THE DISCRETE HARTLEY TRANSFORM

by

CHANDRA CHUDA VARANASI

B.Tech., Jawaharlal Nehru Technological University, India

1983

A THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

Kansas State University
Manhattan, Kansas

1988

Approved by:

[illegible]

Major Professor

A11208 130818

ACKNOWLEDGMENTS

I would like to thank Dr. Stephen A. Dyer for his encouragement and guidance throughout my M.S program. Thanks are also extended to Dr. Brian K. Harms and Dr. John E. Boyer for serving on my committee.

TABLE OF CONTENTS

TABLE OF CONTENTS (Contd.)

TABLE OF CONTENTS (Contd.)

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 Background

A signal, represented mathematically as a function of one or more independent variables, conveys information about the state or behavior of a physical system. For example, speech is represented as a function of time. Signals are said to be continuous-time (CT) signals if they are defined over a continuum of times, and discrete-time (DT) signals if they are defined only at a discrete set of values of the independent variable. In addition, if the amplitude of a CT signal is also continuous, then such a signal is called an analog signal. Similarly, if the amplitude of a DT signal is also discrete, such a signal is called a digital signal. Therefore, digital signals are represented as sequences of numbers.

To extract any meaningful information from a signal, it has to be processed. To that end, development of techniques to process the signals assumes great importance. Usually, these techniques involve transforming a signal to some other advantageous signal that facilitates its analysis. These transformation techniques are useful when we want to separate two or more signals, or when a particular component of the signal has to be enhanced in relation to the others, or when a desired parameter of the signal has to be estimated. When a transformation technique operates on an analog signal and produces another analog signal as its output, it is called analog signal processing. If the transform operates on a digital signal to produce another digital signal as its output, it is called digital signal processing (DSP).

DSP is employed in a variety of fields of science and technology such as biomedical engineering, acoustics, sonar, radar, seismology, speech communication, data communication, nuclear science, spectroscopy and many others. As an example, when a signal is transmitted over a communication channel, it is corrupted in a number of ways, by, e.g., channel distortion, fading, and the insertion of background noise. One of the major functions at the receiving end is to compensate for all the affecting disturbances. DSP techniques play an important role in such an application.

DSP systems are especially attractive because they can be realized with flexibility using digital computer, or they can be realized in hardware using digital components. They can be used to simulate analog systems, and most importantly, can sometimes be used to realize otherwise impossible analog signal transformations.

The Fourier transform (FT) provides a very useful technique for use in CT signal processing. Similarly, the discrete Fourier transform (DFT) plays a vital role in DSP. Spectrum analysis, an important component of DSP, is one area where the DFT finds extensive application. But, for quite a long time, no efficient means of implementing it, either in hardware or in software, was known. However, the disclosure in 1965 by Cooley and Tukey [1] of an efficient algorithm to compute the DFT revolutionized signal processing. The class of algorithms proposed by them has come to be known as the fast Fourier transform or the FFT.

The FFT algorithm reduced the computation time of the DFT phenomenally, facilitating the commercial availability of special-purpose signal processing chips which can operate at extremely high data rates.

Despite its tremendous application, the DFT has an unattractive feature in that it transforms a real-valued sequence also into a complex-valued sequence.

In 1983, R. N. Bracewell [2] proposed an inherently real-valued transform called the Hartley transform (HT). The new transform has the advantage that a real-valued signal always generates a real-valued transform signal. Secondly, unlike the FT, the HT is symmetric that is, both the forward and inverse transforms are identical. Thirdly, the HT is so closely related to the FT, that one can move from the Fourier domain to the Hartley domain and vice versa in a straightforward manner. Bracewell also introduced the discrete Hartley transform (DHT), which has all the above mentioned advantages over the DFT. But, the most important practical advantage of the DHT over the DFT is that it is computationally faster than the DFT.

The HT has an interesting history. While working on problems relating to transmission lines, in 1942, Hartley first proposed it under the name symmetrical Fourier identity [3]. However, it remained relatively obscure for quite a long time until Bracewell revived it in 1983. Interestingly, Zhong De Wang, working independently, proposed an identical transform in 1981 and developed many of its mathematical properties. He called it the W-Transform [4 - 6].

1.2 Implementation of the Present Work

All the properties of the HT and the DHT are developed, analogous to the properties of the FT and the DFT, respectively. Decomposition formulas for all the fast Hartley transform (FHT) algorithms are derived and the computational cost of different algorithms are compared. Studies in the enhancement of signal-to-noise ratio (SNR) are carried out on simulated Raman spectra using the matched filter technique. The matched filter is implemented using both the DFT and the DHT. The computational efficiency and the filter response in both the methods are compared.

1.3 Structure of the Thesis

In Chapter 2, we introduce the HT and develop its properties, and in Chapter 3 we define the DHT and develop its properties. Chapters 4 and 5 present a detailed development of the radix-2 DIT and DIF algorithms, respectively. We discuss the radix-4 DIT algorithm in Chapter 6, and the split-radix algorithm in Chapter 7. In Chapter 8, we discuss an application of the DHT in the analysis of Raman spectra, present the results of the simulation, and compare its performance with the DFT in that application. Finally, Chapter 9 gives a summary of the conclusions drawn.

