the case of a floating-point representation of numbers. Using a systematic approach, made possible by the recursive definition of fast unitary transforms, we have considered all the cases of practical interest for rounding as well as for truncation arithmetic. We have, at the same time, presented a synthetic survey of previous works carried out for the FFT; we have emphasized their common assumptions and their different approaches. We have derived more accurate expressions, specially for the norm bounds, and also some new theoretical results, specially when the round-off errors have a bias. However, we should stress that the results obtained are either bounds which may be loose for specific input vectors or approximations valid under statistical assumptions. We are confident that they give nethertheless reasonable guides in the comparison and choices of transform hardware and software implementations.

C H A P T E R V I

FAST UNITARY TRANSFORMS WITH PRESCRIBED BASIS VECTORS

GENERALIZED SLANT TRANSFORMS

6-1. Introduction:

In chapter 3, we have developed a common framework for unitary transforms with a fast algorithm in terms of recursive generative rules. We have derived several families of such transforms depending on the choice of parent matrices in a given class. Now, we wish to design unitary transforms with desirable properties and still a fast algorithm. The problem considered in this chapter is to include exactly a preassigned set of basis vectors, called replacement vectors.

We first dicuss the problem in its most general form. Then we concentrate on a specific problem of interest: the inclusion of so called slant vectors into the WHH family of transforms between the Haar and Walsh-Hadamard transforms.

6-2. General case:

We have a set of p orthonormal replacement vectors denoted $\vec{X}_1$, $\vec{X}_2$,....., $\vec{X}_p$ of dimension N ($p \leq N$) and we want to design a fast unitary transform $[U]$ which will have the vectors $\vec{X}_1$, $\vec{X}_2$,.. ..., $\vec{X}_p$ among its row vectors.

The first approach to this problem is to examine the set of basis vectors obtained within a family of unitary transforms with a fast algorithm and to try to select a transform which includes the replacement vectors. Unfortunately, the set of basis vectors of

such a family is not large. For example, the basis vectors of order 8 of the IC_2 family belong to the set (coefficients not necessarly ordered)

$$(1 , w_1 , w_2 , w_3 , w_1 w_2 , w_1 w_3 , w_2 w_3 , w_1 w_2 w_3)$$

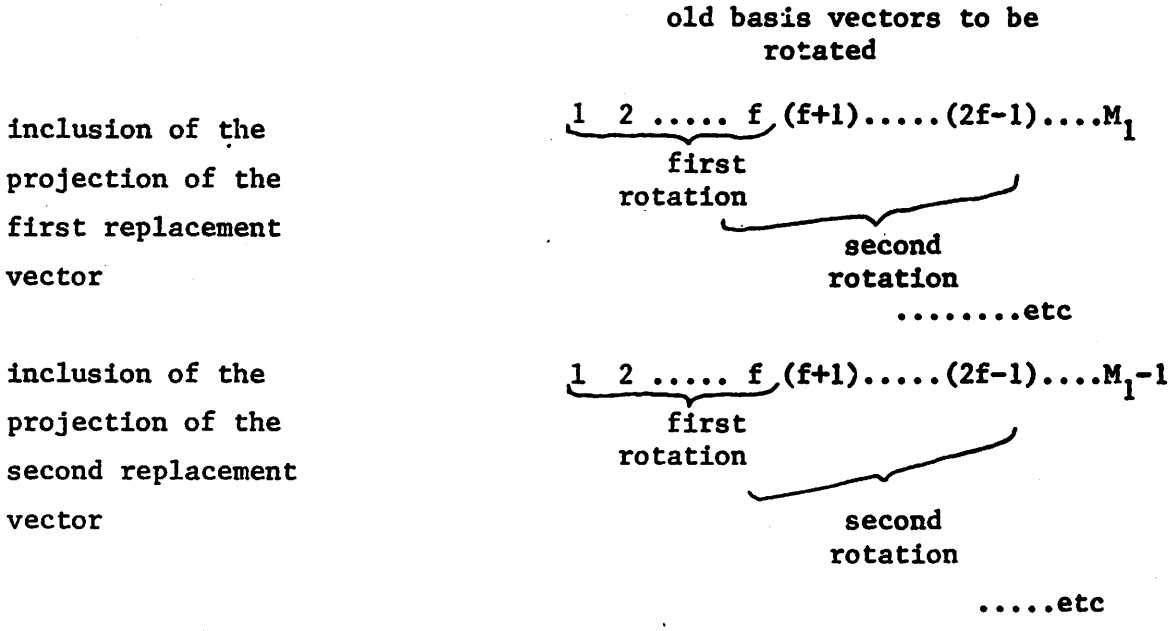
with w_i a root of unity. Therefore this approch is not likely to succeed in general.

The next approach is to modify a unitary transform $[T]$ with a fast algorithm in order to include by some rotations the replacement vectors. Of course, this approach will succeed in a fast algorithm only if the set of replacement vectors belongs to a subspace S_M of low dimensionality, say M ($p \leq M \leq N$), in the basis of $[T]$. The rotation of S_M will add a maximum of M^2 multiplications and M (M-1) additions to the elementary operations already required by $[T]$. It may happen that the rotation of S_M can be further decomposed into rotations of lower dimensionalities if there are subspaces of S_M of dimensions M_1, M_2,... ..., M_m ($M = M_1 + M_2 + \ldots + M_m$), in which the projections of $\vec{X}_1$, $\vec{X}_2$, ...,$\vec{X}_p$ are also orthogonal. Then, the inclusion of the vectors $\vec{X}_1$, ...,$\vec{X}_p$ into the basis will require at most

$$\sum_{k=1}^{m} M_k^2 < M^2 \quad \text{multiplications}$$

and $$\sum_{k=1}^{m} M_k (M_k - 1) < M (M - 1) \quad \text{additions}$$

However we may wish to perform these rotations of various dimensions as a sequence of rotations of fixed dimension, say f. Then, the successive rotations which include the different basis vectors can be performed as shown on the following diagram, where M_1 is the dimension of a subspace of S_M as previously considered:



and similarly up to the last projection of a remplacement vector in this subspace. The total number of rotations of order f is then at most, when S_M is not further decomposed :

$$R = \sum_{k=1}^{p} \left[\frac{M - k + 1}{f-1} \right]$$

where $[x]$ denotes the smallest integer larger than x .

Although this number of rotations is fixed, they depend on the ordering chosen for the original basis vectors, and on the choice of unconstrained projection vectors making an orthogonal base with the projections of the replacement vectors. The new basis vectors depend on these choices.

Two particular cases may reduce the computational complexity of these rotations:

a) two (or more) replacement vectors have merged or orthogonal projections in a subspace rotated by a matrix of order f : their projections may be included simultaneously.

b) two (or more) replacement vectors have orthgonal projections

of the same magnitude: some entries of the rotation matrix are then ±1 so that the number of multiplications is reduced.
In the following we shall make use of these simplifications.

We are not able to discuss further the inclusion of replacement vectors without considering their properties . In the following, we examine the inclusion of a set of Slant vectors.

6-3. Slant vectors:

The slant vectors we define in this section have a piecewise linear variation of their components. Various combinations of these vectors form sets of orthonormal vectors which can be included in the WHH bases with few operations: we obtain a class of "slant transforms" which offer a possible compromise for the trade-off discussed in 5-8. Their interest will be shown in chapter 7 where we compare the performance of several fast unitary transforms for processing a first order Markov process.

6-3-1. Basic slant vectors:

There are three basic slant vectors of length 2^k : the "linear slant vector", denoted $\vec{L}^k$, the "cup slant vector", denoted $\vec{V}^k$, and the "jump slant vector" , denoted $\vec{J}^k$. They are defined as follows:

a) Linear slant vector:

Its components are linearly decreasing and given by:

$$L_i^k = \frac{(2^k - 1) - 2i}{\sqrt{\frac{2^k (2^{2k}-1)}{3}}} \qquad i = 0,\ldots\ldots,2^k-1 \qquad \text{(1a)}$$

The vector $\vec{L}^k$ is normalized and orthonormal to the constant vector, denoted $\vec{C}^k$.

b) Cup slant vector:

Its components are linearly decreasing for $0 \leqslant i < 2^{k-1}$ and symmetrically increasing for $2^{k-1} \leqslant i < 2^{k}$, thus showing a "V" shape ; they are given by:

$$V_i^k = \frac{(2^{k-1}-1) - 2i}{\sqrt{\frac{2^k (2^{2k}-1)}{3}}} \qquad i = 0,\ldots\ldots,2^{k-1}-1 \qquad (1b)$$

$$V_i^k = V_{2^k-1-i}^k \qquad i = 2^{k-1},\ldots\ldots,2^k-1$$

The cup slant vector is normalized and orthonormal to both $\vec{C}^k$ and $\vec{L}^k$.

c) Jump slant vector:

Its components are linearly decreasing with a positive discontinuity in the middle of the sequence:

$$J_i^k = \frac{\frac{(2^k-1)(2^{k-1}-1)}{3} - 2^k i}{\sqrt{\frac{2^k (2^{2k}-1)(2^{2k-1}-1)}{9}}} \qquad i = 0,\ldots\ldots,2^{k-1}-1 \qquad (1c)$$

$$J_i^k = -J_{2^k-1-i}^k \qquad i = 2^{k-1},\ldots\ldots,2^k-1$$

The jump slant vector is normalized and orthonormal to $\vec{C}^k$, $\vec{L}^k$ and $\vec{V}^k$

We note that the basic slant vectors $\vec{C}^k$, $\vec{L}^k$, $\vec{V}^k$ and $\vec{J}^k$ have respective zequencies 0,1,2,and 3 just as the first W-H vectors. However, the average variation between successive components is reduced so that these slant vectors form a smoother basis. The basic slant vectors for k=2 and 3 are shown on Figure 6-1 a and b.

6-3-2. Sets of slant vectors:

We denote by $\vec{L}_\lambda^\ell$, $\vec{V}_\lambda^\ell$, and $\vec{J}_\lambda^\ell$ the slant vectors of length 2^k but

Its components are linearly decreasing for $0 \leq i < 2^{k-1}$ and symmetrically increasing for $2^{k-1} \leq i < 2^k$, thus showing a "V" shape ; they are given by:

$$V_i^k = \frac{(2^{k-1}-1) - 2i}{\sqrt{\frac{2^k (2^{2k}-1)}{3}}} \qquad i = 0,\ldots\ldots,2^{k-1}-1 \qquad (1b)$$

$$V_i^k = V_{2^k-1-i}^k \qquad i = 2^{k-1},\ldots\ldots,2^k-1$$

The cup slant vector is normalized and orthonormal to both $\vec{C}^k$ and $\vec{L}^k$.

c) Jump slant vector:

Its components are linearly decreasing with a positive discontinuity in the middle of the sequence:

$$J_i^k = \frac{\frac{(2^k-1)(2^{k-1}-1)}{3} - 2^k i}{\sqrt{\frac{2^k (2^{2k}-1)(2^{2k-1}-1)}{9}}} \qquad i = 0,\ldots\ldots,2^{k-1}-1 \qquad (1c)$$

$$J_i^k = -J_{2^k-1-i}^k \qquad i = 2^{k-1},\ldots\ldots,2^k-1$$

The jump slant vector is normalized and orthonormal to $\vec{C}^k$, $\vec{L}^k$ and $\vec{V}^k$

We note that the basic slant vectors $\vec{C}^k$, $\vec{L}^k$, $\vec{V}^k$ and $\vec{J}^k$ have respective zequencies 0,1,2,and 3 just as the first W-H vectors. However, the average variation between successive components is reduced so that these slant vectors form a smoother basis. The basic slant vectors for k=2 and 3 are shown on Figure 6-1 a and b.

6-3-2. Sets of slant vectors:

We denote by $\vec{L}_\lambda^\ell$, $\vec{V}_\lambda^\ell$, and $\vec{J}_\lambda^\ell$ the slant vectors of length 2^k but

with only 2^{ℓ} non-null components with index i , $\lambda 2^{\ell} \leqslant i < (\lambda+1)\, 2^{\ell}$ ($\lambda = 0,\ldots\ldots,2^{k-\ell}-1$). For any ℓ_1 such that $\ell_1 \leqslant k$, any $\lambda < 2^{k-\ell}$ and any $\lambda' < 2^{k-\ell_1}$, $\vec{V}_{\lambda}^{\ell}$ and $\vec{J}_{\lambda}^{\ell}$ are orthogonal to $\vec{C}^{k}$, $\vec{L}_{\lambda'}^{\ell_1}$, $\vec{V}_{\lambda'}^{\ell_1}$ and $\vec{J}_{\lambda'}^{\ell_1}$. The slant vectors $\vec{L}_{\lambda}^{2}$, $\vec{V}_{\lambda}^{2}$, and J_{λ}^{2} for $\lambda = 0,1$ and $\vec{L}_{0}^{3}$, $\vec{V}_{0}^{3}$ and $\vec{J}_{0}^{3}$ all for k=4 are shown on Figure 6-1c .

Any set of orthogonal vectors is acceptable as a set of replacement vectors. In the following, we shall consider particularly the "complete"

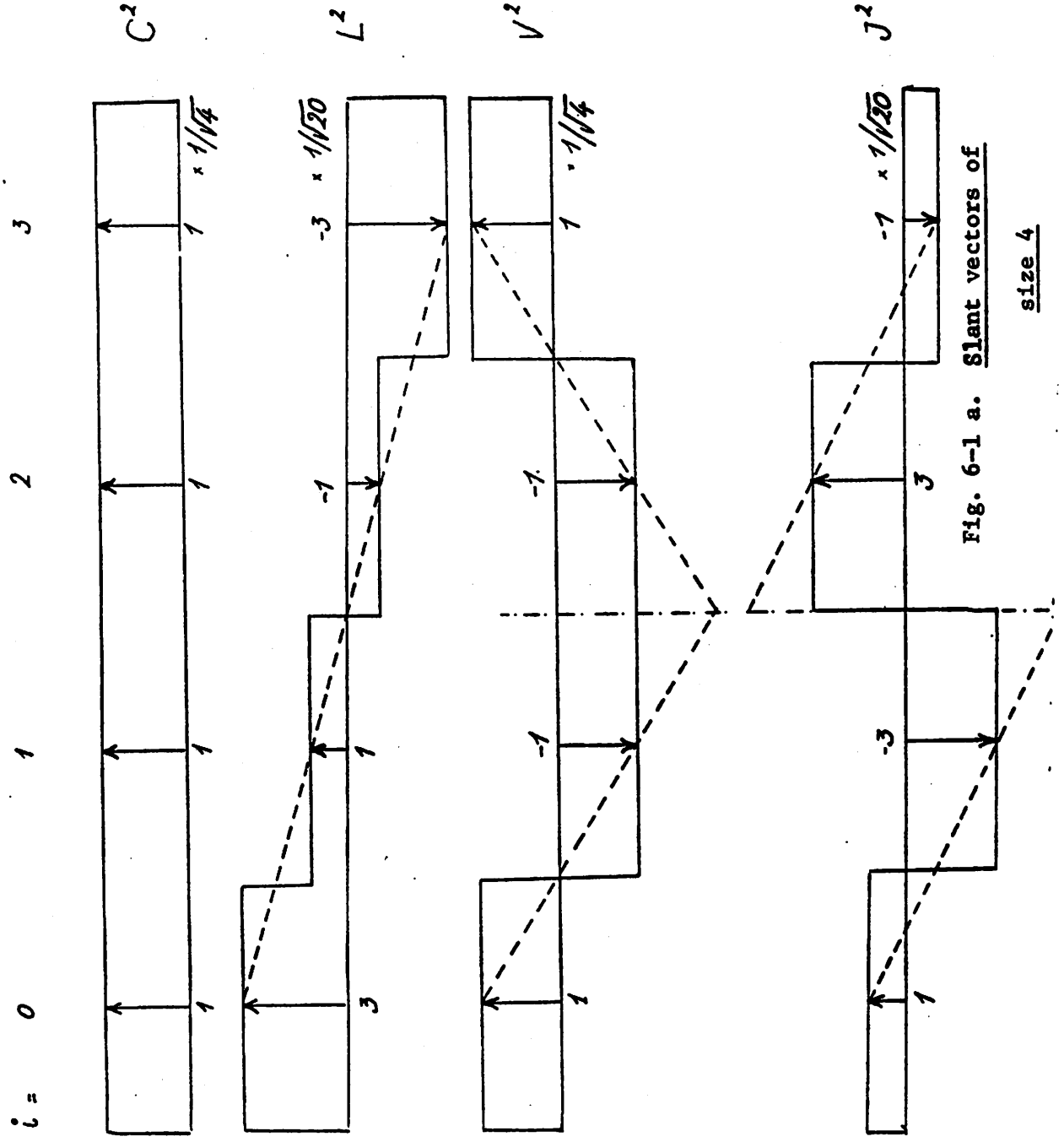


Fig. 6-1 a. Slant vectors of size 4

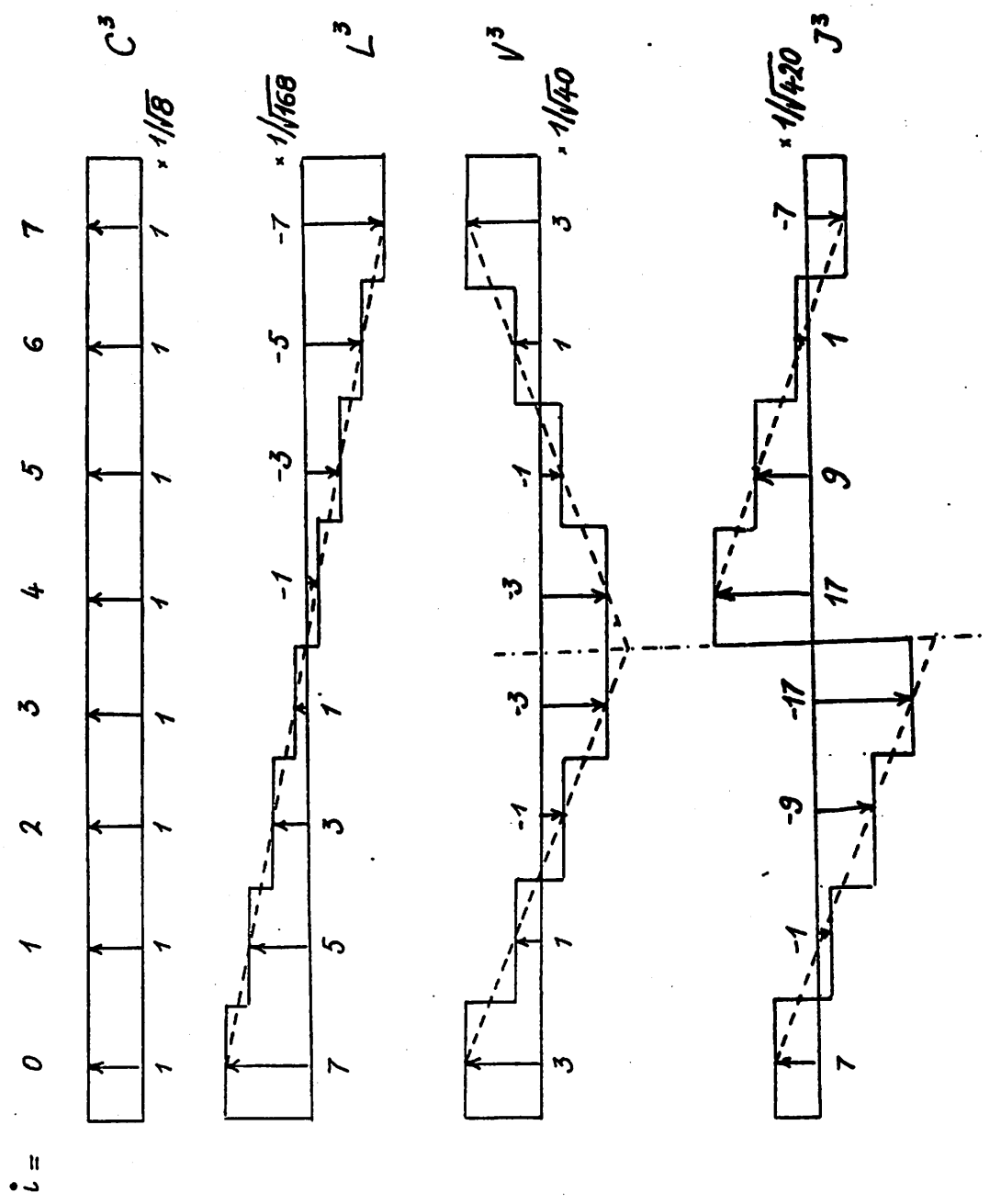


Fig. 6-1 b. Slant vectors of size 8

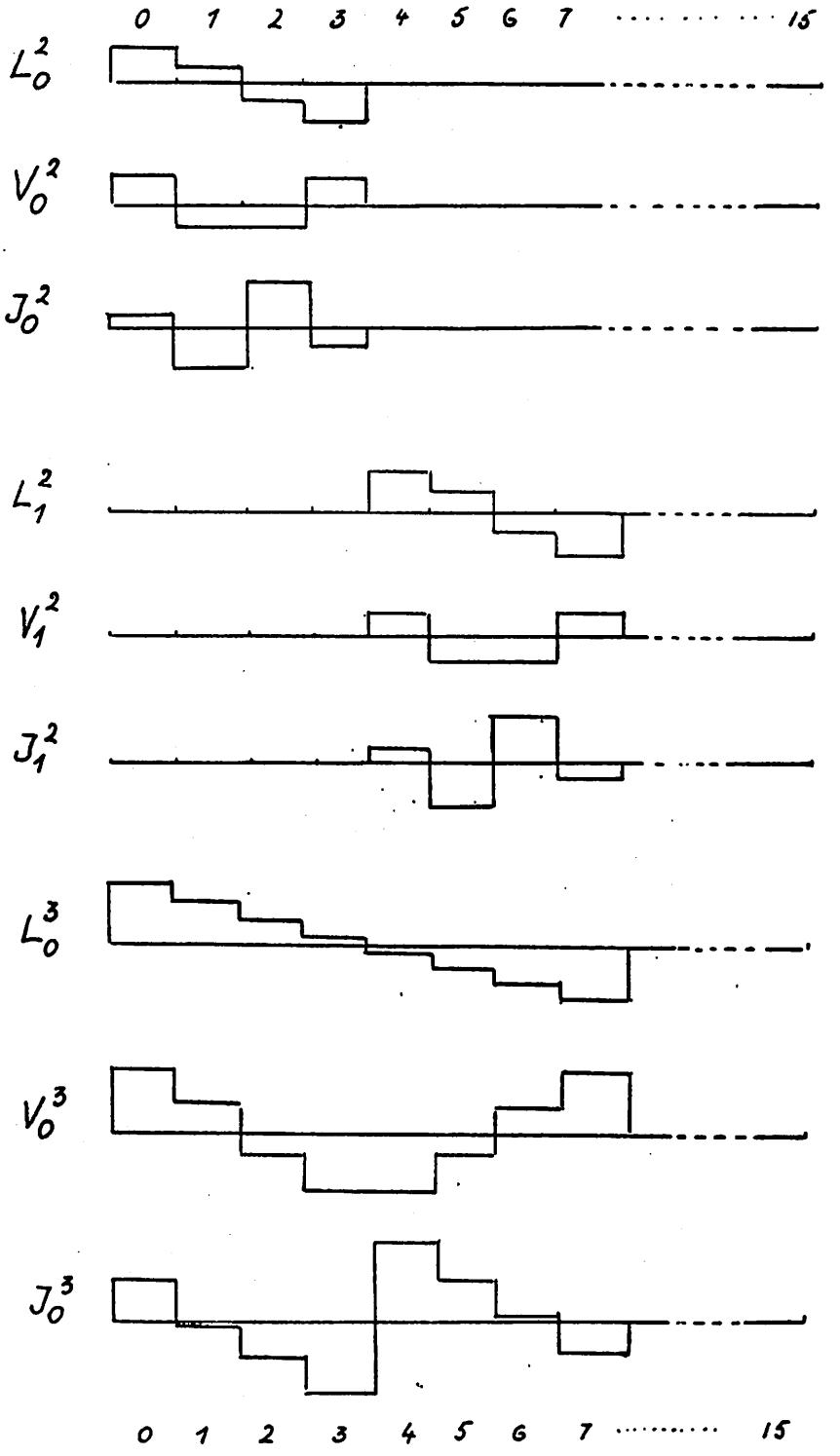


Fig. 6-1 c. Examples of slant vectors

set $\mathcal{S}(\ell_1, \ell_2, n)$ which includes all orthonormal slant vectors of length 2^n and with 2^ℓ non-zero components when

$$2 \leq \ell_1 \leq \ell \leq \ell_2 \leq n$$

This set includes specifically:

$$\begin{cases} \vec{C}^n \\ \vec{L}^{\ell_2}_\lambda \end{cases} \quad \text{for} \quad \lambda = 0, \ldots\ldots, 2^{n-\ell_2} - 1$$

$$\vec{V}^\ell_\lambda \ , \ \vec{J}^\ell_\lambda \quad \text{for} \quad \ell_1 \leq \ell \leq \ell_2 \quad \text{and} \quad \lambda < 2^{n-\ell}$$

For $\ell_1 = 2$ and $\ell_2 = n$ we have the set of all 2^n orthogonal slant vectors of dimension 2^n : they are the basis vectors of the Slant Haar transform (see 6-4).

Ordering of a set of slant vectors:

A set of slant vectors can be ordered according to the following successive rules:

1) decreasing number of non-zero components , so that the ordering is from globally to locally dependent vectors as in the ordering of the Haar basis vectors by their ranks.

2) increasing zequencies : we have seen that slant vectors which are non-zero on the same interval have different zequencies.

3) from left to right: the slant vectors with same number of non-zero components and with same zequency, are defined on disjoint intervals and so can be ordered.

The corresponding ordering is uniquely defined and consistent with the rank ordering of the Haar transform and the zequency ordering of the W-H transform.

6-4. A generalization of Haar transform to include a set of slant vectors:

Our goal is to define one or more discrete fast unitary transform which includes a given set of orthonormal slant vectors. Following the general ideas of section 6-2, we first need a fast transform which can be modified to include the slant vectors. We choose first the Haar transform and we shall generalize in 6-6 to the WHH transforms.

Thus, we have to determine the subspaces of Haar vectors which include the slant vectors of the replacement set: for this, we give first the general expression of a slant vector in the Haar basis.

6-4-1. Expression of a slant vector in the Haar basis:

Our discussion will consider the Haar transform in natural order for sake of simplicity and further generalizations. It is not difficult to verify that the transformed vectors of $\vec{L}_\lambda^k$, $\vec{V}_\lambda^k$ and $\vec{J}_\lambda^k$ by the Haar matrix $\left[H_{2^n}\right]$ are given respectively by:

$$(\overrightarrow{\mathrm{Tr}\ L}_\lambda^k)_i = \frac{C}{\sqrt{2^n\ 2^k\ \frac{(2^{2k}-1)}{3}}} \begin{cases} C = \sqrt{2^{n-1-\ell}}\ 2^{2\ell+1} & \text{if } 0 \leqslant i-\lambda 2^k < 2^k \\ & \text{and } i-\lambda 2^k \equiv 2^\ell \bmod(2^{\ell+1}) \\ & 0 \leqslant \ell \leqslant k-1 \\ C = 0 & \text{otherwise} \end{cases} \tag{2a}$$

$$(\overrightarrow{\mathrm{Tr}\ V}_\lambda^k)_i = \frac{C}{\sqrt{2^n\ 2^k\ \frac{(2^{2k-\ell}-1)}{3}}} \begin{cases} C = \sqrt{2^{n-1-\ell}}\ 2^{2\ell+1} & \text{if } 0 \leqslant i-\lambda 2^k < 2^{k-1} \\ & \text{and } i-\lambda 2^k \equiv 2 \bmod(2^{\ell+1}) \\ & 0 \leqslant \ell \leqslant k-2 \\ C = -\sqrt{2^{n-1-\ell}}\ 2^{2\ell+1} & \text{if } 0 \leqslant i-2^k-2^{k-1} < 2^{k-1} \\ & \text{and } i-\lambda 2^k-2^{k-1} \equiv 2^\ell \bmod(2^{\ell+1}) \\ & 0 \leqslant \ell \leqslant k-2 \\ C = 0 & \text{otherwise} \end{cases} \tag{2b}$$

$$(\overrightarrow{Tr\ J}^{k}_{\lambda})_i = \frac{C}{\sqrt{\frac{2^{n+k}(2^{2k}-1)(2^{2k-1}-1)}{9}}}$$

$$\begin{cases} C = \sqrt{2^{n-1-\ell}}\, 2^{2\ell+k} & \text{if } 0 \leq i-\lambda 2^k < 2^k \\ & \text{and } i-\lambda 2^k \equiv 2^\ell \bmod(2^{\ell+1}) \\ & 0 \leq \ell \leq k-2 \\ C = \sqrt{2^{n-k}}\, \frac{(2^{2k-2}-1)}{3}\, 2^k & \text{if } i = \lambda 2^k + 2^{k-1} \\ C = 0 & \text{otherwise} \end{cases} \qquad (2c)$$

The parameter ℓ indicates the zones as defined in chapter 2.

6-4-2. Properties of the slant vectors in the Haar basis and algorithm to introduce them:

We first observe or show some properties of the projections of the slant vectors using the relations (2a), (2b) and (2c).

a) equal components inside the zones:

As we have noted in 6-2, the presence of equal components leads to rotations without multiplications. In the expressions of the transformed vectors, we note that components in a zone of the definition interval of the slant vector are equal for each vector but vary for different vectors. The practical consequence is that we first rotate each zone with any unitary transform of same order and having the constant vector in its basis. The simplest of such transforms is the Haar transform (but a W-H, Fourier.... or Slant transform would also do with more operations).

After performing the Haar rotations of each zone, the slant vectors would be expressed in the new basis by the following components:

$$(\overrightarrow{TrL}^{k}_{\lambda})'_i = \frac{1}{\sqrt{2^{2k}\,\frac{(2^{2k}-1)}{3}}} \quad \text{for } i = \lambda 2^k + 2^\ell, \ \ell = 0,\ldots\ldots,k-1$$

$$= 0 \quad \text{for } i \neq \lambda 2^k + 2^\ell \qquad (3a)$$

$$(\overrightarrow{Tr\ V}\ {}^{k}_{\lambda})'_{i} = \frac{2^{k+\ell}}{\sqrt{\frac{2^{2k}\,(2^{2k-2}-1)}{3}}} \qquad \text{for} \quad i = \lambda 2^{k} + 2^{k-1} + 2$$
$$\ell = 0,\ldots\ldots,k-2$$
$$= 0 \qquad \text{for} \quad i \neq \lambda 2^{k} + 2^{k-1} + 2^{\ell} \qquad (3b)$$

$$(\overrightarrow{Tr\ J}\ {}^{k}_{\lambda})'_{i} = \frac{C}{\sqrt{\frac{2^{2k}(2^{2k}-1)(2^{2k-1}-1)}{9}}}$$

$$\begin{cases} C = 2^{2k+\ell-1} & \text{for } i = \lambda 2^{k} + 2 \\ & \ell = 0,\ldots,k-2 \\ C = \frac{2^{k}(2^{2k-2}-1)}{3} & \text{for } i = \lambda 2^{k} + 2^{k-1} \\ C = 0 & \text{for } i \neq \lambda 2^{k} + 2^{\ell} \end{cases}$$

Comparing the new components of each slant vector, we note that:

1) $\vec{V}^{k}_{\lambda}$ belongs to a subspace orthogonal to the subspace containing $\vec{L}^{k}_{\lambda}$ and $\vec{J}^{k}_{\lambda}$.

2) $\vec{L}^{k}_{\lambda}$ and $\vec{J}^{k}_{\lambda}$ have proportional components except one ; we can then include both vectors by the same rotations (the last one will account for the different components).

b) <u>slant vectors in nested intervals</u>:

We wish to prove that it is more efficient to include slant vectors defined in nested interval in order of their respective sizes (number of non-zero components in the original definition. Let us assume that the slant vector $\vec{S}^{k}_{\lambda}$ is non-zero only in the interval I of size 2^{k} and that the slant vector $\vec{S}_{1}$ is non-zero in I and at least one consecutive interval. The components of $\vec{S}_{1}$ in I are linearly varying and so $\vec{S}^{I}_{1}$, the restriction of $\vec{S}_{1}$ in I, is linearly related to $\vec{L}^{k}_{\lambda}$ (same indices as $\vec{S}^{k}_{\lambda}$). Then the rotations of the zones of I by Haar transforms, which were introduced in the previous paragraph to include $\vec{S}^{k}_{\lambda}$, will be useful to include $\vec{S}_{1}$.

Morever, if $\vec{S}_{\lambda}^{k} = \vec{J}_{\lambda}^{k}$, we may choose rotations which include simultaneously $\vec{J}_{\lambda}^{k}$ and $\vec{L}_{\lambda}^{k}$: all these rotations will be useful to include $\vec{S}_{1}$.

c) <u>algorithm:</u>

Making use of these properties, we state now the algorithm to design a fast transform which includes a set of orthonormal slant vectors.

Initialization: $[T]$, Haar transform of appropriate order is first applied to the input sequence.

Step 1 : Express a slant vector with smallest size in the basis of $[T]$.

Step 2 : Rotate the zones with equal coefficients with Haar transforms of appropriate orders. The transform is now $[T']$.

Step 3 : Rotate the rows with non-zero coefficients in the expression of the slant vector in $[T']$. For the last vectors to be included, if they are of the type L and J defined on the same interval, they have to be included simultaneously . Let us call the obtained transform $[T]$.

Step 4 : Start step 1 again if the whole set has not yet been included.

This method generates a fast transform which includes the prescribed set of slant vectors in its basis and requires, when starting from the Haar transform, the minimum number of operations.

As the coefficients of an Haar transform are obtained at various stages of computation, some rotations for the inclusion of slant vectors can be performed earlier in the algorithm : in general, all

operations to include a slant vector $\vec{S}^k$ can be performed at the k th stage of computation. Consequently, the corresponding slant transform of order 2^n can be expressed by a generalized Kronecker product.

6-5. Examples of slant transforms ; computational complexity :

6-5-1. Inclusion of basic slant vectors:

a) Linear slant vector:

Let us first include the vector $\vec{L}^k$ into the Haar transform of order 2^n . From the previous discussion, we first rotate each zone by corresponding Haar transforms and thus reduce the dimension of the subspace representing $\vec{L}^k$ to k (for $\ell = 0,\ldots,k-1$). We then rotate the corresponding rows by (k-1) matrices of order 2. We have $\binom{k}{2} \times (k-2)! = k!/2$ possible choices, $(k-1)!/2$ of them including also the vector $\vec{J}^k$. For n=k=3, we have 3 possible matrices, one including also the vector $\vec{J}^3$. We present them in Fig. 6-2 a and their corresponding algorithms in Fig. 6-2 b.

The fast algorithm for the corresponding slant transform requires:

additions : $2(2^n-1) + 2(2^{k-1}-1) + \ldots\ldots + 2(2-1) + 2(k-1)$

$= 2(2^n-1) + 2^2 (2^{k-1}-1)$

multiplications :(k-1)-1 (one multiplication per matrix is postponed to the next rotation matrix, two happen to be a multiplication by a power of 2 and therefore are performed with a shift.

shifts : 2(k-1)

normalizations : $2^n - 2^{n-2} - (k-1)$ (we normalize to the 2^{n-2} vectors with 4 non zero components and the k rotated components are assumed to be normalized by the multiplications.

$$[A] = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 7 & 5 & 3 & 1 & -1 & -3 & -5 & -7 \times 1/\sqrt{21} \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 7 & -1 & -9 & -17 & 17 & 9 & 1 & -7 \times 1/\sqrt{105} \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -3 & 3 & -1 & 1 & -3 & 3 & -1 \times 1/\sqrt{5} \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \times \sqrt{2} \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \times \sqrt{2} \end{bmatrix}$$

$$[B] = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 7 & 5 & 3 & 1 & -1 & -3 & -5 & -7 \times 1/\sqrt{21} \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 7 & 11 & -27 & -23 & 23 & 27 & -11 & -7 \times 1/\sqrt{357} \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 3 & -5 & 3 & -5 & 5 & -3 & 5 & -3 \times 1/\sqrt{17} \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \times \sqrt{2} \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \times \sqrt{2} \end{bmatrix}$$

$$[C] = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 7 & 5 & 3 & 1 & -1 & -3 & -5 & -7 \times 1/\sqrt{21} \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -3 & -3 & 3 & 3 & -1 & -1 \times 1/\sqrt{5} \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 7 & -13 & 9 & -11 & 11 & -9 & 13 & -7 \times 1/\sqrt{105} \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \times \sqrt{2} \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \times \sqrt{2} \end{bmatrix}$$

Fig. 6-2 a. : Slant matrices derived from $[H_8]$ and including $\overleftarrow{L}^3$