2. THE HARTLEY TRANSFORM

2.1 Introduction

Since the HT is so closely related to the FT and many of its properties follow directly from those of the FT, we begin the chapter by defining the FT of an aperiodic signal. Then we define the HT as proposed by Bracewell and develop various properties of the transform.

2.2 The Fourier Transform

Given a signal x(t), its Fourier transform X(f) is defined as

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}\,dt. \tag{2.1}$$

To recover x(t) from X(f), we perform the inverse Fourier transform (IFT) on X(f). The IFT is defined as

$$x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}\,df. \tag{2.2}$$

Since

$$e^{-j2\pi ft} = \cos(2\pi ft) - j\sin(2\pi ft),$$

we can rearrange Eqn. (2.1) as

$$X(f) = \int_{-\infty}^{\infty} x(t)\cos(2\pi ft)dt - j\int_{-\infty}^{\infty} x(t)\sin(2\pi ft)dt. \tag{2.3}$$

As evident from (2.3), in general, the FT X(f) of a real signal x(t) is complex, with the real part

$$\mathrm{Re}\ \{X(f)\} = \int_{-\infty}^{\infty} x(t)\cos(2\pi ft)dt, \tag{2.4}$$

and the imaginary part

$$\mathrm{Im}\ \{X(f)\} = - \int_{-\infty}^{\infty} x(t)\sin(2\pi ft)dt. \tag{2.5}$$

2.3 The Hartley Transform

The HT H(f) of a real signal x(t) is defined as

$$H(f) = \int_{-\infty}^{\infty} x(t)\mathrm{cas}(2\pi ft)dt \tag{2.6}$$

where

$$\mathrm{cas}(2\pi ft) = \cos(2\pi ft) + \sin(2\pi ft). \tag{2.7}$$

The HT is symmetric that is, if we need to recover x(t) from H(f), we perform the transformation in (2.6) on H(f). Thus both the forward and inverse transforms are identical.

Using (2.7) we can rewrite (2.6) as

$$H(f) = \int_{-\infty}^{\infty} x(t)\cos(2\pi ft)dt + \int_{-\infty}^{\infty} x(t)\sin(2\pi ft)dt. \tag{2.8}$$

From (2.8) we observe that, unlike the FT, the HT of a signal x(t) is real.

Using the trigonometric identities

$$\cos(- x) = \cos(x)$$

and

$$\sin(- x) = - \sin(x)$$

we can compute H(-f) as

$$H(-f) = \int_{-\infty}^{\infty} x(t)\cos(2\pi ft)dt - \int_{-\infty}^{\infty} x(t)\sin(2\pi ft)dt. \tag{2.9}$$

Now, by splitting H(f) into even and odd parts $H_e(f)$ and $H_o(f)$ respectively, we get

$$H_e(f) = [H(f) + H(-f)] / 2, \tag{2.10}$$

and

$$H_o(f) = [H(f) - H(-f)] / 2. \tag{2.11}$$

Substituting the integral expressions for H(f) and H(-f) that we obtained in (2.8) and (2.9) in (2.10) and (2.11), we get

$$H_e(f) = \int_{-\infty}^{\infty} x(t)\cos(2\pi ft)dt, \tag{2.12}$$

and

$$H_o(f) = \int_{-\infty}^{\infty} x(t)\sin(2\pi ft)dt \tag{2.13}$$

2.4 Properties

The HT has many interesting properties that can be successfully employed in many signal processing applications. In this section we will derive some of its important properties.

2.4.1 Relation Between the FT and the HT

From (2.4) and (2.12), we observe that

$$\text{Re}\ \{X(f)\} = H_e(f),$$

and

from (2.5) and (2.13), it is clear that

$$\text{Im}\ \{X(f)\} = -\ H_o(f).$$

Therefore, once we have the HT of a signal x(t), we can compute its FT as shown below.

$$X(f) = H_e(f) - j\ H_o(f). \tag{2.14}$$

By referring to (2.4), (2.5), (2.12) and (2.13) again, we see that the HT of a signal x(t) can be computed from its FT X(f) as follows:

$$H(f) = \text{Re}\ \{X(f)\} - \text{Im}\ \{X(f)\}. \tag{2.15}$$

2.4.2 Linearity

The HT is a linear transform.

Let us find the transform of the signal $\{x_1(t) + x_2(t)\}$. Let the HT of $x_1(t)$ be $H_1(f)$ and that of $x_2(t)$ be $H_2(f)$. Let the HT of $\{x_1(t) + x_2(t)\}$ be H(f). Then H(f) is given by

$$H(f) = \int_{-\infty}^{\infty} [x_1(t) + x_2(t)]\text{cas}(2\pi ft)dt.$$

The above integral can be split into two integrals as shown below.

$$H(f) = \int_{-\infty}^{\infty} x_1(t)\text{cas}(2\pi ft)dt + \int_{-\infty}^{\infty} x_2(t)\text{cas}(2\pi ft)dt$$

$$= H_1(f) + H_2(f).$$

Therefore, the HT of a sum of signals equals the sum of the HTs of individual signals.

2.4.3 Reversal

If the HT of x(t) is H(f) then the HT of x(-t) is H(-f).

The HT of x(-t) is given by

$$HT\ \{x(-t)\} = \int_{-\infty}^{\infty} x(-t)\mathrm{cas}(2\pi ft)dt.$$

Now, let -t = u. Then dt = -du. Making these substitutions in the above integral, we get

$$HT\ \{x(-t)\} = \int_{-\infty}^{\infty} x(u)\mathrm{cas}\{2\pi f(-u)\}du$$

$$= \int_{-\infty}^{\infty} x(u)\mathrm{cas}\{2\pi u(-f)\}du$$

$$= H(-f).$$

Therefore, if the HT of x(t) is H(f), the HT of x(-t) is H(-f).

2.4.4 Transforms of Even and Odd Functions

Let x(t) be an even function. i.e.,

$$x(t) = x(-t).$$

The HT of x(t) is obtained as

$$H(f) = \int_{-\infty}^{\infty} x(t)\mathrm{cas}(2\pi ft)dt.$$

Since x(t) is an even function, replacing x(t) by x(-t) in the above integral, we get

$$H(f) = \int_{-\infty}^{\infty} x(-t)\text{cas}(2\pi ft)dt$$

By the reversal theorem, the right hand side of the above equation is H(-f). Therefore,

$$H(f) = H(-f).$$

Therefore, the HT of an even function is also even.

Now, let x(t) be an odd function i.e., x(t) = - x(-t).

If we denote the HT of x(t) by H(f),

$$H(f) = \int_{-\infty}^{\infty} x(t)\text{cas}(2\pi ft)dt.$$

Since x(t) = - x(-t), replacing x(t) by - x(-t) in the above equation, we get

$$H(f) = - \int_{-\infty}^{\infty} x(-t)\text{cas}(2\pi ft)dt.$$

By the reversal theorem, the right hand side of the above equation is - H(-f). Thus, the above equation reduces to

$$H(f) = - H(-f).$$

Therefore, the HT of an odd function is also odd.

2.4.5 The Infinite Integral Theorem

The infinite integral of a signal x(t) can be computed very easily from its HT. This result follows directly from the definition of the HT. We know that

$$H(f) = \int_{-\infty}^{\infty} x(t)\mathrm{cas}(2\pi ft)dt.$$

Substituting f = 0 in the above equation, we get

$$H(0) = \int_{-\infty}^{\infty} x(t)dt, \tag{2.16}$$

since

$$\mathrm{cas}(0) = \cos(0) + \sin(0) = 1.$$

2.4.6 Shift Theorem

Now, let us find the HT of x(t - T), which is the signal x(t) shifted by T units.

We know that the HT of x(t - T) is given by

$$HT\ \{x(t - T)\} = \int_{-\infty}^{\infty} x(t - T)\mathrm{cas}(2\pi ft)dt. \tag{2.17}$$

Let (t - T) = h. Then

$$t = (T + h),$$

and

$$dt = dh.$$

Using the above substitutions, we rearrange (2.17) as

$$HT\{x(t - T)\} = \int_{-\infty}^{\infty} x(h)\text{cas}\{2\pi f(h + T)\}dh. \tag{2.18}$$

Let us expand $\text{cas}\{2\pi f(h + T)\}$.

$$\begin{aligned}\text{cas}\{2\pi f(h + T)\} &= \cos\{2\pi f(h + T)\} + \sin\{2\pi f(h + T)\} \\ &= \cos(2\pi fh)\cos(2\pi fT) - \sin(2\pi fh)\sin(2\pi fT) \\ &+ \sin(2\pi fh)\cos(2\pi fT) + \cos(2\pi fh)\sin(2\pi fT).\end{aligned} \tag{2.19}$$

Collecting terms in the above expansion, we get

$$\begin{aligned}\text{cas}\{2\pi f(h + T)\} &= \cos(2\pi fT)\{\cos(2\pi fh) + \sin(2\pi fh)\} \\ &+ \sin(2\pi fT)\{\cos(2\pi fh) - \sin(2\pi fh)\}.\end{aligned}$$

Recognizing that

$$\cos(2\pi fh) + \sin(2\pi fh) = \text{cas}(2\pi fh),$$

and

$$\begin{aligned}\cos(2\pi fh) - \sin(2\pi fh) &= \cos\{2\pi h(-f)\} + \sin\{2\pi h(-f)\} \\ &= \text{cas}\{2\pi h(-f)\},\end{aligned}$$

we can rewrite (2.19) as

$$\begin{aligned}\text{cas}\{2\pi f(h + T)\} &= \cos(2\pi fT)\text{cas}(2\pi fh) \\ &+ \sin(2\pi fT)\text{cas}\{2\pi h(-f)\}.\end{aligned} \tag{2.20}$$

Substituting the result of (2.20) in (2.18), we get

$$\begin{aligned}HT\{x(t - T)\} &= \int_{-\infty}^{\infty} x(h)[\cos(2\pi fT)\text{cas}(2\pi fh) \\ &+ \sin(2\pi fT)\text{cas}\{2\pi h(-f)\}]dh.\end{aligned}$$

Splitting the above integral into two integrals, we get

$$HT\{x(t - T)\} = \cos(2\pi fT)\int_{-\infty}^{\infty} x(h)\text{cas}(2\pi fh)dh$$

$$+ \sin(2\pi fT) \int_{-\infty}^{\infty} x(h)\operatorname{cas}\{2\pi h(-f)\}dh$$

$$= \cos(2\pi fT)H(f) + \sin(2\pi fT)H(-f). \qquad (2.21)$$

Note that if x(t) is even, and thus H(f) is even, then the above result reduces to