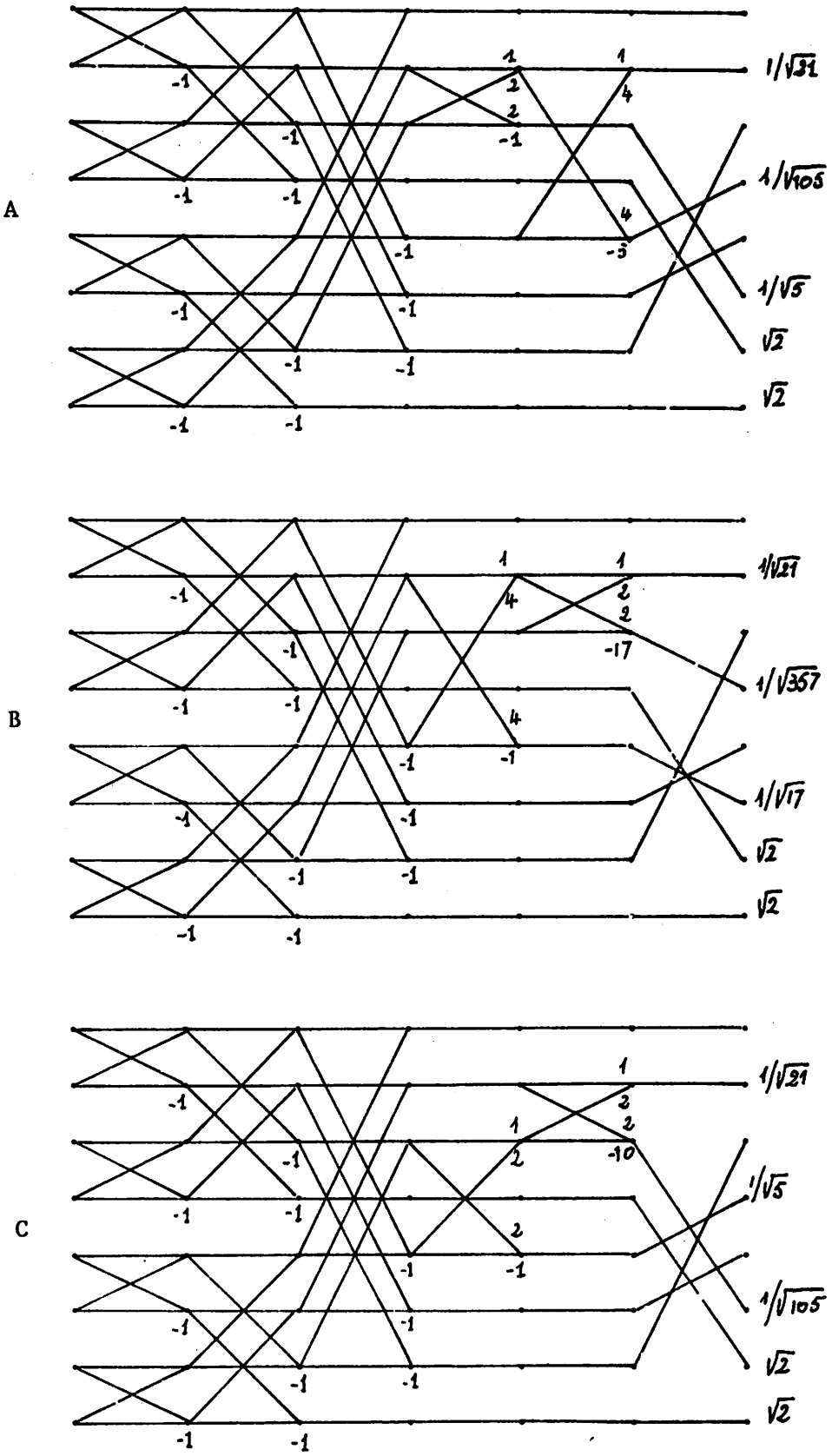


Fig. 6-2 b. Fast algorithms for the slant matrices of 6-2 a.

b) Cup slant vector :

A very similar development yields (k-2) rotation matrices of order 2 after the Haar transforms of the rows and therefore $(k-1)!/2$ possible choices. The corresponding fast algorithm requires :

additions : $2(2^n - 1) + 2^2 (2^{k-1} - 1) + 2$

multiplications : $k-3$

shifts : $2(k-2)$

normalizations : $2^n - 2^{n-2} - (k-2)$

In Fig. 6-3 , we show the matrix obtained after inclusion of $\vec{V}_0^2$ to $[H_8]$ with rows ordered by their ranks.

c) Slant vectors $\vec{L}^k$, $\vec{V}^k$ and $\vec{J}^k$:

Using both previous results, we need (2k-3) rotation matrices and we have $\left[\frac{(k-1)!}{4}\right]^2$ choices. The corresponding fast algorithm requires :

additions: $2(2^n - 1) + 2^2 (2^{k-1} - 1) + 2 (k-2)$

multiplications : $(2k - 5)$ if $k > 2$

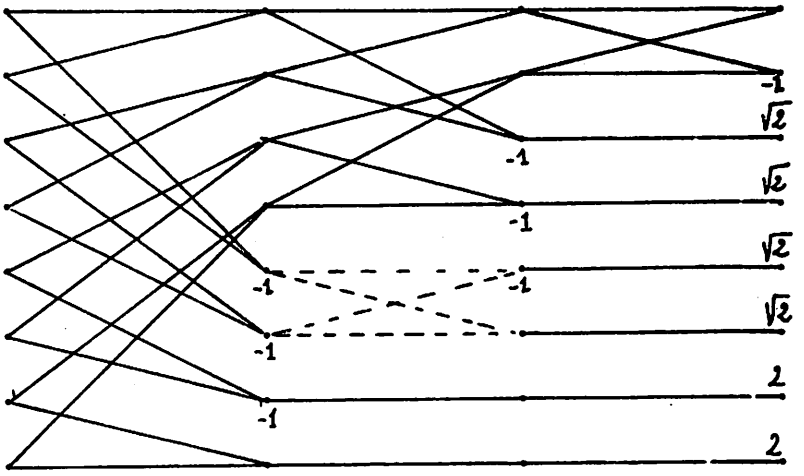
shifts : $2(2k - 3)$ (4)

normalizations: $2^n - 2^{n-2} - 2k + 3$

In Fig. 6-4, we present the slant transform of order 8 obtained from the Haar transform and which includes the vectors $\vec{L}_0^2$, $\vec{V}_0^2$, $\vec{J}_0^2$ (in this case our method gives only one transform). On Fig. 6-5, we present the slant transform of order 8 which includes all slant vectors with k = 2.

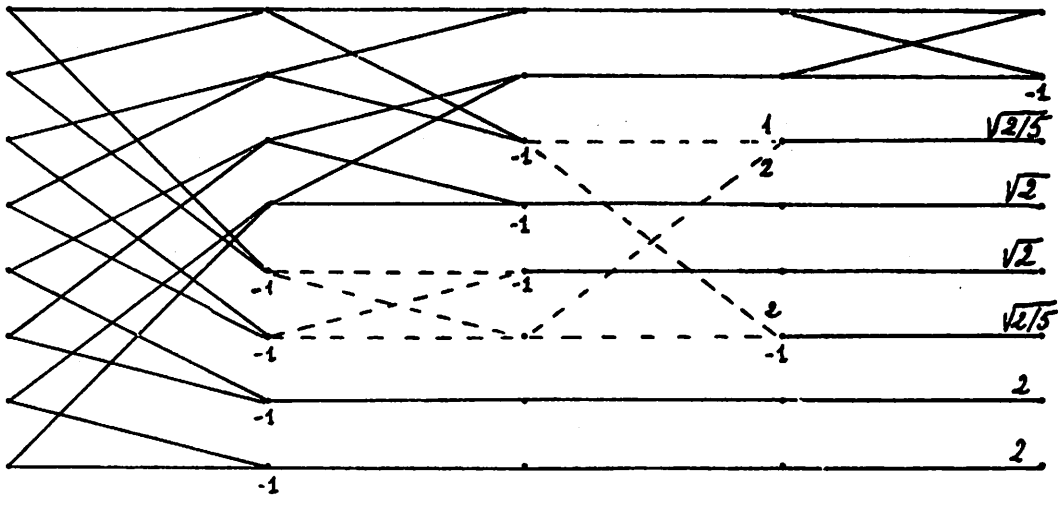
6-2. Inclusion of a complete set of slant vectors :

We consider now the slant transforms obtained from the Haar transform which include a complete set of slant vectors. In this case, we can



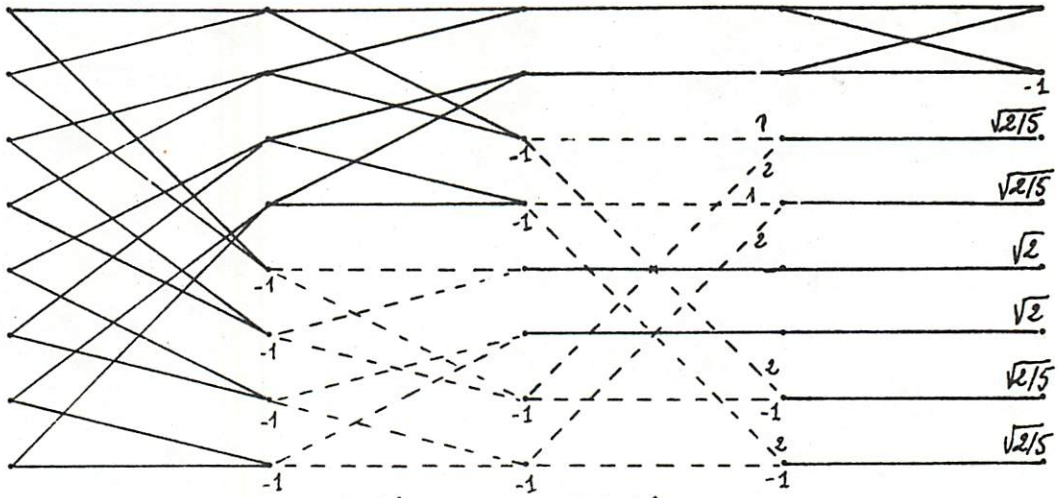
$$\frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ \sqrt{2} & -\sqrt{2} & -\sqrt{2} & \sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & \sqrt{2} \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}$$

Fig. 6-3. Slant matrix derived from $[H_8]$ and including $\vec{V}_0^2$ and corresponding fast algorithm



$$\frac{1}{\sqrt{8}}\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & \\
3 & 1 & -1 & -3 & 0 & 0 & 0 & 0 & \times\sqrt{2/5} \\
0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & \\
\sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 & \\
1 & -3 & 3 & -1 & 0 & 0 & 0 & 0 & \times\sqrt{2/5} \\
0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 &
\end{bmatrix}$$

Fig. 6-4. Slant matrix derived from $[H_8]$, including $\vec{L}_0^2$, $\vec{V}_0^2$, $\vec{J}_0^2$ and corresponding fast algorithm



$$\frac{1}{\sqrt{8}}\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & \\
3 & 1 & -1 & -3 & 0 & 0 & 0 & 0 & \sqrt{2/5} \\
0 & 0 & 0 & 0 & 3 & 1 & -1 & -3 & \sqrt{2/5} \\
1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & \sqrt{2} \\
0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & \sqrt{2} \\
1 & -3 & 3 & -1 & 0 & 0 & 0 & 0 & \sqrt{2/5} \\
0 & 0 & 0 & 0 & 1 & -3 & 3 & -1 & \sqrt{2/5}
\end{bmatrix}$$

Fig. 6-5. Slant matrix derived from $[H_8]$, including all slant vectors of size 4 and corresponding algorithm.

define the corresponding slant transform with a set of parameters and express the computational complexity in function of these parameters. Let $\left[SH(\ell_1, \ell_2, n)\right]$ denote the slant transform of order 2^n, obtained from the Haar transform $\left[H_{2^n}\right]$ and which includes the complete set of slant vectors $\mathcal{S}(\ell_1, \ell_2, n)$ with $2 \leqslant \ell_1 \leqslant \ell_2 \leqslant n$.

We first study how the inclusion of slant vectors defined on nested intervals can be efficiently implemented. Then, we express the generation of the matrices $\left[SH(\ell_1, \ell_2, n)\right]$ with recursive formulas and derive the computational complexity of these transforms.

a) Inclusion of slant vectors defined in nested intervals:

Let us assume that we have derived a slant tranform which includes the slant vectors $\vec{L}_0^{n-1}$ and $\vec{L}_1^{n-1}$ (and therefore $\vec{J}_o^{n-1}$ and $\vec{J}_1^{n-1}$). We show now that we need only two rotations of order 2 to include the slant vectors $\vec{L}^n$, $\vec{V}^n$, and $\vec{J}^n$. It is first obvious that:

$$\vec{V}^n = 1/\sqrt{2} \ (\vec{L}_0^{n-1} - \vec{L}_1^{n-1})$$

so that the rotation by the matrix $\left[F_2\right] = 1/\sqrt{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ will include $\vec{V}^n$ in the basis of the transform and also a vector $\vec{Z}^n = 1/\sqrt{2} \ (\vec{L}_0^{n-1} + \vec{L}_1^{n-1})$ which has the following components in the original Haar basis :

$$Z^n_i = \frac{2^{n-1} - 2i}{\sqrt{\frac{2^n(2^{2n-2} - 1)}{3}}} \qquad i=0,\ldots\ldots,2^{n-1}-1$$

$$Z^n_{2^{n-1}+i} = Z^n_i$$

Let us denote $\vec{S}^n$ the vector of $\left[H_{2^n}\right]$ with components :

$$S^n_i = 1/\sqrt{2^n} \quad \text{for} \quad i = 0,\ldots\ldots,2^{n-1}-1$$

$$S^n_i = -1/\sqrt{2^n} \quad \text{for} \quad i = 2^{n-1},\ldots\ldots,2^n-1$$

define the corresponding slant transform with a set of parameters and express the computational complexity in function of these parameters. Let $\left[SH(\ell_1, \ell_2, n)\right]$ denote the slant transform of order 2^n, obtained from the Haar transform $\left[H_{2^n}\right]$ and which includes the complete set of slant vectors $\mathcal{S}(\ell_1, \ell_2, n)$ with $2 \leqslant \ell_1 \leqslant \ell_2 \leqslant n$.

We first study how the inclusion of slant vectors defined on nested intervals can be efficiently implemented. Then, we express the generation of the matrices $\left[SH(\ell_1, \ell_2, n)\right]$ with recursive formulas and derive the computational complexity of these transforms.

a) Inclusion of slant vectors defined in nested intervals:

Let us assume that we have derived a slant tranform which includes the slant vectors $\vec{L}_0^{n-1}$ and $\vec{L}_1^{n-1}$ (and therefore $\vec{J}_o^{n-1}$ and $\vec{J}_1^{n-1}$). We show now that we need only two rotations of order 2 to include the slant vectors $\vec{L}^n$, $\vec{V}^n$, and $\vec{J}^n$. It is first obvious that:

$$\vec{V}^n = 1/\sqrt{2}\ (\vec{L}_0^{n-1} - \vec{L}_1^{n-1})$$

so that the rotation by the matrix $\left[F_2\right] = 1/\sqrt{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ will include $\vec{V}^n$ in the basis of the transform and also a vector $\vec{Z}^n = 1/\sqrt{2}\ (\vec{L}_0^{n-1} + \vec{L}_1^{n-1})$ which has the following components in the original Haar basis :

$$Z^n_i = \frac{2^{n-1} - 2i}{\sqrt{\frac{2^n(2^{2n-2}-1)}{3}}} \qquad i=0,\ldots\ldots,2^{n-1}-1$$

$$Z^n_{2^{n-1}+i} = Z^n_i$$

Let us denote $\vec{S}^n$ the vector of $\left[H_{2^n}\right]$ with components :

$$S^n_i = 1/\sqrt{2^n} \quad \text{for} \quad i = 0,\ldots\ldots,2^{n-1}-1$$

$$S^n_i = -1/\sqrt{2^n} \quad \text{for} \quad i = 2^{n-1},\ldots\ldots,2^n-1$$

This vector is not affected by the inclusion of $\vec{L}_0^{n-1}$ and $\vec{L}_1^{n-1}$.

We prove now that there is θ_n such that:

$$\begin{pmatrix} L^n \\ J^n \end{pmatrix} = \begin{bmatrix} \text{Sin } \theta_n & \text{Cos } \theta_n \\ \text{Cos } \theta_n & -\text{Sin } \theta_n \end{bmatrix} \begin{pmatrix} Z^n \\ S^n \end{pmatrix}$$

This relation requires :

$$\frac{(2^{n-1}-1) - 2i}{\sqrt{\frac{2^n(2^{2n-2}-1)}{3}}} \text{Sin } \theta_n + (1/2^n) \text{Cos } \theta_n = \frac{(2^n-1) - 2i}{\sqrt{\frac{2^n(2^n-1)}{3}}}$$

$$\frac{(2^{n-1}-1) - 2i}{\sqrt{\frac{2^n(2^{2n-2}-1)}{3}}} \text{Cos } \theta_n - (1/2^n) \text{Sin } \theta_n = \frac{\frac{(2^n-1)(2^{n-1}-1)}{3} - 2^n i}{\sqrt{\frac{2^n(2^{2n}-1)(2^{2n-2}-1)}{9}}}$$

$$i = 0, \ldots\ldots, 2^{n-1}-1$$

These equations give :

$$\text{Cos } \theta_n = \frac{2^{n-1}}{\sqrt{\frac{2^{2n}-1}{3}}}$$

$$\text{Sin } \theta_n = \sqrt{\frac{2^{2n-2}-1}{2^{2n}-1}}$$

The relations obtained for $i = 2^{n-1}, \ldots\ldots, 2^n-1$ are symmetric and yield the same value of θ_n.

b) Recursive generation of the slant transform $\left[SH(\ell_1, \ell_2, n)\right]$

We make use of the previous results to obtain the recursive generation of the slant transform $\left[SH(\ell_1, \ell_2, n)\right]$. The generation is obtained in three steps : the first leads to the transform $SH(\ell_1, \ell_1, \ell_1)$, the second to the transform $SH(\ell_1, \ell_2, \ell_2)$ and the last step to the transform $SH(\ell_1, \ell_2, n)$.

step 1 : recursive relations to obtain the Haar transform of order 2^{ℓ_1} followed by the rotations to include the slant vectors of size 2^{ℓ_1} occording to the algorithm of 6-4-2. We obtain $\left[SH(\ell_1, \ell_1, \ell_1)\right]$

step 2 : recursive relation between slant transforms $\left[SH(\ell_1, \ell, \ell)\right]$ and $\left[SH(\ell_1, \ell-1, \ell-1)\right]$ which leads to the generation of $\left[SH(\ell_1, \ell_2, \ell_2)\right]$ from $\left[SH(\ell_1, \ell_1, \ell_1)\right]$:

$$\left[SH(\ell_1, \ell, \ell)\right] = \left\{ \left[F_2\right]\left[F_2\right]\left[I_2\right].....\left[I_2\right] \right\} \text{ @ } \left[SH(\ell_1, \ell-1, \ell-1)\right]$$

followed by rotation of the rows 1 and 2^{n-1} by the matrix $\left[F_2(\theta_\ell)\right]$:

$$\left[F_2(\theta_\ell)\right] = \begin{bmatrix} \sin\theta_\ell & \cos\theta_\ell \\ \cos\theta_\ell & -\sin\theta_\ell \end{bmatrix}$$

where θ_ℓ is given by (5).

step 3: recurive relation similar to the Haar matrices recursive relation between $\left[SH(\ell_1, \ell_2, \ell)\right]$ and $\left[SH(\ell_1, \ell_2, \ell-1)\right]$ leading to the generation of $\left[SH(\ell_1, \ell_2, n)\right]$:

$$SH(\ell_1, \ell_2, \ell) = \left\{ \left[F_2\right]\left[I_2\right] ..... \left[I_2\right] \right\} \text{ @ } \left[SH(\ell_1, \ell_2, \ell-1)\right]$$

c) Computational complexity of the slant transform $SH(\ell_1, \ell_2, n)$

The previous recursive relations and the results of chapter 3 allows the computation of the number of elementary operations required by the fast algorithm for $\left[SH(\ell_1, \ell_2, n)\right]$. In the first step, we include the slant vectors $\vec{L}^1$, $\vec{V}^1$ and $\vec{J}^1$ into the basis of $2^{n-\ell_1}$ Haar transform. Relations (4) gives the number of elementary operations:

additions : $2^2 (2^{\ell_1} - 1) + 2\ell_1 - 6$; shifts : $2(2\ell_1 - 3)$

multiplications : $(2\ell_1 - 5)\ell_1$ for $\ell_1 > 2$, 0 for $\ell_1 = 2$

normalizations : $2^{\ell_1} - 2^{\ell_1 - 2} - 2\ell_1 + 2$

Then, we have $\ell_2 - \ell_1$ stages of computation for the step 2 requiring 6 Q additions, 2Q multiplications, Q shifts and $2Q + 2^{n-\ell_2}$ normalizations

with $Q = 2^{n-\ell_1+1} + \ldots\ldots + 2^{n-\ell_2} = 2^{n-\ell_1} - 2^{n-\ell_2}$

The last step performs a Haar transform of order $2^{n-\ell_2}$ and requires:
$2(2^{n-\ell_2})$ additions and $2^{n-\ell_2}$ normalizations.