$H(f)[\cos(2\pi fT) + \sin(2\pi fT)] = \operatorname{cas}(2\pi fT)H(f).$

and if x(t) is odd, and thus H(f) is odd, (2.21) becomes

$H(f)[\cos(2\pi fT) - \sin(2\pi fT)] = \operatorname{cas}\{2\pi T(-f)\}H(f).$

2.4.7 Derivative Theorem

From FT theory, we know that if X(f) is the FT of the signal x(t), then the FT of its derivative is given by $j2\pi f\, X(f)$ i.e.,

$$FT\ \{dx(t)/dt\} = j2\pi f\ X(f).$$

From (2.14), we can rewrite $j2\pi f\ X(f)$ as follows:

$$j2\pi f\ X(f) = j2\pi f\ [H_e(f) - jH_o(f)]$$

$$= 2\pi f\ [H_o(f) + jH_e(f)].$$

where $H_e(f)$ and $H_o(f)$ are the even and odd parts of the HT H(f) of the signal x(t).

Since the above expression is the FT of the derivative of x(t), from (2.15), the HT of the derivative is obtained as

$$HT\ \{dx(t)/dt\} = Re\ [FT\ \{dx(t)/dt\}] - Im\ [FT\ \{dx(t)/dt\}]$$

$$= 2\pi fH_o(f) - 2\pi fH_e(f). \qquad (2.22)$$

Using the relations

$$H_o(f) = [H(f) - H(-f)] / 2,$$

and

$$H_e(f) = [H(f) + H(-f)] / 2,$$

we rewrite (2.21) as

$$HT \{dx(t)/dt\} = - 2\pi f\ H(-f).$$

Alternatively, we can use the shift theorem to prove the derivative theorem. We know that the derivative dx(t)/dt is defined as

$$dx(t)/dt = \lim_{h \to 0} [x(t + h) - x(t)] / h. \tag{2.23}$$

If H(f) is the HT of x(t), then by the shift theorem

$$HT \{x(t + h)\} = \cos(2\pi fh)H(f) - \sin(2\pi fh)H(-f).$$

Taking the HT on both sides of (2.23) and applying the above result, we get

$$HT \{dx(t)/dt\} = \lim_{h \to 0} [\cos(2\pi fh)H(f) - \sin(2\pi fh)H(-f) - H(f)] / h.$$

Evaluating the above limit using L'Hospital's rule, we get

$$HT \{dx(t)/dt\} = -2!fH(-f).$$

2.4.8 HT of the nth Derivative

The FT of the nth derivative of a signal x(t), whose FT is X(f), is given by $(j2\pi f)^n X(f)$.

By (2.14),

$$(j2\pi f)^n X(f) = j^n (2\pi f)^n [H_e(f) - jH_o(f)],$$

where H(f) is the HT of x(t).

Noting that

$$j^n = e^{jn\pi/2},$$
$$= \cos(n\pi/2) + j\sin(n\pi/2),$$

we rewrite the above equation as

$$\text{FT}\ \{d^n x(t)/dt^n\} = (2\pi f)^n[\cos(n\pi/2) + j\sin(n\pi/2)]$$
$$[\{H(f) + H(-f)\} \,/\, 2 - j\{H(f) - H(-f)\} \,/\, 2].$$

(2.15) tells us that if we subtract the imaginary part of the above expression from its real part, we get the HT of the nth derivative of x(t). Therefore,

$$\text{HT}\ \{d^n x(t)/dt^n\} = (2\pi f)^n[\cos(n\pi/2)H(f) - \sin(n\pi/2)H(-f)].$$

However, when n is odd, the first term in the above expression vanishes and, when n is even the second term vanishes. Therefore,

$$\text{HT}\ \{d^n x(t)/dt^n\} = -\ (2\pi f)^n \sin(n\pi/2)H(-f) \qquad \text{(n odd)}$$

$$\text{HT}\ \{d^n x(t)/dt^n\} = \ (2\pi f)^n \cos(n\pi/2)H(f). \qquad \text{(n even)}$$

2.4.9 First-Moment Theorem

Given a signal x(t), its first moment is given by

$$\text{F. M.} = \int_{-\infty}^{\infty} tx(t)dt.$$

Now, we know that the HT H(f) of x(t) is given by

$$H(f) = \int_{-\infty}^{\infty} x(t)\text{cas}(2\pi ft)dt.$$

Taking the derivative of the above equation with respect to f yields

$$dH(f)/df = 2\pi \int_{-\infty}^{\infty} x(t)\ t[\cos\ (2\pi ft) - \sin\ (2\pi ft)]\ dt.$$

Evaluating the above expression at f = 0 gives

$$dH(0)/df = 2\pi \int_{-\infty}^{\infty} x(t)tdt.$$

Therefore, the first moment can be found as

$$F.\ M. = \int_{-\infty}^{\infty} tx(t)dt = [dH(0)/df]\ /\ 2\pi.$$

2.4.10 Second-Moment Theorem

Given a signal x(t), its second moment is defined as

$$S.\ M. = \int_{-\infty}^{\infty} t^2 x(t)dt.$$

Differentiating (2.6) two times with respect to f yields

$$H\ (f) = 4\pi^2 \int_{-\infty}^{\infty} t^2\ x(t)\ [-\cos(2\pi ft) - \sin\ (2\pi ft)]\ dt.$$

Evaluating the above expression at f = 0 gives

$$H\ (f) = -\ 4\pi^2 \int_{-\infty}^{\infty} t^2\ x(t)dt$$

Therefore, the second moment is

$$\text{S. M.} = \int_{-\infty}^{\infty} t^2 \, x(t)dt$$

$$= - H \; (0)/4\pi^2 .$$

2.4.11 Centroid

Given a function x(t), its centroid is defined as

$$[\int_{-\infty}^{\infty} tx(t)dt] \; / \int_{-\infty}^{\infty} x(t)dt = H'(0) \; / \; \{2\pi[H(0)]\}.$$

The result follows directly from the infinite integral and first-moment theorems.

2.4.12 Autocorrelation Theorem

From FT theory, we know that the FT of the autocorrelation of x(t) is given by $|X(f)|^2$ where, X(f) is the FT of x(t).

However,

$$|X(f)|^2 = \text{Re} \; [X(f)]^2 + \text{Im} \; [X(f)]^2.$$

From (2.14), it follows that

$$|X(f)|^2 = H_e^2(f) + H_o^2(f).$$

Since

$$H_e(f) = [H(f) + H(-f)] \; / \; 2,$$

and

$$H_o(f) = [H(f) - H(-f)] \; / \; 2,$$

the above equation can be rearranged as

$$|X(f)|^2 = (1/2)[H^2(f) + H^2(-f)].$$

Thus, the HT of the autocorrelation function is a non-negative and even function. Observe that if H(f) is even, the above expression becomes $H^2(f)$, which resembles the familiar FT result.

2.4.13 Convolution Theorem

If the FTs of signals $x_1(t)$ and $x_2(t)$ are $X_1(f)$ and $X_2(f)$, respectively, then we know that the FT of the convolution of the two signals is the product of the individual FTs $X_1(f)$ and $X_2(f)$ i.e.,

$$FT\ \{x_1(t) * x_2(t)\} = X_1(f)X_2(f)$$

By (2.14), we can rewrite $X_1(f)X_2(f)$ as

$$X_1(f)X_2(f) = [H_{1e}(f) - jH_{1o}(f)][H_{2e}(f) - jH_{2o}(f)].$$

where

$H_{1e}(f)$ is the even part of the HT $H_1(f)$ of $x_1(t)$,

$H_{1o}(f)$ is the odd part of $H_1(f)$,

$H_{2e}(f)$ is the even part of the HT $H_2(f)$ of $x_2(t)$,

and

$H_{2o}(f)$ is the odd part of $H_2(f)$.

Now, carrying out the multiplication in the above equation, we get

$$X_1(f)X_2(f) = [H_{1e}(f)H_{2e}(f) - H_{1o}(f)H_{2o}(f)]$$
$$- j[H_{1e}(f)H_{2o}(f) + H_{1o}(f)H_{2e}(f)].$$

From (2.15), the HT of the convolution is given by

$$HT\ \{x_1(t)*x_2(t)\} = H_{1e}(f)H_{2e}(f) - H_{1o}(f)H_{2o}(f) + H_{1e}(f)H_{2o}(f) + H_{1o}(f)H_{2e}(f).$$

Collecting the terms, we get

$$\begin{aligned} HT\{x_1(t)*x_2(t)\} &= H_{1e}(f)\ [H_{2e}(f) + H_{2o}(f)] \\ &\quad - H_{1o}(f)\ [H_{20}(f) - H_{2e}(f)]. \\ &= H_{1e}(f)H_2(f) - H_{1o}(f)\ [H_{2o}(f) - H_{2e}(f)]. \\ &= [\{H_1(f) + H_1(-f)\} / 2]H_2(f) \\ &\quad - [\{H_1(f) - H_1(-f)\} / 2]\ [\{H_2(f) - H_2(-f)\} \\ &\quad - \{H_2(f) + H_2(-f)\}] / 2. \end{aligned}$$

Simplifying the above expression, we get

$$HT\ \{x_1(t)*x_2(t)\} = (1/2)\ [H_1(f)H_2(f) + H_1(-f)H_2(f) + H_1(f)H_2(-f) - H_1(-f)H_2(-f)].$$

However, if we have any symmetries in the functions being convolved, the above result simplifies. Following is a summary:

<u>SYMMETRY</u>	<u>HT OF THE CONVOLUTION</u>
$H_1(f)$ is even	$H_1(f)H_2(f)$
$H_2(f)$ is even	$H_1(f)H_2(f)$
Both are even	$H_1(f)H_2(f)$
$H_1(f)$ is odd	$H_1(f)H_2(-f)$
$H_2(f)$ is odd	$H_1(-f)H_2(f)$
Both are odd	$-H_1(f)H_2(f)$