Summing these results, we find finally the computational complexity of the slant transform $\left[SH(\ell_1, \ell_2, n)\right]$:

$$\begin{aligned}
&\text{additions: } 2^{n-\ell_1}(2^{\ell_1+2} + 2\ell_1 - 4) - 2^{n-\ell_2+2} - 2 \\
&\text{multiplications: } 2^{n-\ell_1+1}(\ell_1-2) - 2^{n-\ell_2} \text{ if } \ell_1 > 2,\ 2^{n-2}-1 \text{ if } \ell_1 = 2 \qquad (6) \\
&\text{shifts: } 2^{n-\ell_1+2}(\ell_1-1) - 2^{n-\ell_2} \\
&\text{normalizations: } 2^n - 2^{n-2} - 2^{n-\ell_1}(2\ell_1-4)
\end{aligned}$$

d) Slant Haar transform :

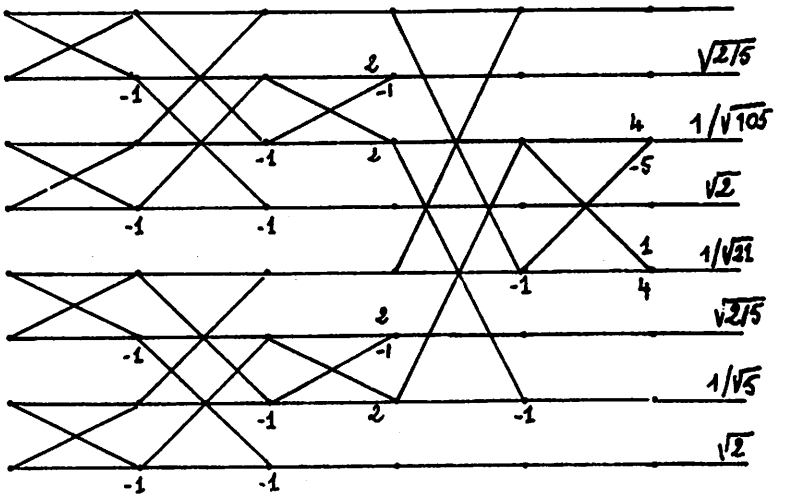
A particular case of the inclusion of a complete set of slant vectors is the slant transform $\left[SH(2,n,n)\right]$ which includes all orthonormal slant vectors into $\left[H_{2^n}\right]$; we call it the slant Haar transform. The recursive definition of the slant Haar transform only involves step 2 of the previous section and the slant Haar transforms, denoted $\left[SH_{2^n}\right]$, is recursively obtained by the Kronecker product $\left\{\left[F_2\right]\left[F_2\right]\left[I_2\right]\ldots\ldots\left[I_2\right]\right\} \otimes \left[SH_{2^{n-1}}\right]$ followed by the rotation of the rows 1 and 2^{n-1} by the matrix $\left[F_2(\theta_n)\right]$.

θ_n can be also expressed recursively :

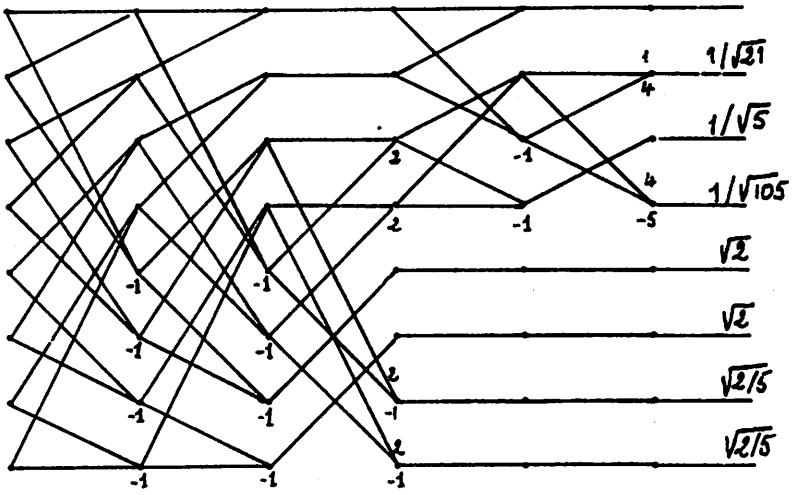
$$\text{Cos}\,\theta_{n-1} = \frac{\text{Cos}\,\theta_n}{\text{Sin}\,\theta_n}$$

$$\text{Sin}^2\theta_n = \frac{1}{1 + 4\text{Cos}^2\theta_{n-1}}$$

In Fig. 6-6 a we present the corresponding algorithm of order 8, in Fig 6-6 b a modified algorithm such that the coefficients are ordered, and in Fig. 6-6 c, we present the ordered slant Haar matrix



a) algorithm (natural order)



b) algorithm (ordered rows)

$$\frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\ 7 & 5 & 3 & 1 & -1 & -3 & -5 & -7 & \times 1/\sqrt{21} \\ 3 & 1 & -1 & -3 & -3 & -1 & 1 & 3 & \times 1/\sqrt{5} \\ 7 & -1 & -9 & -17 & 17 & 9 & 1 & -7 & \times 1/\sqrt{105} \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & \times \sqrt{2} \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & \times \sqrt{2} \\ 1 & -3 & 3 & -1 & 0 & 0 & 0 & 0 & \times \sqrt{2/5} \\ 0 & 0 & 0 & 0 & 1 & -3 & 3 & -1 & \times \sqrt{2/5} \end{bmatrix}$$

c) matrix

Fig. 6-6. Slant Haar transform of order 8

of order 8.

Computational complexity : By application of ( ), or by direct application of the results of chapter 3, we obtain the following number of elementary operations for the slant Haar transform:

additions : $2^{n+2} - 6$

multiplications: $2^{n-2} - 1$

shifts : $2^n - 2$

normalizations : $2^n - 2^{n-2} = 3\ 2^{n-2}$

As the slant Haar transform of order 4 (which is also the slant transform of order 4 presented in chapter 3) can be performed somewhat differently, we can trade, in the above results, 2^{n-1} additions and 2^{n-1} shifts for 2^{n-1} multiplications as we did for the slant transform presented in chapter 3.

Comparison with the slant transform of chapter 3, which we call now slant W-H transform to avoid confusion, shows about the same number of each elementary operation except for $(n-3)\ 2^n + 4$ fewer additions.

6-6. Generalization to combined slant vectors and their inclusion into the WHH transforms:

So far, we have considered the inclusion of the slant vectors only into the Haar basis. We have developed in some detail the slant transforms that could be thus generated and found a large number of slant transforms. In this section, we suggest different generalizations. The first considered is the inclusion of so called slant Walsh-Hadamard vectors.

6-6-1. Slant Walsh-Hadamard vectors:

The linear combination, through a unitary matrix $[U]$, of a set

of orthonormal slant vectors yields also a set of orthonormal vectors and thus a suitable set of replacement vectors. Obviously, the inclusion of these combined slant vectors requires not only the operations for the original slant transform, but also the operations for the rotation by the matrix $[U]$.

We consider in particular the W-H linear combination of slant vectors of same size and type but defined on disjoint intervals (variation of the parameter λ in the notation of 6-3-2). We obtain a set of piece-wise linear vectors of larger size than the original slant vectors. In Fig. 6-7, we show the combined slant vectors obtained from the vectors $\vec{L}^2_\lambda$ ($\lambda = 0,1,2,3$) and rotation by $[WH_4]$.

6-6-2. Inclusion into WHH transforms :

Instead of using the Haar matrix as original unitary transform, we may consider using any of the WHH transforms defined in chapter 3. In order to avoid some of the basis vectors of these transforms having a smaller size than the slant vectors of size 2^k, we must have:

$$k \leq n - h$$

where h is the parameter of the WHH family. Among the numerous possible transforms we can generate, we present now two examples.

a) Inclusion of $\vec{L}^3$ into $[WH_8]$:

With a method similar to that used for the Haar transform as original matrix, we obtain 3 different transforms. We present them in Fig. 6-8.

b) Slant W-H transform :

We consider the complete set of slant vectors which lead to the slant Haar transform and combine the vectors of same type and size by W-H transforms of corresponding orders. We obtain a set of 2^n

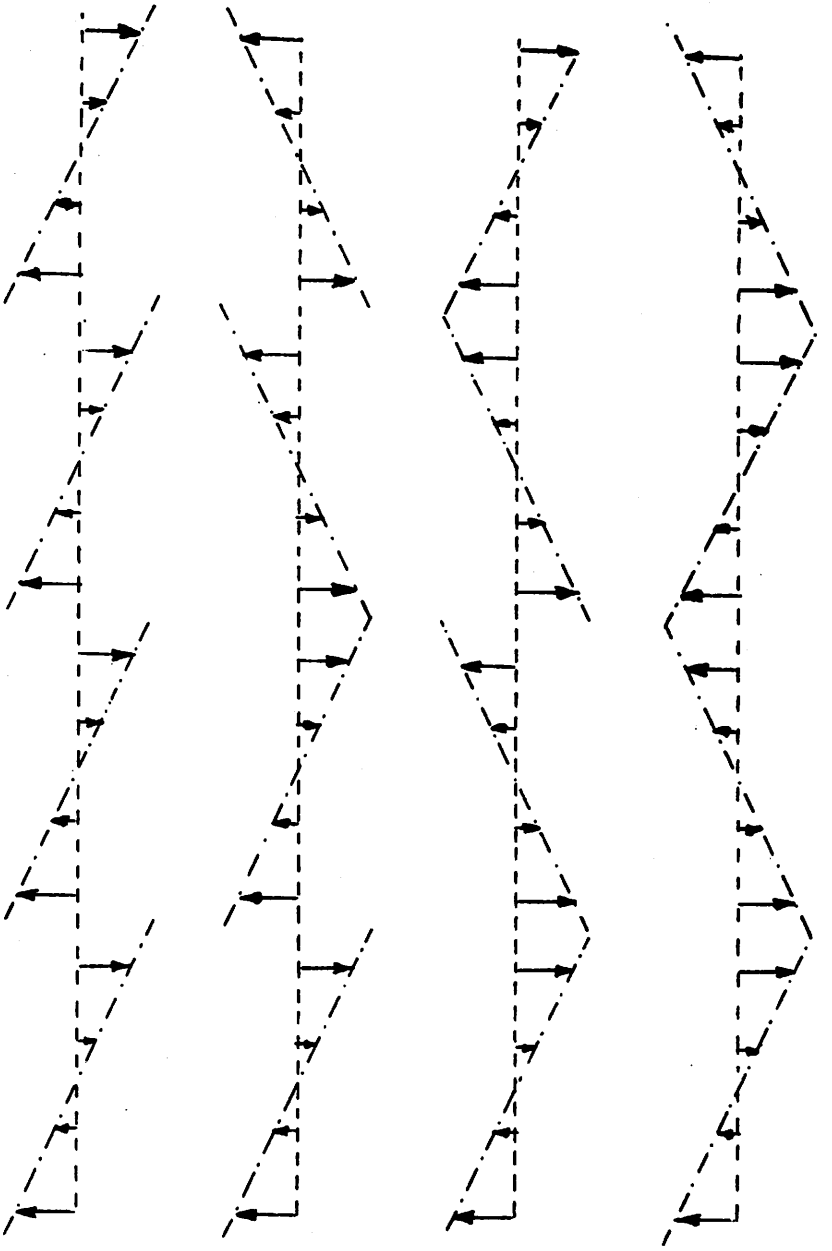


Fig. 6-7. Examples of slant W-H vectors

$$\frac{1}{\sqrt{8}}\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
7 & 5 & 3 & 1 & -1 & -3 & -5 & -7 & 1/\sqrt{21} \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & \\
7 & -1 & -9 & -17 & 17 & 9 & 1 & -7 & 1/\sqrt{105} \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & \\
1 & -3 & 3 & -1 & 1 & -3 & 3 & -1 & 1/\sqrt{5}
\end{bmatrix}$$

$$\frac{1}{\sqrt{8}}\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
7 & 5 & 3 & 1 & -1 & -3 & -5 & -7 & 1/\sqrt{21} \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & \\
7 & 11 & -27 & -23 & 23 & 27 & -11 & -7 & 1/\sqrt{357} \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & \\
3 & -5 & 3 & -5 & 5 & -3 & 5 & -3 & 1/\sqrt{17}
\end{bmatrix}$$

$$\frac{1}{\sqrt{8}}\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
7 & 5 & 3 & 1 & -1 & -3 & -5 & -7 & 1/\sqrt{21} \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & \\
1 & 1 & -3 & -3 & 3 & 3 & -1 & -1 & 1/\sqrt{5} \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & \\
7 & -13 & 9 & -11 & 11 & -9 & 13 & -7 & 1/\sqrt{105}
\end{bmatrix}$$

Fig. 6-8. Slant matrices derived from WH_8 and including L^3

orthonormal vectors which are the basis vectors of the slant W-H transform presented in chapter 3. We have given in 3-7 a recursive definition of this transform.

We note finally that the zonal relations between the Haar and W-H transforms (chapter 2) or between the WHH transforms (chapter 3) can be extended to the slant transforms with slant or combined slant vectors.

6-6-3. Step slant vectors:

If we consider the Kronecker product of a slant matrix, denoted $[SLT]$ with a WHH matrix, we obtain a unitary transform, denoted $[SST]$, $[SST] = [SLT] \otimes [WHH]$ with piece-wise constant basis vectors and these constants vary linearly.We present in Fig. 6-9 the step slant matrix given by the product $[SH_4] \otimes [WH_4]$.

6-6-4. Approximation of a vector with piece-wise linear vector:

By combination of slant linear vectors defined on disjoint intervals and the corresponding constant vectors with variable gains, we can approximate any vector with linear segments. However, the other vectors of the set have then constraints and in particular the constant vector will not be in the basis in general.

6-7. Conclusions :

In this chapter, we have mainly considered the definition of fast unitary transforms which include a set of slant vectors. All the transforms considered in this chapter are new, except the slant W-H transform. The results obtained with this transform in image processing give hope that the transforms developed here will be of interest.

$$\frac{1}{\sqrt{16}}\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
3 & 3 & 3 & 3 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -3 & -3 & -3 & -3 \; * \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & -3 & -3 & -3 & -3 & 3 & 3 & 3 & 3 & -1 & -1 & -1 & -1 \; * \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
3 & 3 & -3 & -3 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -3 & -3 & 3 & 3 \; * \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
1 & 1 & -1 & -1 & -3 & -3 & 3 & 3 & 3 & 3 & -3 & -3 & -1 & -1 & 1 & 1 \; * \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
3 & -3 & -3 & 3 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -3 & 3 & -3 & 3 \; * \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\
1 & -1 & -1 & 1 & -3 & 3 & 3 & -3 & 3 & -3 & -3 & 3 & -1 & 1 & 1 & -1 \; * \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
3 & -3 & 3 & -3 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -3 & 3 & -3 & 3 \; * \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
1 & -1 & 1 & -1 & -3 & 3 & -3 & 3 & 3 & -3 & 3 & -3 & -1 & 1 & 1 & -1 \; *
\end{bmatrix}$$

rows with * have a factor $1/\sqrt{5}$

Fig. 6-9. : Slant matrix $[SLH_4]$ @ $[WH_4]$

C H A P T E R VII

SOME APPLICATIONS OF FAST UNITARY TRANSFORMS

7-1. Introduction :

In the previous chapters, we have considered fast unitary transforms and their properties independently of their applications. Our approach has particularly stressed the recursive generation of fast unitary transforms, the evaluation of the computational complexity of their algorithms and the analysis of the computational errors. This chapter illustrates how fast unitary transforms can be applied to different domains and how the results of this thesis open new directions in these applications.

In the following, we first describe the main applications of fast unitary transforms : signal representation and dimensionality reduction, encoding and filtering. Our presentation is based on mean square error analysis because it is most commonly encountered in the literature and also because it leads to analytically tractable computations. However, it does not necessarly represent the state of the art in these applications nor is a limit of the application of our results.

All applications involve the computation of the covariance matrix of the transform coefficients when mean square error is used. An interesting application of the recursive definition of fast unitary transforms is the recursive expression of this covariance matrix, leading to a fast computation.

We shall see that all applications lead to a ternary trade-off between quantities which will be qualified as :

However, we may still have a good representation of the original sequence knowing only a few terms of the transformed sequence. We denote with "$\wedge$" the approximate values. We can the express the mean square error in the representation of the sequence :

$$\sum_{i=0}^{N-1} \left\| \hat{X}_i - X_i \right\|^2 = \sum_{i=0}^{N-1} \sum_{k=0}^{N-1} \left\| T_{ik} \right\|^2 \left\| \hat{Y}_k - Y_k \right\|^2$$

$$= \sum_{k=0}^{N-1} \left\| \hat{Y}_k - Y_k \right\|^2 \qquad (1)$$

If the input sequences has known second order statistics ,

$$\text{mean } (\mu_X)_i = E(X_i)$$

and covariance matrix $([R_{XX}])_{ij} = E\left(\left[X_i - (\mu_X)_i \right] \left[X_j^* - (\mu_X)_j^* \right] \right)$

then, we can express the second order statistics of the transformed sequence:

$$(\mu_Y)_i = [T]\, \vec{\mu}_X \qquad (2)$$

$$(R_{YY})_{ij} = E\left(\left[Y_i - (\mu_Y)_i \right] \left[Y_j^* - (\mu_Y)_j^* \right] \right)$$

$$= E \sum_k \left[T_{ik} (X_k - (\mu_X)_k) \right] \left[T_{j\ell} (X_\ell - (\mu_X)_\ell) \right]^*$$

$$= \sum_k T_{ik}\, T_{j\ell}^*\, E\left(\left[X_k - (\mu_X)_k \right] \left[X_\ell - (\mu_X)_\ell \right] \right)^*$$

Thus

$$[R_{YY}] = [T]\, [R_{XX}]\, [T]^{*t} \qquad (3)$$

Now, we assume that we take only a few coefficient formed sequence and neglect the others. By inverse transform of this reduced sequence, we obtain an approximation of the original sequence. The mean square error of this approximation is particularly easy to compute since , by Parseval's theorem it is the energy of the omitted coefficients. From (1) , (2) and (3), we have :

$$MSE(X) = \sum_{k \in \text{omitted coefficients}} \left([T]\, [R_{XX}]\, [T]^{*t} \right)_{kk} \qquad (4)$$

In representation problems, the parameters of interest are:

- the quality of the approximation (e. g. MSE)
- the number of maintained coefficients
- the complexity of the transform.