2.4.14 Product Theorem

If two signals $x_1(t)$ and $x_2(t)$ have FTs $X_1(f)$ and $X_2(f)$, respectively, we know that the FT of the product of the two signals is the convolution of the individual FTs i.e.,

$$FT\{x_1(t)x_2(t)\} = X_1(f)*X_2(f).$$

By (2.14),

$$X_1(f)*X_2(f) = [H_{1e}(f) - jH_{1o}(f)] * [H_{2e}(f) - jH_{2o}(f)].$$

Rearranging the right hand side of the above equation into real and even parts, we get

$$X_1(f)*X_2(f) = [H_{1e}(f) * H_{2e}(f) - H_{1o}(f) * H_{2o}(f)]$$
$$-j[H_{1e}(f) * H_{2o}(f) + H_{1o}(f) * H_{2e}(f)].$$

(2.15) tells us that the HT of the product is obtained by subtracting the imaginary part of the above expression from its real part. Therefore,

$$HT\{x_1(t)x_2(t)\} = H_{1e}(f) * H_{2e}(f) - H_{1o}(f) * H_{2o}(f)$$
$$+ H_{1e}(f) * H_{2o}(f) + H_{1o}(f) * H_{2e}(f).$$

since

$$H_{1e}(f) = [H_1(f) + H_1(-f)] / 2,$$

$$H_{1o}(f) = [H_1(f) - H_1(-f)] / 2,$$

$$H_{2e}(f) = [H_2(f) + H_2(-f)] / 2,$$

$$H_{2o}(f) = [H_2(f) - H_2(-f)] / 2,$$

the above equation can be simplified as

$$HT\{x_1(t)x_2(t)\} = (1/2)[H_1(f) * H_2(f)$$

$$+ H_1(-f) * H_2(f) + H_1(f) * H_2(-f)$$

$$- H_1(-f) * H_2(-f)].$$

However, if the functions being multiplied possess some symmetries, the above result simplifies. Following is a summary:

SYMMETRY	HT OF THE PRODUCT
$H_1(f)$ is even	$H_1(f)*H_2(f)$
$H_2(f)$ is even	$H_1(f)*H_2(f)$
Both are even	$H_1(f)*H_2(f)$
$H_1(f)$ is odd	$H_1(f)*H_2(-f)$
$H_2(f)$ is odd	$H_1(-f)*H_2(f)$
Both are odd	$-H_1(f)*H_2(f)$

2.4.15 Cross-Correlation Theorem

If two signals $x_1(t)$ and $x_2(t)$ have FTs $X_1(f)$ and $X_2(f)$, respectively, the FT of the cross-correlation of the two signals is given by $X_1^*(f)X_2(f)$, where the superscript ' * ' indicates the complex conjugate. Using the relation between the FT and the HT, we get

$$X_1^*(f)X_2(f) = [H_{1e}(f) - jH_{1o}(f)]^* [H_{2e}(f) - jH_{2o}(f)]$$

Carrying out the multiplication and rearranging the above expression, we get

$$X_1^*(f)X_2(f) = [H_{1e}(f)H_{2e}(f) + H_{1o}(f)H_{2o}(f)]$$

$$- j[H_{1e}(f)H_{2o}(f) - H_{1o}(f)H_{2e}(f)].$$

From (2.15), we know that the HT of the cross-correlation is obtained by subtracting the imaginary part of the above expression from its real part.

HT{cross-correlation of $x_1(t)$ and $x_2(t)$} =

$$H_{1e}(f)H_{2e}(f) + H_{1o}(f)H_{2o}(f) + H_{1e}(f)H_{2o}(f) - H_{1o}(f)H_{2e}(f).$$

Substituting for $H_{1e}(f)$, $H_{1o}(f)$, $H_{2e}(f)$, $H_{2o}(f)$ in terms of $H_1(f)$, $H_2(f)$, $H_1(-f)$ and $H_2(-f)$ in the above expression, we get

HT{cross correlation of $x_1(t)$ and $x_2(t)$} =

$$(1/2)[H_1(f)H_2(f) + H_1(-f)H_2(f) - H_1(f)H_2(-f) + H_1(-f)H_2(-f)].$$

However, if one of the functions in the cross-correlation possesses some symmetry, the above result simplifies. Following is a summary:

SYMMETRY	HT OF THE CROSS CORRELATION
$H_1(f)$ is even	$H_1(f)H_2(f)$
$H_2(f)$ is even	$H_1(-f)H_2(f)$
Both are even	$H_1(f)H_2(f)$
$H_1(f)$ is odd	$-H_1(f)H_2(-f)$
$H_2(f)$ is odd	$H_1(f)H_2(f)$
Both are odd	$H_1(f)H_2(f)$

2.4.16 Parseval's Theorem

Parseval's theorem states that

$$\int_{-\infty}^{\infty} | x(t)|^2 dt = \int_{-\infty}^{\infty} | X(f)|^2 df$$

where

X(f) is the FT of the signal x(t).

The above theorem holds true for the HT also.

Noting that

$$x(t) = \int_{-\infty}^{\infty} H(f)\mathrm{cas}(2\pi ft)df,$$

we can write

$$\int_{-\infty}^{\infty} x^2(t)dt = \int_{-\infty}^{\infty} x(t) \int_{\infty}^{\infty} H(f)\mathrm{cas}(2\pi ft)dfdt.$$

Interchanging the order of integration on the right hand side, we get

$$\int_{\infty}^{\infty} x^2(t)dt = \int_{-\infty}^{\infty} H(f) \int_{-\infty}^{\infty} x(t)\mathrm{cas}(2\pi ft)dt\ df.$$

The second integral on the right hand side is the HT of the signal.

Therefore,

$$\int_{-\infty}^{\infty} x^2(t)dt = \int_{-\infty}^{\infty} H^2(f)df.$$

Wang [5] described a real series representation for periodic signals. Since from a signal processing point of view we are more interested in data sequences than in CT signals, in the next chapter the DHT is defined and its properties are derived.

3. THE DISCRETE HARTLEY TRANSFORM

3.1 Introduction

We begin the chapter by defining the DFT of a finite-length sequence. The DHT as defined by Bracewell [2] is presented next. Finally, many useful properties of the DHT are derived and comparisons made with those of the DFT.

3.2 The Discrete Fourier Transform

The DFT $\{X(k)\}$ of a finite-duration sequence $\{x(n)\} = \{x(0), x(1), \ldots, x(N-1)\}$ is defined as

$$X(k) = (1/N) \sum_{n=0}^{N-1} x(n) W_N^{-nk}, \quad k = 0, 1, \ldots, N-1 \tag{3.1}$$

where

W_N is called the kernel and is given by $W_N = e^{-j2\pi/N}$.

Using the definition for the kernel given above, we can expand (3.1) as

$$X(k) = (1/N) \sum_{n=0}^{N-1} x(n)\cos(2\pi nk/N)$$

$$- j(1/N) \sum_{n=0}^{N-1} x(n)\sin(2\pi nk/N), \quad k = 0, 1, \ldots, N-1. \tag{3.2}$$

From (3.2) it is clear that the DFT of a sequence is, in general, complex, with the real part

$$\mathrm{Re}\,\{X(k)\} = (1/N)\sum_{n=0}^{N-1} x(n)\cos(2\pi kn/N), \tag{3.3}$$

and the imaginary part

$$\mathrm{Im}\,\{X(k)\} = (1/N)\sum_{n=0}^{N-1} x(n)\sin(2\pi kn/N). \tag{3.4}$$

To recover {x(n)} from {X(k)}, we perform the inverse DFT (IDFT), which is defined as

$$x(n) = \sum_{n=0}^{N-1} X(k)W_N^{nk}, \quad n = 0, 1, \ldots, N-1 \tag{3.5}$$

where, W_N is the kernel as defined earlier.

3.3 The Discrete Hartley Transform

The DHT H(k) of an N-point real sequence {x(n)} is defined as

$$H(k) = (1/N)\sum_{n=0}^{N-1} x(n)\mathrm{cas}(2\pi nk/N), \quad k = 0, 1, \ldots, N-1. \tag{3.6}$$

where

$\mathrm{cas}(2\pi nk/N) = \cos(2\pi nk/N) + \sin(2\pi nk/N).$

Using the above expansion for cas(2πnk/N), we split (3.6) into two summations as shown below.

$$H(k) = (1/N)\sum_{n=0}^{N-1} x(n)\cos(2\pi nk/N) + (1/N)\sum_{n=0}^{N-1} x(n)\sin(2\pi nk/N),$$
$$k = 0, 1, \ldots, N-1. \tag{3.7}$$

We can compute H(−k) from the above equation as

$$H(-k) = (1/N) \sum_{n=0}^{N-1} x(n)\cos\{2\pi n(-k)/N\}$$

$$+ (1/N) \sum_{n=0}^{N-1} x(n)\sin\{2\pi n(-k)/N\}, \ k = 0, 1, \ldots, N-1. \tag{3.8}$$

Using the trigonometric identities

$$\cos\{2\pi n(-k)\} = \cos(2\pi nk)$$

and

$$\sin\{2\pi n(-k)\} = - \sin(2\pi nk)$$

we can rewrite (3.8) as

$$H(-k) = (1/N) \sum_{n=0}^{N-1} x(n)\cos(2\pi nk/N)$$

$$- (1/N) \sum_{n=0}^{N-1} x(n)\sin(2\pi nk/N), \ k = 0, 1, \ldots, N-1. \tag{3.9}$$

We can split $H(k)$ into even and odd parts, with the even part $H_e(k)$ being given by

$$H_e(k) = [H(k) + H(-k)] / 2, \tag{3.10}$$

and the odd part $H_o(k)$ obtained as

$$H_o(k) = [H(k) - H(-k)] / 2. \tag{3.11}$$

Substituting the summations of (3.7) and (3.9) for $H(k)$ and $H(-k)$ in (3.10) and (3.11), we get

$$H_e(k) = (1/N) \sum_{n=0}^{N-1} x(n)\cos(2\pi nk/N), \ k = 0, 1, \ldots, N-1 \tag{3.12}$$

$$H_o(k) = (1/N) \sum_{n=0}^{N-1} x(n)\sin(2\pi nk/N), \ k = 0, 1, \ldots, N-1. \tag{3.13}$$

In all the above equations, $H(-k) = H(N - k)$. For example, for a sixteen point sequence, i.e. $N = 16$, $H(-1)$ is $H(15)$, $H(-2)$ is $H(14)$ and so on. The notation $H(-k)$ instead of $H(N - k)$ is only an extension of the habit formed in dealing with the CT functions.