The trade-offs between these parameters have been only partially studied so far. When the complexity of the transform is not a problem, the Karhunen-Loeve transform (which diagonalizes the covariance matrix) has been shown to be optimal in many ways : for a fixed number of maintained coefficients [1], for a fixed level of the mean square error with Gaussian sequences [2]. However, when the complexity is considered to be critical, only very few results with a limited number of transforms have been reported [3] [4]. In section 6 of this chapter, we present a theoretical comparison of transforms for a first order Markov process and a MSE approximation measure.

7-2-2. Multiclass signal representation :

For a long time the feature selection stage of a pattern recognition scheme has been heuristic while the classification stage received more attention. Only recently , some efforts have been given to design automatic feature selectors [5] [6] [7]. It is not the place here to present these methods; however, we want to stress the use of unitary transforms in feature selection.

The rotation of the axes of the pattern space by a unitary transform may give axes more meaningful to represent an separate the different classes. The K-L transform is again optimal [8] and is the bas of factor analysis (where classes are unknown). However, other unitary transforms may perform closely with less computational time and therefore are of interest. Andrews [9] and Carl [10] have applied

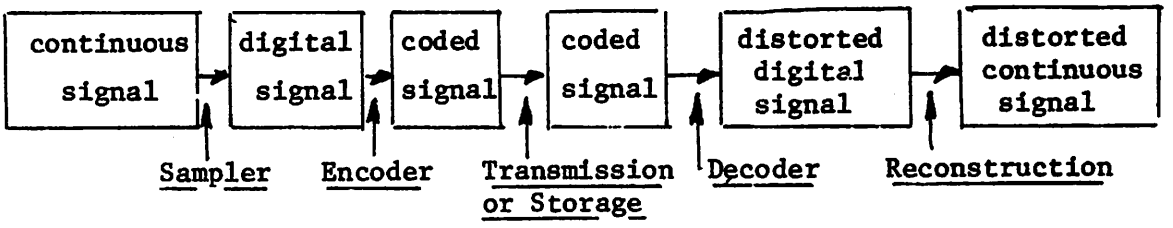
this approach to a large variety of patterns and found promis results.

7-3. Signal encoding :

The problem of signal encoding is very similar to the representation problem ; however, the representation is no longer the first step in the processing of the signal but the ultimate goal. After quantization and coding of the maintained coefficients, we have a bit rate which expresses clearly the performance of the encoding scheme, for transmission or storage. This is the most popular application of the fast unitary transforms for a wide variety of signals : speech [11] and image encoding [12] [13] [14] are the most common examples. In this section, we analyse the basic problems of signal encoding and discuss the possible options.

7-3-1. Basic problems of signal encoding :

Signal encoding and decoding are performed schematically as follows:



Signal encoding techniques take advantage of the redundancies in the signals which belong to the same class to reduce the transmitted or stored data. Statistical characteristics of the class of signals are necessary to analyse an encoding scheme and in section 7-6, we consider the case of first order Markov process which models especially well speech and image signals. For a first order Markov process, one single parameter characterizes the signal complexity. In the evaluation of the performance of an encoding scheme, the parameters of interest are the bit rate,

the distortion level (with a distortion measure) and the computational complexity.

a) distortion : If a human is the ultimate user, through his senses, of the decoded signal, the distortion measure must consider the subjective effect of the errors. To our knowledge, such distortion measure is still under study and commonly the mean square error is considered, mainly because it leads to a tracktable analysis, but also because it is a rough approximation of the human perception of distortion [12]. A better but still somewhat unsatisfactory measure is the frequency weighted mean square error [13] [15]. The global distortion of the encoded signal appear at various stages of the encoding process:

- sampling error : since most signals of interest are frequency limited and since the human senses (hearing and vision in particular) are mostly sensitive to low frequencies, sampling errors can usually be made negligible.

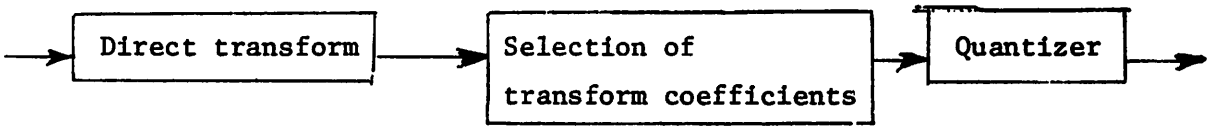
- transmission (storage) errors: we neglect them in the following.

- encoding errors on sampled data due to the data compression technique and depending on this technique.

b) computational complexity : It controls the equipment complexity and the computation time. The constraints on complexity for various applications are certain quite different. In this thesis, we shall measure the computational complexity uniquely by the total number of elementary operations required in the fast algorithms. By so doing, we neglect an important constraint of practical encoders : the buffer size necessary for each encoding technique.

7-3-2. Unitary transform encoding techniques :

In a transform encoding technique (also called block quantization technique), the samples are treated by blocks. These blocks are transformed by a fast unitary transform (therefore the blocks have usually 2^n elements). In the transform domain, many coefficients have a small amplitude and may be discarted without noticeable loss of quality. The remaining coefficients are then quantized. The encoder has the following structure :



while the decoder performs the inverse transform. A transform encoding technique is then determined by the transform, the blocksize, the method of selection of transmitted coefficients and the quantizer. We comment on them :

transforms : only a few transforms have been considered : Fourier, W-H, Haar, Slant, Discrete Cosine (a variation of the Fourier transform defined recently).[1] In fact, it is known that, in its K-L basis, the signal is represented by uncorrelated coefficients which are independent if we assume (experimental evidence somewhat justified by the law of large numbers) that they are Gaussian [1]. The rate-distortion theory tells us that these coefficients can be optimally encoded but without concern for speed of computation. If fast unitary transforms are used instead, we have a gain in computational complexity achieved at the expense of a larger distortion resulting from the remaining correlation among the coefficients.

selection of transmitted coefficients: There are many ways

[1] The Discrete Cosine matrix is given by $(C_{2^n})_{i,j} = \cos\left(\frac{\pi i(2k+1)}{2N}\right) \times \begin{cases} 1 \text{ if } i=0 \\ \sqrt{2} \text{ if } i\neq 0 \end{cases}$

to select the transmitted coefficients and to quantize them. Our goal here is to concentrate on the effect of the choice of the transform and therefore we compare in section 7-6 theoretical bounds obtained from information theory for different transforms ; the corresponding selection and quantization rules parallel the theoretically optimal rules used for the Karhunen-Loeve transform.

7-4. Signal filtering ;

A common problem in electronics as well as in digital circuits is to design filters that improve the signal-to-noise ratio of a noisy signal. Filters are analysed in the signal (time) domain or equivalently in the spectral (frequency) domain. Generalized filters analysed in the transform domain have been presented by Pratt [17]. In the following, we consider the simple example of a stationary vector signal $\vec{X}$ mixed with an uncorrelated additive noise $\vec{Y}$. We denote their respective covariance matrices by $[R_{XX}]$ and $[R_{YY}]$. The noisy signal is transformed by a unitary transform [T] and then filtered by a matrix multiplication with the filter matrix $[\mathcal{H}]$. The inverse transform $[T]^{*t}$ is then applied to obtain an estimate $\hat{\vec{X}}$ of the signal. It is well known that the optimal filter in the signal domain is given by

$$[H] = [R_{XX}] \; ([R_{XX}] + [R_{YY}])^{-1}$$

and the mean square error of the resulting estimate is

$$\varepsilon^2 = \mathrm{Tr}\left\{ [R_{XX}] \; ([R_{XX}] + [R_{YY}])^{-1} \; [R_{YY}] \right\} \qquad (5)$$

If before filtering, we transform the signal by a unitary transform [T], we have an optimal filter $[\mathcal{H}]$ given by :

$$[\mathcal{H}] = [T] \; [R_{XX}] \; [T]^{*t} (\; [T] \; ([R_{XX}] + [R_{YY}]) \; [T]^{*t})^{-1}$$

$$[\mathcal{H}] = [T] \quad [H] \; [T]^{*t} \qquad (6)$$

and the mean square error is still given by (5).

The interesting consequence of this result is that we may be able to simplify the filtering operation if $[\mathcal{H}]$ is easier to perform than [H]. Or, for a suboptimal filter, we may constrain $[\mathcal{H}]$ and then find the best choice for [T]. Pearl [18] has considered the case of a diagonal filter matrix $[\mathcal{S}]$, called scalar filtering, which is optimal (under the constraint of a diagonal matrix) when :

$$[\mathcal{S}] = \text{Diag} \frac{[C_{XX}]_{ii}}{[C_{XX}]_{ii} + [C_{YY}]_{ii}} \tag{7}$$

where $[C_{XX}] = [T]\ [R_{XX}]\ [T]^{*t}$

$[C_{YY}] = [T]\ [R_{YY}]\ [T]^{*t}$

The resulting mean square error is then :

$$MSE(X) = 1 - \frac{1}{N} \sum_{i=0}^{N-1} \frac{[C_{XX}]_{ii}^2}{[C_{XX}]_{ii} + [C_{YY}]_{ii}} \tag{8}$$

In general, we want to minimized the mean square error given by:

$$MSE(X) = Tr \left\{ [C_{XX}] - 2\ [\mathcal{H}]\ [C_{XX}] + [\mathcal{H}]\ [\mathcal{H}]\ ([C_{XX}] + [C_{YY}]) \right\}$$

with some specified constraints on the matrix $[\mathcal{H}]$.

As in the previous applications, a generalized filter is characterized by a ternary trade-off between :

- filter performance or simplicity
- approximation of the signal or signal-to-noise ratio
- computational complexity of the transform.

In section 7-6, we consider this trade-off for a first order Markov process and we complete earlier results reported by Pratt [17] and Ahmed & al.[19]. These results show that transform filters can perform very close to optimal filters and with high computational efficiency.

7-5. Recursive relation of covariance matrices in the transform domain :

In the applications of fast unitary transforms presented in the previous sections, the covariance matrix of the transform coefficients and particularly its diagonal elements play an important role in evaluating the performance or even in determining parameters.

In this section, we use the recursive definition of the fast unitary transforms to derive a recursive relation for the covariance matrices of the transform coefficients. Then, using this relation, we develop a fast procedure to compute the elements of these covariance matrices.

7-5-1. Recursive relation :

We first examine the covariance matrices in a generalized Kronecker product, denoted $[C] = \{\mathcal{A}\} \otimes \{\mathcal{B}\}$. The two sets of matrices $\{\mathcal{A}\}$ and $\{\mathcal{B}\}$ have respectively m matrices $[A^i]$ of order n and n matrices $[B^j]$ of order m. We know, from chapter 3, that :

$$[C] = [P]^t \; [\text{Diag}\{\mathcal{A}\}] \; [P] \; [\text{Diag}\{\mathcal{B}\}] \tag{9}$$

We wish to compute $[R_{YY}]$ as given by (3) for the unitary matrix [C]

$$[R_{YY}] = [C] \; [R_{XX}] \; [C]^{*t} \tag{10}$$

From (9) and (10), we can express $[R_{YY}]$ as:

$$[R_{YY}] = [P]^t \; [\text{Diag}\{\mathcal{A}\}] \; [P] \; [\text{Diag}\{\mathcal{B}\}] \; [R_{XX}] \; [\text{Diag}\{\mathcal{B}\}]^{*t} \; [P]^t \; [\text{Diag}\{\mathcal{A}\}]^{*t} \; [P] \tag{11}$$

In order to take advantage of the hermitian property (symmetric elements across the main diagonal are complex conjuguate) or even

Toeplitz property (element $R_{XX\ i,i-j}$ are equal for all j) of the matrix $[R_{XX}]$ of order (mn), we partition it as follows :

$$[R_{XX}] = \begin{bmatrix} \overbrace{[R_{XX}^{0,0}]}^{m} & \overbrace{[R_{XX}^{0,1}]}^{m} & \cdots\cdots & [R_{XX}^{0,n-1}] \}m \\ [R_{XX}^{0,1}]^{*t} & [R_{XX}^{1,1}] & \cdots\cdots & [R_{XX}^{1,n-1}] \}m \\ \vdots & & & \vdots \\ [R_{XX}^{0,n-1}]^{*t} & [R_{XX}^{1,n-1}]^{*t} & \cdots\cdots & [R_{XX}^{n-1,n-1}] \end{bmatrix}$$

so that $[\text{Diag } \{\mathcal{A}\}]\ [R_{XX}]\ [\text{Diag } \{\mathcal{B}\}]^{*t}$ is a matrix $[R_{ZZ}]$ which can be partitioned into :

$$[R_{ZZ}] = \begin{bmatrix} \overbrace{[R_{ZZ}^{0,0}]}^{m} & \overbrace{[R_{ZZ}^{0,1}]}^{m} \cdots\cdots & \overbrace{[R_{ZZ}^{0,n-1}]}^{m} \} m \\ [R_{ZZ}^{1,0}] & [R_{ZZ}^{1,1}] \cdots\cdots & [R_{ZZ}^{1,n-1}] \} m \\ \vdots & & \vdots \\ [R_{ZZ}^{n-1,0}] & [R_{ZZ}^{n-1,1}] \cdots\cdots & [R_{ZZ}^{n-1,n-1}] \} m \end{bmatrix}$$

$$\text{with } [R_{ZZ}^{k,\ell}] = [B^{k}]\ [R_{XX}^{k,\ell}]\ [B^{\ell}]^{*t} \qquad (12)$$

We prove now a property of this matrix $[R_{ZZ}]$:

$[R_{ZZ}^{k,\ell}]$ is hermitian as $[R_{XX}]$ is hermitian :

$$[R_{ZZ}^{k,\ell}]^{*t} = [B^{\ell}]\ [R_{XX}^{\ell,k}]\ [B^{k}]^{*t} = [R_{ZZ}^{\ell,k}]$$

For $k = \ell$, we have the covariance matrices of the previous stage of computation and for $k \neq \ell$, we have the cross-correlation matrices of the previous stage. Using (11) and expanding (12), we find :

$$[R_{YY}]_{um+w,vm+z} = \sum_{u'=0}^{n-1} \sum_{v'=0}^{n-1} A^{w}_{uv'}\ R_{ZZ\ w,z}^{v',u'}\ A^{z}_{vu'}{}^{*} \qquad (13)$$

We prove now that $[R_{YY}]^{*}_{vm+z,um+w} = [R_{YY}]_{um+w,vm+z}$:

$$[R_{YY}]^{*}_{vm+z,um+w} = \sum_{u'=0}^{n-1} \sum_{v'=0}^{n-1} A^{w}_{uu'} \; R^{v',u'\,*}_{ZZ\,z,w} \; A^{z}_{vv'}$$

$$= [R_{YY}]_{um+w,vm+z} \quad \text{(using the hermitian property of } [R_{ZZ}^{u',v'}] \text{)}$$

If we are only interested in the diagonal terms of $[R_{YY}]$, we obtain from (13) :

$$[R_{YY}]_{um+w,vm+w} = \sum_{u'=0}^{n-1} \sum_{v'=0}^{n-1} A^{w}_{uv'} \; R^{v'u'}_{ZZ\,ww} \; A^{w\,*}_{uu'} \qquad (14)$$

showing a dependence only on the diagonal terms of the correlation matrices $[R_{ZZ}^{k,\ell}]$

7-5-2. Fast computation :

Relations (12) and (13) provide a computational method for the covariance matrix $[R_{YY}]$ which uses efficiently the particular structure of the covariance matrix $[R_{XX}]$. Let us denote by P_1 and P_2 the respective number of elementary operations required by the computations corresponding to (12) and (13) and by p_n^{w} and p_m^{k} the numbers of these elementary operations required by the fast algorithms for the matrices $[A^{w}]$ and $[B^{k}]$. Then, if we compute by (12) all the matrices $[R_{ZZ}^{k,\ell}]$ and by (13) all the coefficients of $[R_{YY}]$ without using the mentioned properties of these matrices, we find easily that:

$$P = P_1 + P_2 = 2\,mn \left(\sum_{w=0}^{m-1} p_n^{w} + \sum_{k=0}^{n-1} p_m^{k} \right) \qquad (15)$$

This result is in fact the required number of operations when $[R_{YY}]$ is computed directly from (10) using the fast algorithm for [C] , since [C] requires (see (4) of chapter 3)

$$\sum_{w=0}^{m-1} p_n^{w} + \sum_{k=0}^{n-1} p_m^{k} \quad \text{operations.}$$

However, the properties of the matrices $[R_{ZZ}^{k,\ell}]$ and of the coefficients $[R_{YY}]^{*}_{um+w,vm+z}$ help reduce the computation of $[R_{YY}]$:

$$P_1 = \sum_{k=0}^{m-1} \sum_{\ell=k}^{m-1} m(p_m^{\,k} + p_m^{\,\ell}) = mn \sum_{k=0}^{m-1} p_m^{\,k}$$

and similarly for P_2, so that :

$$P = mn \left(\sum_{k=0}^{n-1} p_m^{\,k} + \sum_{w=0}^{m-1} p_n^{\,w} \right) \qquad (16)$$

This result shows a reduction by half of the number of elementary operations required by the direct computation with a fast algorithm from (10). Moreover, when a transform is defined recursively, P includes the computation of the covariance matrices at all the previous stages while with the direct computation, we start the computation from scratch at each stage. When the covariance matrices of the successive stages of computation have to be obtained, this method brings a substantial saving much over the reduction by half.

If the original covariance matrix $[R_{XX}]$ has also the Toeplitz property, we have an additional gain when the matrices $[B^k]$ are identical. Then, it is easy to show that $[R_{XX}^{k,\ell}] = [R_{XX}^{|k-\ell|}]$ so that we have only n different matrices $[R_{ZZ}^{k,\ell}]$ and P_1 is then reduced to

$$P_1 = mn\, P_n$$

7-6. Comparison of transforms for a first order Markov process :

In this section, we consider that the signal is a first order Markov process with correlation function $R_{i,j} = e^{-\alpha |i-j|}$ $i,j = 0,.....,n-1$. In the following we consider that this signal is represented, encoded or filtered (when white noise is added) with some fast unitary transforms. We compare their performances for each specific task.

7-6-1. Signal representation :

In signal representation, the basic parameters to consider are the number of maintained coefficients, the mean square error and the computational complexity. In the following, we weight equally all the different elementary operations (additions, multiplications, normalizations and shifts) . Our results therefore are only of theoretical interest but , using weights which reflect time or cost, similar curves can be obtained for each particular implementation. In Fig. 7-1, we present some typical curves obtained for a first order Markov process with $\alpha = 0.05$ and 1/4 of the transform coefficients maintained. The points of the curves are obtained for different blocksizes. When the block size increases, there is an improvement of the representation at the expense of a higher complexity. The blocksize is indicated whenever possible. In these curves, we have taken as reference for the mean square error the error obtained with the K-L transform of order 128 and we have plotted the loss in dB. Notice the interesting result obtained with the slant transforms.

7-6-2. Signal encoding :

We begin by a brief review of previous works.

a) review of previous works: Many papers present transform techniques for signal encoding but very few include an analysis of their performances.

Goblick and Holsinger [20] have compared the rate-distortion bound and a theoretical bound for the Fourier transform in the case of a first order Markov process.

Pearl [21] has extended this approach to some other transforms and also proposed a distance measure that we discussed in [22].

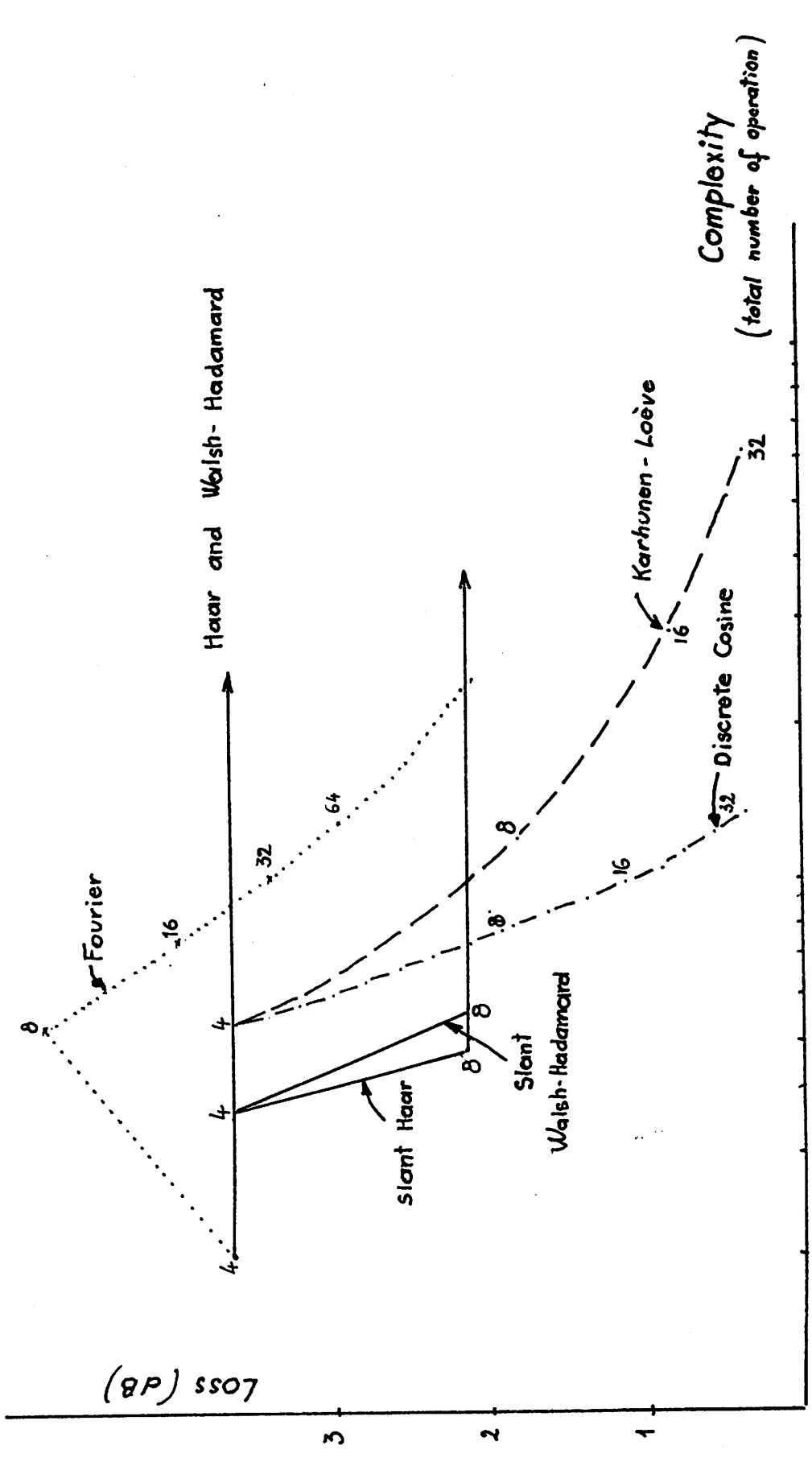


Fig. 7-1. Representation of a first order Markov process (= 0.05 , 1/4 maintained coefficients)

Habibi and Wintz [12] have compared experimental errors with the theoretical curves and also considered the subjective appreciation of images for Fourier and Walsh-Hadamard encoding techniques.

Schwartz,and al. [23] have studied the statistical properties of the transform coefficients for the Fourier, W-H and Haar transforms and deduced optimal encoding zones in the transform domain.

Pratt and al. have given some comparisons of fast unitary transforms in [24].

Ahmed and al. [19] have also compared the discrete cosine transform with the common transforms.

In the following, we emphasize two parameters of importance which we have mentioned earlier : the image complexity and the computational complexity. Our approach is similar to these of Goblick and Holsinger or Pearl. We present now its theoretical basis from the rate-distortion theory.

b) <u>rate-distortion with fast unitary transforms</u>:

We assume in this derivation that :

- the transform coefficients have a normal distribution.
- they are coded independently.

Therefore, we can evaluate the rate-distortion bound for each coefficient. It is known [16] that the rate-distortion curve is given in this case by the parametric equations :

$$R(\mu) = \frac{1}{2N} \sum_{i=0}^{N-1} \log_2 \frac{\sigma_i^2}{\mu} \qquad (17)$$

$$D(\mu) = \frac{1}{N} \left(\sum_{\sigma_i^2 > \mu} \mu + \sum_{\sigma_i^2 \leq \mu} \sigma_i^2 \right)$$

We present in Fig. 7-2 the corresponding curves for $\alpha = 0.05$ and a

blocksize of 32. Note that the Haar and W-H transforms and also the Slant Haar and Slant Hadamard transforms have very close performances This fact is a consequence of the zonal relations between these transforms which we proved in chapter 2 and 3. Note also that the semi-logarithmic scale adopted for these curves gives parallel lines at low distortion levels. This occurs for mean square errors lower than the smallest variance of the transform coefficients since we have, from (17) :

$$R(D) = \frac{1}{2N} \log_2 \frac{\prod_{i=0}^{N-1} \sigma_i^2}{D^N} \qquad (18)$$

and therefore we can express the performance of the transforms with a bit difference d_{R_1,R_2} with the Karhunen-Loeve transform.

For two transforms denoted with subscripts $_1$ and $_2$, we have :

$$d_{R_1,R_2} = R_1(D) - R_2(D) = \frac{1}{N} \log_2 \prod_{i=0}^{N-1} \left(\frac{\sigma_{1i}}{\sigma_{2i}} \right)$$

Note that d_{R_1,R_2} is no longer dependent on the distortion D.

To illustrate the dependence of the performance upon the source statistics, we have plotted in Fig. 7-3. the bit difference as function of the signal complexity α for a fixed block size 32. It is noteworthy that the curve for Fourier intersects the other curves, showing that no general conclusion can be obtained on the relative merits of the Fourier transform with the other transforms.

On Fig. 7-4. , we show the variation of the bit difference when the block size increases. As expected, the bit difference between the Fourier and Karhunen-Loeve transforms decreases. It can be shown that the bit difference for the Haar transform increases to a limit and it is conjectured that the Walsh-Hadamard and Slant transforms have a similar limiting behaviour.

Finally in Fig. 7-5. we have plotted the bit difference with the rate-distortion bound as a function of the computational complexity.

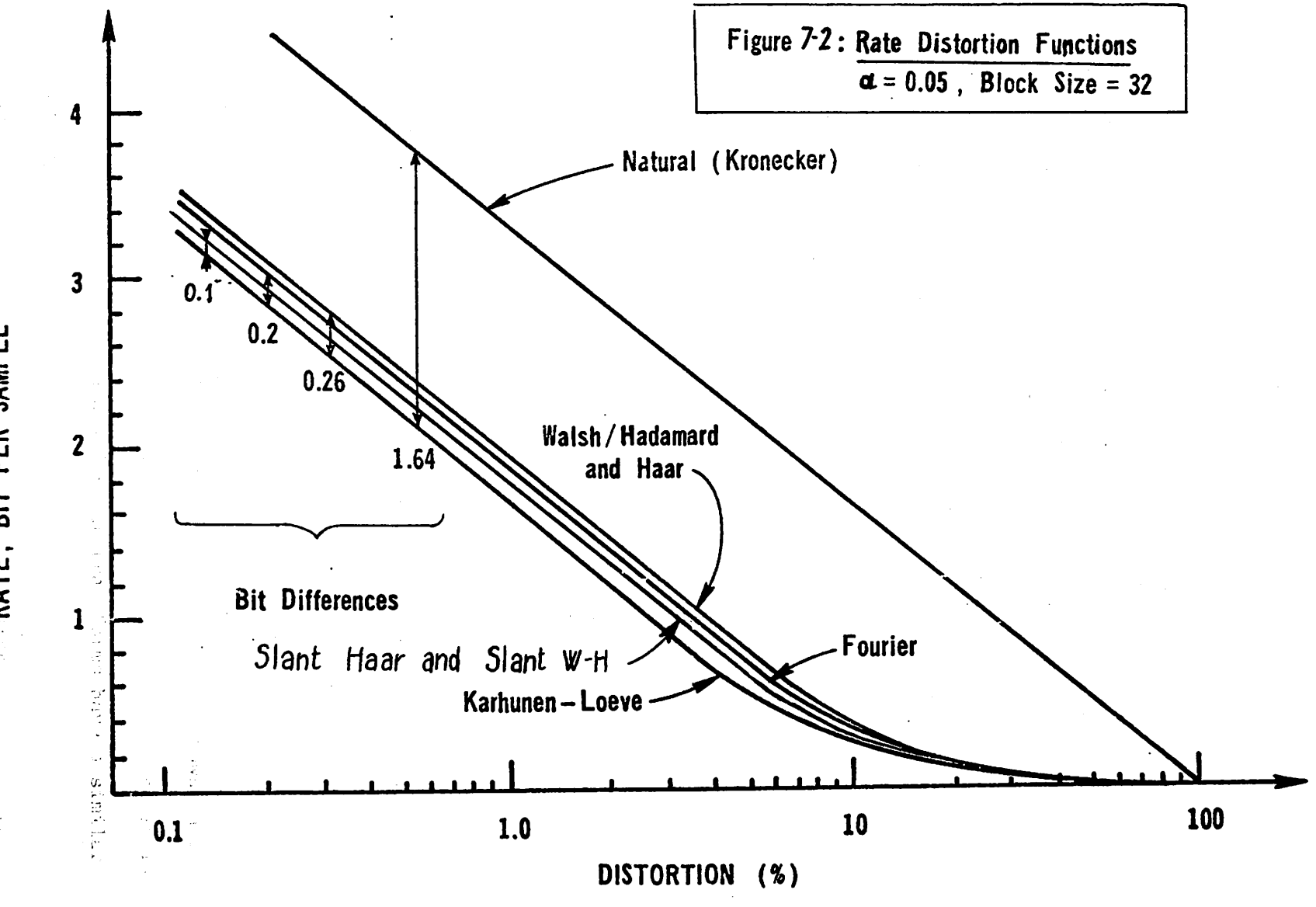


Figure 7-2: Rate Distortion Functions
α = 0.05, Block Size = 32
RATE, BIT PER SAMPLE
4
3
2
1
0.1
1.0
10
100
DISTORTION (%)
Natural (Kronecker)
0.1
0.2
0.26
1.64
Bit Differences
Walsh/Hadamard
and Haar
Slant Haar and Slant W-H
Karhunen-Loeve
Fourier

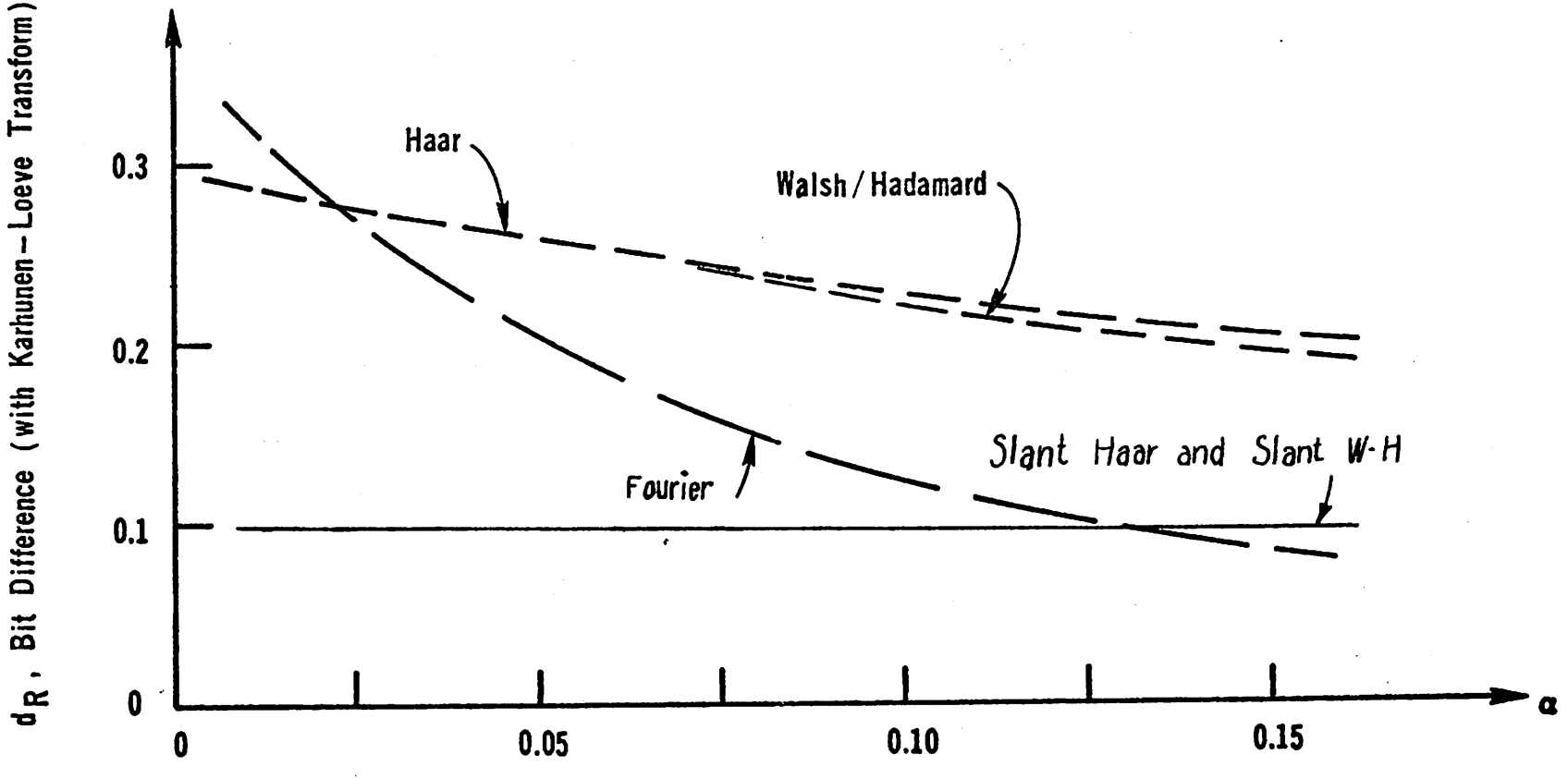


Figure 7-3: Effect of Statistics on Rate Difference
Block Size = 32

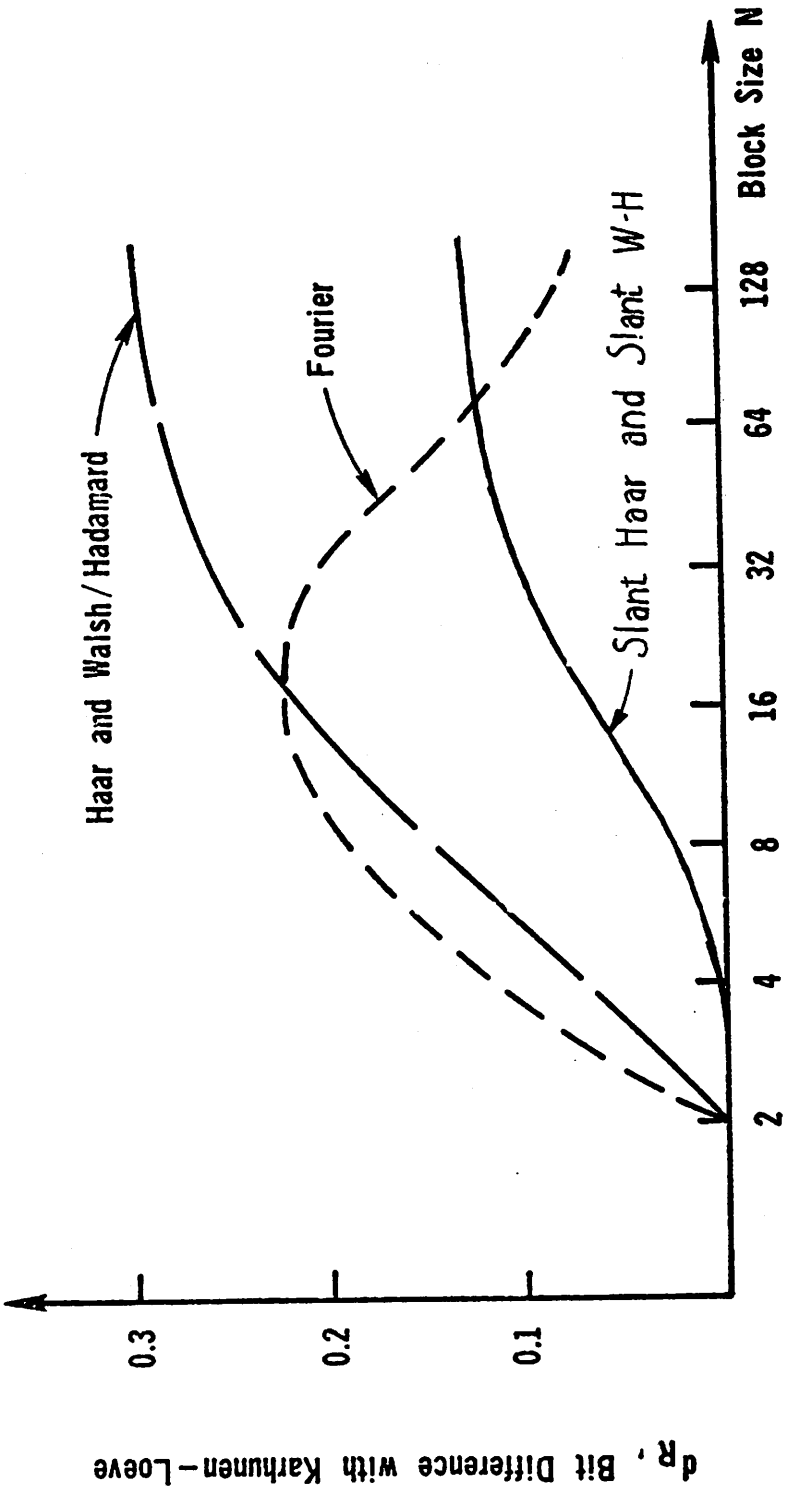


Figure 7-4: Effect of Block Size on Rate Difference
$\alpha = 0.05$

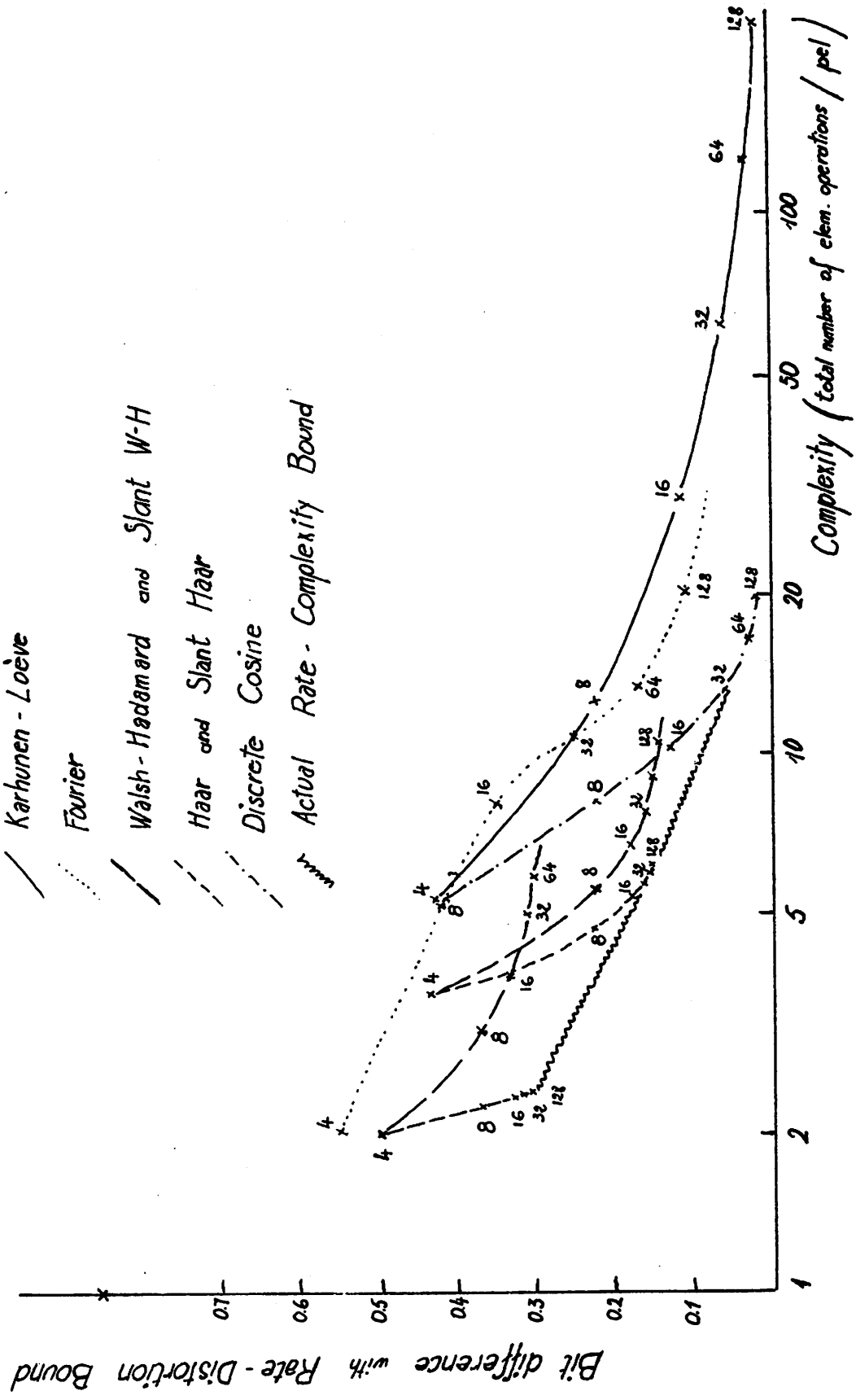


Fig. 7-5. Comparison of transforms :
Bit difference function of computational complexity

α = 0.05

when varrying the block size. The rate-distortion bound is the limit obtained for the Karhunen-Loeve transform of infinite block size: this limit is

$$R(D) = \frac{1}{2} \log_2 (1 - e^{-2\alpha}) - \frac{1}{2} \log_2 D$$

This figure shows the trade-off between performance and computational complexity and we have plotted on the figure an actual rate-complexity bound valid for the transforms considered.

7-6-3. Signal filtering :

Pratt [17] and Ahmed & al. [19] have presented some comparisons in the case of scalar filtering of a first order Markov process with additive white noise. We present in Fig. 7-6 and 7-7 similar curves with a different scale for the mean square error : again we consider the loss over the performance of the Karhunen-Loeve transform of order 128. For Fig. 7-6 , the signal to noise ratio is 1 and for Fig. 7-7. the signal to noise ratio is 50. Again, on this theoretical example, we find that no intrinsic ranking of the transforms exists and we suggest again a loss-complexity bound.

7-7. Conclusions :

The application of fast unitary transforms to signal representation, encoding and filtering leads to a ternary trade-off between performance, approximation and computational complexity. In the case of a first order Markov process, we have presented some theoretical curves to express this trade-off.

In these applications the unified structure of the fast unitary transforms appears useful to

- generate new transforms such as the slant Haar transform which seems particularly interesting in the case of a Markov process.

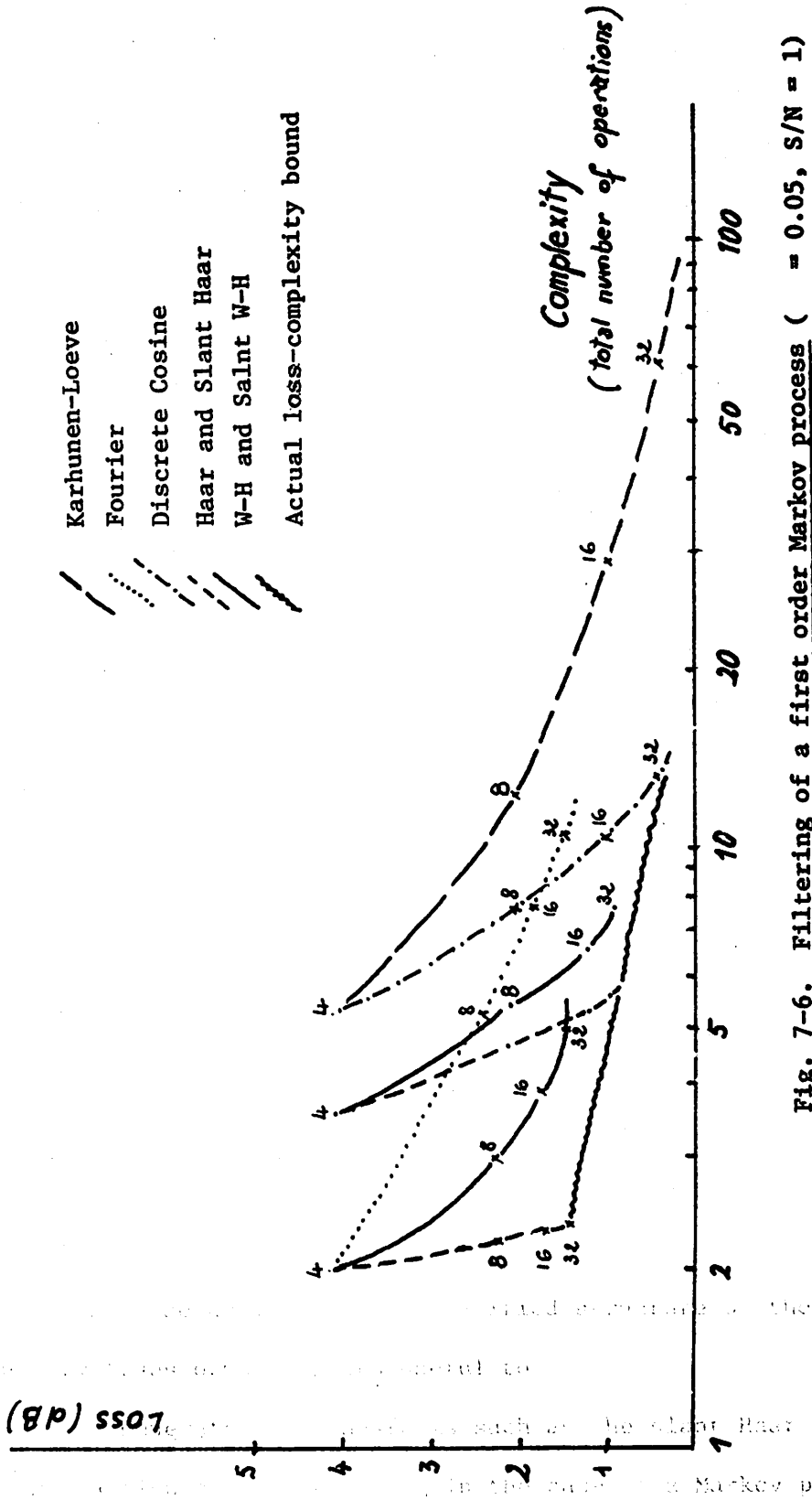


Fig. 7-6. Filtering of a first order Markov process (= 0.05, S/N = 1)

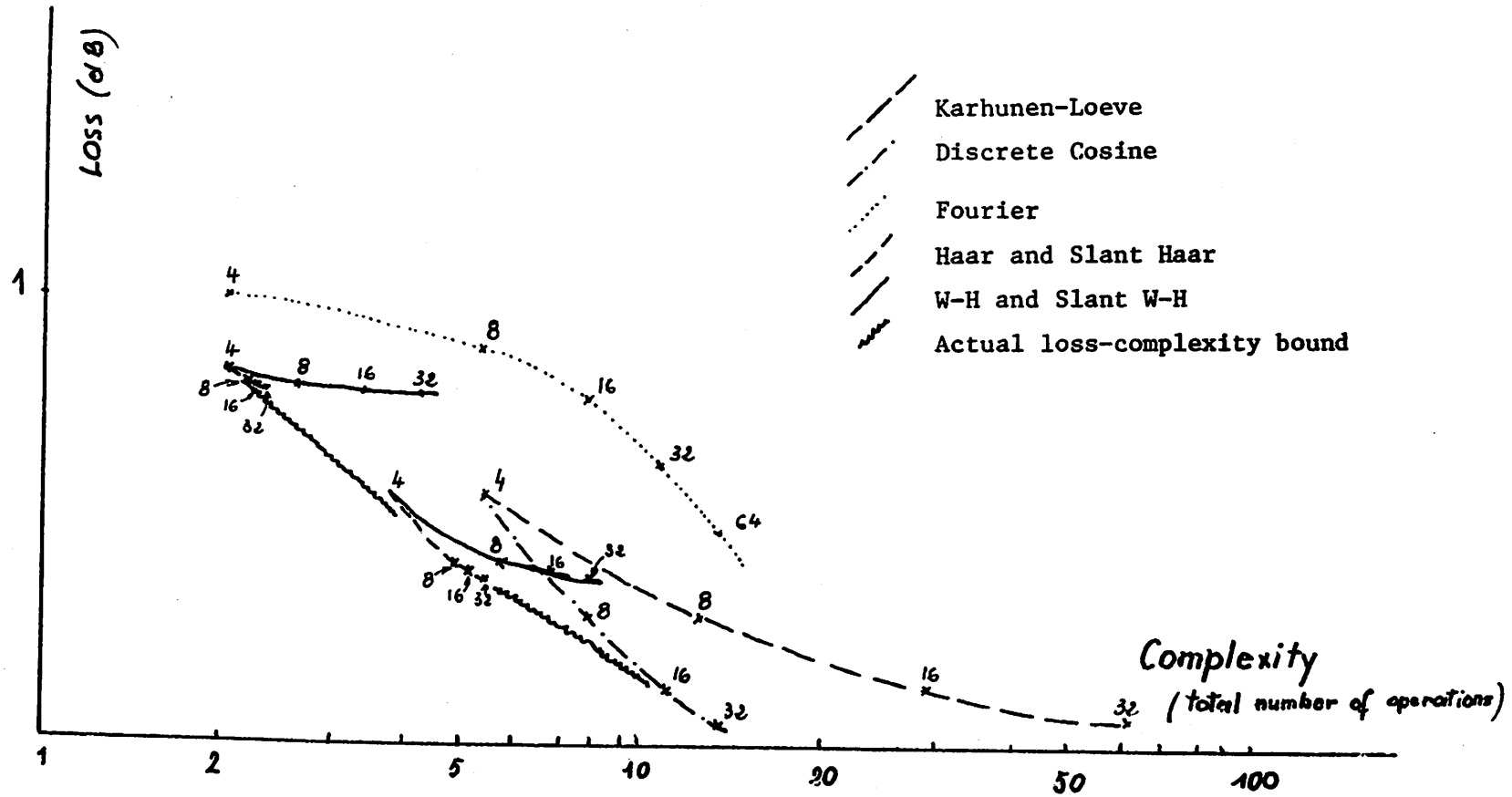


Fig. 7-7. Filtering od a first order Markov process (= 0.05, S/N = 50)

- compute the computational complexity of each transform : in the considered example all transforms perform very closely and their main difference comes from the computational complexity.

- compute the covariance matrix of the transform coefficients with a fast procedure.

C H A P T E R VIII

CONCLUSIONS

This chapter concludes the dissertation with a review of the results obtained and presents various suggestions for further research.

8-1. Results :

In most of this dissertation, our approach has been theoratical. However, we have obtained some specific results of current practical interest, principally for the FFT algorithm and we believe that our formal results will have practical importance. In this section, we review the results obtained in the perspective of their applications.

The formal relations between the Haar and Walsh-Hadamard transforms, their consequences for the fast algorithms and for the identical "zonal" repartition of the energy, show that the Haar transform may be prefered to the W-H transform in some of its applications.

The main theoretical result of this thesis is our unified treatment of fast unitary transforms which has stressed their recursive structure. It allowed us to organize all known fast unitary transforms into a common framework which clarifies many of their properties. It lead us to define large families of fast unitary transforms and in particular the IC transforms. This large family includes most transforms and we believe that additional transforms of practical interest can be found in that family. The recursive definition of fast unitary transforms has provided us with a systematic way to estimate the computational complexity of each transform by the exact computation of the number of elementary

operations required by the fast algorithms.

The unified treatment of fast unitary transforms is also the basis of the error analysis of the fast algorithms. Our derivation, for both fixed-point and floating-point arithmetic, is valid for most transforms. We note that we were able to derive simply and with more accuracy previous results of practical interest concerning the FFT. We even obtained new theoretical results for the FFT in the case of number representation with truncation for both fixed-point and floating-point computations and these results are in surprisingly good agreement with earlier experimental results.

Our study of fast unitary transforms with a given set of basis vectors leads to the definition of fast unitary transforms with constraints which are of importance for a specific application. We defined, in particular, the family of fast generalized Slant transforms which, considering the results obtained elsewhere for one of its members, the Slant transform, should be useful in signal encoding. We have great hopes especially in the slant Haar transform, the fastest and simplest of the family.

In our rapid description of the main applications of the fast unitary transforms in signal processing (representation and dimensionality reduction, encoding and filtering), we showed the important role of the transform coefficients covariance matrix and the trade-off between performance, approximation and complexity. We derived a fast computation procedure of this covariance matrix and, for an important example, we compared several fast unitary transforms. These comparisons showed the interest of the Haar and Slant Haar transforms

and also illustrated the importance of the computational complexity in the application of unitary transforms.

8-2. Further research :

This thesis brings a unifying and clarifying light upon the generation of fast unitary transforms. We have shown the power of this approach in several directions and in particular in the analysis of their algorithms. However, we think that our work could be used in many diversified areas ; in particular, we hope to relate these ideas to data manipulation problems[1] and other areas in future work.

But, restricting our suggestions to the subjects dicussed in this dissertation, we believe that many fast unitary transforms of interest should be found within the scope of the generative rules we have defined. Moreover, for all applications we have considered, it is desirable to compute the optimal transform given some constraints (determining the trade-off between performance, approximation and complexity) : this design problem would certainly open many applications to the fast unitary transforms and should become the central issue in the use of fast unitary transforms in engineering applications.

---

[1] W. Dere, Data organization in linear memory , Ph. D. dissertation, EECS Dept. of the University of California at Berkeley, Aug. 1973.

R E F E R E N C E S

The references are listed in order of chapters in which they appear.

References for chapter 1 :

1. J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Computation of Complex Fourier Series," Math. Comput., Vol. 19, pp. 297-301, Apr. 1965.
2. Proceedings, "Applications of Walsh Functions," National Technical Information Service, U. S. Department of Commerce, Springfield, Virginia 22151, 1970: AD 707 431, 1971: AD 727 000, 1972: AD 744 650, 1973: AD
3. L. R. Rabiner and C. M. Rader, Digital Signal Processing,IEEE Press,New York, 1972.
4. H. F. Harmuth,Transmission of Information by Orthogonal Functions, New York: Springer, Second edition, 1972.
5. H. C. Andrews, Computer techniques in Image Processing, New York: Academic Press, 1970.
6. H. C. Andrews, An Introduction to Mathematical Techniques in Pattern Recognition, New York: Wiley, 1972.
7. J. Hadamard, " Resolution d'une question relative aux determinants," Bull. Sc. Math, serie 2, 17-1, 1893.
8. J.L. Walsh, " A Closed Set of Orthonormal Functions," Amer. J. Math, Vol. 55, pp. 5-24,1923.
9. R.E. A. C. Paley, "On Orthogonal matrices," J. Math. Phys., Vol 12, pp. 311-320, 1933.
10. J. E. Whelchel,Jr. and D. F. Guinn, "The Fast Fourier-Hadamard transform and its use in signal representation and classification," EASCON Rec.,pp.561-573, Sept. 1968.

References for chapter 1 (continued)

11. B. Fino, " Etude experimentale du codage d'images par les transformations de Haar et de Hadamard complexe," Ann. Telecom., tome 27, Nos, 5-6,pp. 185-208, May-June 1972.
12. I. J. Good,"The Interaction Algorithm and Practical Fourier Analysis, J. Roy. Statistical. Soc., Vol. B 20, pp. 361-372, 1958 and addendum Vol. B 22, pp. 372-375, 1960.
13. W. M. Gentleman and G. Sande, " Fast Fourier Transform - for Fun and Profit," AFIPS, 1966. Fall Joint Comp. Conf., pp. 563-578.
14. G. D. Bergland, "A Fast Fourier Transform Algorithm Using Base 8 Iterations," Math. Comp., Vol. 22, pp. 275-279, Apr. 1968.
15. R. C. Singleton, "An Algorithm for Computing the Mixed-Radix Fast Fourier Transform," IEEE Trans. on Audio and Electroac., Vol. AU-17, No. 2, pp. 93-103, June 1969.
16. F. Theilheimer, "A Matrix Version of the Fast Fourier Transform," IEEE Trans. Audio Electroacoust., Vol. AU-17, pp. 158-161, June 1969.
17. D. K. Kahaner, "Matrix description of the Fast Fourier Transform," IEEE Trans. Audio Electroacoust., Vol. AU-18, pp. 442-450, Dec. 1970.
18. J.A. Glassman," A Generalization of the Fast Fourier Transform," IEEE Trans. Comput., Vol. C-19, pp. 105-116, Feb. 1970.
19. W. T. Cochran and al.,"What is the Fast Fourier Transform?," Proc. IEEE. Vol. 55, pp. 1664-1674, Oct. 1967.
20. IEEE Trans. Audio Electroacoust., special issue on Fast Fourier Transform, Vol. AU-17, June 1969.
21. IEEE Trans. Audio Electroacoust., special issue on Fast Fourier Transform, Vol. AU-15, June 1967.
22. N. J. Fine, "On the Walsh functions," Trans. Amer. Math. Soc., Vol. 65, pp. 372-414, 1949.
23. F. Pichler, "Walsh Functions and Linear System Theory," Proc. Applic. of Walsh functions, 1970 Symposium.
24. N. Ahmed and al., "BIFORE or Hadamard Transform," IEEE Trans. Audio Electroacoust., Vol. AU-19, pp. 225-234, Sept. 1971.
25. C.-K. Yuen, "Remarks on the rdering of Walsh Functions,"IEEE Trans. Comput., Vol. C-21, pp. 1452, Dec. 1972.

References for chapter 1 (continued)

26. N. Ahmed and K. R. Rao, "Discrete Fourier and Hadamard Transforms," Elect. Lett., Vol. 6, No. 7, pp. 221-224, 2nd Apr. 1970.

27. A. Haar, "Zur Theorie der Orthogonalen Funktionen-Systeme," Math. Ann.,Vol. 69, pp. 331-371, 1910.

28. S. Kaczmarz, Theorie der Orthogonalenreihen, New York : Chelsea Publ. Co., 1951.

29. G. Alexits, Convergence Problems of Orthogonal Series. London : Pergamon Press, 1961.

30. H. Enomoto and K. Shibata, "Orthogonal Transform Coding System for Television Signals," Proc. Applic. Walsh Functions 1971 Symposium, pp. 11-17.

31. M. Hatori and al., " On the band compression of television signals by the E-sequence transformation technique," 1970 IECEJ Papers Tech. Group Inform. Theory, IT70-13(1970-13).

32. W. K. Pratt, L. R. Welch and W. H. Chen, "Slant Transform for Image Coding," Proc. Applic. Walsh Functions 1972 Symp, pp. 229-234.

33. H. E. Chrestenson, "A class of Generalized Walsh Functions;" Pacific J. Math., Vol. 5, pp. 17-31, 1955.

34. C. Watari, "A Generalization of Haar Functions," Tohoku Math. J., Vol. 8,pp. 286-290, 1956.

35. H. C. Andrews and K.L. Caspari, " A Generalized Technique for Spectral Analysis," IEEE Trans. Comput., Vol. C-19, pp. 16-25, Jan. 1970.

36. H. C. Andrews and K. L. Caspari, " Degrees of Freedom and Modular Structure in Matrix Multiplication," IEEE Trans. Comput., Vol. C-20, No. 2, pp. 133-141, Feb. 1971.

References for chapter 1 (continued)

37. J. E. Gibbs, "Discrete Complex Walsh Functions," Proc. Applic. Walsh Functions 1970 Symp.,

38. F. R. Ohnsorg, " Application of Walsh Functions to Complex Signals," Proc. Applic. Walsh Functions 1970 Symp.,

39. N. Ahmed and K. R. Rao, " Complex BIFORE Transform," Electron. Letts., Vol. 6, pp. 256-258, Apr. 16 1970.

40. N. Ahmed and K. R. Rao, " Additional Properties of Complex BIFORE Transform," IEEE Trans. Audio Electroacoust., Vol.AU-19 , pp. 252-253, Sept. 1971.

41. K. R. Rao and N.Ahmed, " Modified Complex BIFORE Transform," Proc. IEEE, Vol. , pp. 1010-1012, Aug. 1972.

42. N. Ahmed and al., "Discrete Cosine Transform," to be published.

43. N. Ahmed, K. R. Rao and R. B. Schultz, " A Generalized Discrete Transform," Proc. IEEE, Vol. 59, pp. 1360-1362, Sept. 1971.

44. K. R. Rao, N. Ahmed and R. B. Schultz, " A Class of Discrete Orthogonal Functions," to be published.

45. H. C. Andrews and J. Kane, "Kronecker Matrices Computer Implementation and Generalized Spectra," J. Ass Comput. Mach., Vol. 17, pp. 260-268, Apr. 1970.

References for chapter 2 :

1. C.-K. Yuen, "Remarks on the Ordering of the Walsh Functions," IEEE Trans. on Comput.,Vol. C-21, pp. 1452, Dec. 1972.

2. A. Haar, "Zur Theorie der Orthogonalen Funktionen-Systeme," Math. Ann., Vol. 69, pp. 331-371, 1910.

3. R. Bellman, Introduction to Matrix Analysis, New York : McGraw Hill, 1960, p. 227.

4. J. L. Walsh, " A Closed set of orthonormal Functions," Amer. J. Math., Vol. 55, pp. 5-24, 1923.

5. H. F. Harmuth, Transmission of Information by Orthogonal Functions, New York : Springer, Second Edition, 1972.

6. G. Alexits, Convergence problems of Orthogonal Series, New York Pergamon, 1961, pp.46-62.

7. B. J. Fino, "Etude experimentale du codage d'images par les transformation de Haar et de Hadamard complexe," Ann. Telecomm., tome 27, pp.185-208, May-June 1972.

8. H. C. Andrews,Computer Techniques in Image Processing.New York : Academic Press, 1970, pp. 73-70.

9. J. E. Whelchel, Jr and D.F. Guinn,"The Fast Fourier-Hadamard transform and its use in signal representation and calssification," EASCON Rec., pp. 561-573, Sept. 1968.

10. W. K. Pratt and al., "Hadamard Transform Image Coding," Proc. IEEE, Vol. 57, pp. 58-68, Jan. 1969.

11. J.L. Shanks, "Computation of the Fast Walsh-Fourier Transform," IEEE Trans. Comput.,Vol. C-18, pp. 457-459, May 1969.

References for chapter 3 :

1. I. J. Good, "The Interaction Algorithm and Practical Fourier Analysis," J. Roy. Statistical Soc., Vol. B 20, pp. 361 - 372, 1958, and addendum Vol. B 22, pp. 372 - 375, 1960.

2. J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Computation of Complex Fourier Series," Math. Comput., Vol. 19, pp. 297 - 301, Apr. 1965.

3. W. M. Gentleman and G. Sande, "Fast Fourier Transform - for Fun and Profit," AFIPS, 1966 Fall Joint Com. Conf., pp. 563 - 578.

4. J. E. Whetchel and D. F. Guinn, "The Fast Fourier-Hadamard Transform and its use in Signal Representation and Classification," Eascon '68 Rec., pp. 561 - 573.

5. Proceedings, "Application of Walsh Functions," National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22151, 1970: AD-707 431, 1971: AD-727 000, 1972: AD-744 650.

6. B. J. Fino, "Etude experimentale du codage d'images par les transformations de Haar et Hadamard complexe," Ann. Telecomm.,tome 27, pp. 185-208, May-June 1972.

7. J. E. Shore, "On the Application of Haar Functions," IEEE Trans. on Comm., Vol. COM-21, No. 3, pp. 209-216, March 1973.

8. J. Duan and P. A. Wintz, "Error Free Coding," to be published.

9. H. C. Andrews, An Introduction to Mathematical Techniques in Pattern Recognition, New York: Wiley, 1972.

10. S. J. Campanella and G. S. Robinson, "A Comparison of Orthogonal Transformations for Digital Speech Processing," IEEE Trans. on Comm., Vol. COM-19, No. 6, pp. 1045-1050, Dec. 1971.

References for chapter 3(continued)

11. H. C. Andrews, Computer Techniques in Image Processing, New York: Academic Press, 1970.

12. T. S. Huang, W. F. Schreiber and O. J. Tretiak, "Image Processing," Proc. IEEE, Vol. 59, pp. 1586-1609, Nov. 1971.

13. P. A. Wintz, "Transform Picture Coding," Proc. IEEE, Vol. 60 No. 7, pp. 809-820, July 1972.

14. H. C. Andrews and J. Kane, "Kronecker Matrices Computer Implementation and Generalized Spectra," J. Ass. Comput. Mach., Vol. 17, pp. 260-268, Apr. 1970.

15. H. C. Andrews and K. L. Caspari, "A Generalized Technique for Spectral Analysis," IEEE Trans. Comput. Vol. C-19, pp. 16-25, Jan. 1970.

16. H. C. Andrews and K. L. Caspari, "Degrees of Freedom and Modular Structure in Matrix Multiplication," IEEE Trans. on Computers, Vol. C-20, No. 2, pp. 133-141, Feb. 1971.

17. N. Ahmed, K. R. Rao and R. B. Schultz, "A Generalized Discrete Transform," Proc. IEEE, Vol. 59, No. 9 , pp. 1360-1362, Sept. 1971.

18. K. R. Rao, N. Ahmed and R. B. Schultz, "A Class of Discrete Orthogonal Transforms," to be published.

19. H. F. Harmuth, Transmission of Information by Orthogonal Functions, New York: Springer, Second Edition, 1972, pp. 30-36.

20. C-K. Yuen, "Remarks on the Ordering of Walsh Functions," IEEE Trans. Comput., Corresp., Vol. C-21, No. 12, pp. 1452, Dec. 1972.

21. P. Y. Schwartz, J. Poncin and B. Fino, "Statistical Properties of Orthogonal Transforms," Proc. Conf. on Digital Processing of Signals in Communications, Vol. 23, pp. 151-174, London, Apr. 1972.

References for chapter 3 (continued):

22. J. A. Glassman, "A Generalization of the Fast Fourier Transform," IEEE Trans. on Computers, Vol. C-19, No. 2, pp. 105-113, Feb. 1970.

23. R. Singleton, "An Algorithm for Computing the Mixed Radix Fast Fourier Transform," IEEE Trans. on Audio and Electroacoustic, Vol. AU-17, No. 2, pp. 93-103, June 1969.

24. R.E.A.C. Paley, "On Orthogonal Matrices," J. Math. Phys. Vol. 12, pp. 311-320, 1933.

25. J. L. Walsh, "A Closed Set of Normal Orthonormal Functions," Amer. J. Math., Vol. 55, pp. 5-24, 1923.

26. J. W. Manz, "A Sequency-Ordered Fast Walsh Transform," IEEE Trans. on Audio and Electroacoustics, Vol. AU-20, No. 3, pp. 204-205, Aug. 72.

27. H. E. Chrestenson, "A Class of Generalized Walsh Functions," Pacific J. Math., Vol. 5, pp. 17-31, 1955.

28. C. Watari, "A Generalization of Haar Functions," Tohoku Math. J., Vol. 8, pp. 286-290, 1956.

29. H. Enomoto and K. Shibata, "Orthogonal Transform Coding System for Television Signals," Proc. 1971 Symp. on Appl. of the Walsh Functions, pp. 11-17. AD-727 000.

30. W. K. Pratt, L. R. Welch and W. H. Chen, "Slant Transform for Image Coding," Proc. 1972 Symp. on Appl. of the Walsh Functions, pp. 229-234. AD-744 650.

31. W. K. Pratt, "Walsh Functions in Image Processing and Two Dimensional Filtering," Proc. 1972 Symp. on Appl. of the Walsh Functions, pp. 14-22. AD-744 650.

32. N. Ahmed and K. R. Rao, "Complex Bifore Transform," Electron. Lett., Vol. 6, No. 8, pp. 256-258, 16th Apr. 1970.

References for chapter 3 (continued) :

33. F. R. Ohnsorg, "Application of Walsh Functions to Complex Signals," Proc. 1970 Symp. on Applications of the Walsh Functions, pp. 123-127. AD-707 431.

34. K. R. Rao and N. Ahmed, "Modified Complex BIFORE Transform," Proc. IEEE, Vol. 60, No. 8, pp. 1010-1012, Aug. 1972.

References for chapter 4 :

1. P. D. Welch, "A Fixed-Point Fast Fourier Transform Error Analysis," IEEE Trans. Audio and Electroacoust., Vol AU-17, pp. 151-157, June 1969.

2. A. V. Oppenheim and C. J. Weinstein, "Effect of Finite Register Length in Digital Filtering and the Fast Fourier Transform," Proc. IEEE, Vol. 60, pp. 957-976, Aug. 1972.

3. C. L. Weinstein, "Quantization Effects in Digital Filters," M. I. T. Lincoln Lab. Tech. Rep. 468, ASTIA Doc. DDC AD-706862, Nov. 21, 1969.

4. J. H. Wilkinson, Rounding Errors in Algebraic Processes. Englewood Cliffs, N. J. : Prentice Hall, 1963.

References for chapter 5 :

1. T. Kaneko and B. Liu, " Accumulation of Roundoff Errors in Fast Fourier Transforms," J. ACM, Vol. 17,ppp. 637-654, Oct. 1970.

2. C. J. Weinstein, "Roundoff Noise in Floating Point Fast Fourier Transform," IEEE Trans. Audio Electroacoust., Vol. AU-17, pp. 209-215, Sept. 1969.

3. C. J. Weinstein, "Quantization Effects in Digital Filters," M. I. T. Lincoln Lab. Tech. Rep. 468, ASTIA Doc DDC AD-706862, Nov. 21, 1969.

4. A. V. Oppenheim and C.J. Weinstein, "Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform," Proc. IEEE, Vol. 60,pp. 957-976, Aug. 1972.

5. O. W. C. Chan and E. I. Jury, "Round-off Error in Multi-Dimensional Generalized Discrete Transforms, to appear in IEEE Trans. Circuit Th., Vol. CT-21, Jan. 1974.

6. W. M. Gentleman and G. Sande, "Fast Fourier Transforms - for Fun and Profit," Proc. AFIPS Fall joint Computer Conf., pp. 563-577, 1966.

7. G. U. Ramos, "Roundoff Error Analysis of the Fast Fourier Transform," Math. Computation, Vol. 25, No. 116, pp. 757-768, Oct. 1971.

8. J. H. Wilkinson, Rounding Errors in Algebraic Processes. Englewood Cliffs, N. J. : Prentice Hall, 1963.

9. J. M. Yohe, " Roundings in Floating-Point Arithmetic," IEEE Trans. Comput., Vol. C-22 , pp.577-586, June 1973.

10. W. J. Cody,Jr., "Static and Dynamic Numerical Characteristics of Floating-Point Arithmetic," IEEE Trans. Comput., Vol. C-22, pp. 598-600, June 1973.

References for Chapter 5 (continued):

11. B. Liu and T. Kaneko, " Error Analysis of Digital Filters Realized with Floating-Point Arithmetic," Proc. IEEE, Vol. 57, pp. 1735-1747, Oct. 1969.

12. M. Tasto and P. A. Wintz, "Note on Error Signal of Block Quantizers," IEEE Trans. Communic., Vol. COM-21, pp. 216-219, March 1973.

13. V. R. Algazi, "Adaptative Line-by-Line Encoder," Proc. I. T. C. Conf., Los Angeles, pp. 363-371, 1972.

14. B. J. Fino, "Etude experimentale du codage d'images par les transformations de Haar et Hadamard complexe," Ann. Telecom. tome 27, pp. 185-208, May-June 1972.

References for chapter 7 :

1. S. Watanabe, Karhunen-Loeve Expansion and Factor analysis- Theoretical Remarks and applications", Proc 4th Prague Conf. on Inform. Th., 1965.

2. V. R. Algazi and D. J. Sakrison,"On the Optimality of the Karhunen-Loeve Expansion", IEEE Trans. Inform. Th., Vol. IT-15, pp. 319-321, March 1969.

3. P. A. Wintz, "Transform Image Coding," Proc. IEEE, Vol. 60, pp. 809-820, July 1972.

4. H. C. Andrews, Computer Techniques in Image Processing, New York : Academic Press. 1970.

5. M. D. Levine, "Feature Extraction : a Survey," Proc. IEEE, Vol. 57, pp. 1391-1407,

6. K. S. Fu, Statistical Pattern Recognition, in Adaptative, Learning and Pattern Recognition Problems : Theory and Applications New York : Academic Press. 1970.

7. IEEE Trans. Comput. , Special issue on Feature extraction and selection, Vol. C-20, pp. 965-1137, Sept. 1971.

8. K. Fukunaga and W. L. Koontz, "Application of the Karhunen-Loeve Expansion to feature selection and ordering," IEEE Trans. Comput., Vol. C-19, pp. 311-318, Apr. 1970.

9. H. C. Andrews, "Multidimensional Rotations in Feature Selection," IEEE Trans. Comput., Vol. C-20, pp. 1045-1051, Sept. 1971.

10. J. W. Carl and C. F. Hall, " The Application of Filtered Transforms to the General Classification Problem" , IEEE Trans. Comput., Vol. C-21, pp. 785-790, July 1972.

References for chapter 7 (continued) :

11. S. J. Campanella and G. S. Robinson, "A comparison of Orthogonal Transformations for Digital Speech Processing", IEEE Trans. Communic., Vol. COM- 19, pp. 1045-1050, Dec. 1971.

12. A. Habibi and P. A. Wintz, "Image Coding by Linear Transformations and Block Quantization", IEEE Trans. Com. Tech., Vol. COM-19, pp. 50-61, Feb. 1971.

13. D. J. Sakrison and V. R. Algazi, "Comparison of Line-by-Line and Two-Dimensional Encoding of Random Images", IEEE Trans. Inf. Th., Vol. IT-17, pp. 386-398, July 1971.

14. T. S. Huang and al., "Image Processing," Proc. IEEE, Vol. 59, pp. 1586-1609, Nov. 1971.

15. J. Mannos, A Class of Fidelity Criteria for the Encoding of Visual Images, Ph. D. Thesis, Electr. Eng. , Univ. of Calif., 1972.

16. R. G. Gallager, Information Theory and Reliable Communication, New York : Wiley. 1969. Chap. 9.

17. W. K. Pratt, "Generalized Wiener Filtering Computation Techniques," IEEE Trans. Comput., Vol. C-21, pp. 636-641, July 1972.

18. J. Pearl, "Walsh Processing of of Random Signals", Symposium on Appl. Walsh Functions 1970.

19. N. Ahmed and al.,"Discrete Cosine Transform", to be published.

20. T. Goblick and J. Holsinger,"Analog Source digitalization : a comparison of Theory and Practice,"IEEE Tran. Inf. Th., Vol. IT-13, pp. 323-326, Apr. 1967.

21. J. Pearl,"Basis-Restricted Transformations and Performance Measures for Spectral Representations", IEEE Trans. Inf. Th. , Vol. IT-17, pp.751-752, Nov. 1971.

References for chapter 7 (continued) :

22. V. R. Algazi and B. J. Fino, "Comment on Basis-Restricted Transformations and Performance Measures for Spectral Representations", IEEE Trans. Inf. Th., Vol. IT-19, pp. 564-565, July 1973.

23. P-Y. Schwartz, J. Poncin and B. Fino, "Statistical Properties of orthogonal Transforms", Proc. Conf. on Digital Signal Processing in Communicat., Vol. 23, pp. 151-174, April 1972.