Unlike the DFT, the DHT does not have a separate inverse DHT. If we want to recover the original data sequence $\{x(n)\}$, we only need to perform the DHT operation on $\{H(k)\}$.

$$x(n) = \sum_{k=0}^{N-1} H(k)\mathrm{cas}(2\pi nk/N), \ n = 0, 1, \ldots, N - 1. \tag{3.14}$$

Note the absence of the (1/N) factor in the above equation before the summation symbol, which was present when we defined H(k). Its absence does not alter the fact that the DHT is symmetric, because by symmetry we mean invariance of the kernel in (3.6) and (3.14). The factor (1/N) is only a constant that does not affect the nature of the transform. In fact, some authors omit (1/N) in (3.6) as well.

If we compare (3.7) and (3.2), it is evident that for any real sequence the DHT generates a real-valued sequence, whereas the DFT produces a complex-valued sequence. In fact, it is this particular property of the DHT that makes it attractive in many applications.

3.4 Properties

The DHT possesses many useful properties that enable us to develop fast algorithms to compute it, and also apply it successfully in various signal processing applications such as digital filtering, fast convolution, spectral analysis, etc. In this section many of its useful properties are developed.

3.4.1 Relation Between the DFT and the DHT

The DFT and the DHT are so closely related that we can move easily from one transform to the other. By looking at (3.3) and (3.12) we observe that

$$\mathrm{Re}\{X(k)\} = H_e(k)$$

and

from (3.4) and (3.13) we note that

$$\mathrm{Im}\{X(k)\} = H_o(k).$$

Therefore, if we have the DHT $\{H(k)\}$ of an N-point real-valued sequence, we can find the DFT of the sequence as shown below.

$$X(k) = H_e(k) - jH_o(k),\ k = 0,\ 1,\ \ldots,\ N - 1. \tag{3.15}$$

Referring to (3.3), (3.4), (3.12), and (3.13) we note that if we have the DFT $\{X(f)\}$ of a real-valued sequence, we can compute the DHT $\{H(k)\}$ as

$$H(k) = \mathrm{Re}\{X(k)\} - \mathrm{Im}\{X(k)\},\ k = 0,\ 1,\ \ldots,\ N - 1. \tag{3.16}$$

3.4.2 Periodicity of the Kernel

We know that the DHT of an N-point real-valued sequence is given by

$$H(k) = (1/N) \sum_{n=0}^{N-1} x(n)\mathrm{cas}(2\pi nk/N),\ k = 0,\ 1,\ \ldots,\ N - 1.$$

In the above equation the factor cas($2\pi nk/N$) is called the kernel. Let us focus on the kernel briefly.

$$\mathrm{cas}[2\pi n(k + N)/N] = \cos[2\pi n(k + N)/N] + \sin[2\pi n(k + N)/N]$$

Expanding the right hand side using the familiar trigonometric identities, we get

$$\begin{aligned}\mathrm{cas}[2\pi n(k + N)/N] &= \cos(2\pi nk/N)\cos(2\pi nN/N) - \sin(2\pi nk/N)\sin(2\pi nN/N) \\ &\quad + \sin(2\pi nk/N)\cos(2\pi nN/N) + \cos(2\pi nk/N)\sin(2\pi nN/N). \\ &= \cos(2\pi nk/N) + \sin(2\pi nk/N) \\ &= \mathrm{cas}(2\pi nk/N).\end{aligned}$$

The above equation shows that the kernel is periodic on N. In the light of this fact let us compute the (k + N)th element in the DHT of the data sequence {x(n)}.

$$H(k + N) = (1/N)\sum_{n=0}^{N-1} x(n)\mathrm{cas}[2\pi n(k + N)/N],\ k = 0, 1, \ldots, N - 1.$$

Since the kernel is periodic on N,

$$\begin{aligned}H(k + N) &= (1/N)\sum_{n=0}^{N-1} x(n)\mathrm{cas}(2\pi nk/N),\ k = 0, 1, \ldots, N - 1. \\ &= H(k).\end{aligned}$$

Therefore, each of the first N elements in the DHT sequence repeats itself after an interval of N points. Now, let us compute the $(n + N)^{th}$ element in the data sequence {x(n)} from its DHT.

$$x(n + N) = \sum_{k=0}^{N-1} H(k)\mathrm{cas}[2\pi k(n + N)/N],\ n = 0, 1, \ldots, N - 1.$$

Since the kernel is periodic on N, the above equation reduces to

$$x(n + N) = \sum_{k=0}^{N-1} H(k)\mathrm{cas}(2\pi nk/N), \quad n = 0, 1, \ldots, N - 1.$$

$$= x(n).$$

As we have seen above, the periodicity of the kernel imposes periodicity on the data sequence as well as its DHT. If the length of the sequence is N, whenever we perform the DHT on it, the sequence is implicitly assumed to be periodic with period N. In this respect the DHT is identical to the DFT.

3.4.3 Orthogonality of the Kernel

The kernel of the DHT is orthogonal and real, which makes the DHT an orthogonal transform. Let us consider the following summation:

$$\sum_{k=0}^{N-1} \mathrm{cas}(2\pi nk/N)\mathrm{cas}(2\pi mk/N) \tag{3.17}$$

Expanding the terms in the summation, we get

$$\sum_{k=0}^{N-1} [\cos(2\pi nk/N) + \sin(2\pi nk/N)]\ [\cos(2\pi mk/N) + \sin(2\pi mk/N)]$$

Simplifying the above summation further, we get

$$\sum_{k=0}^{N-1} \cos(2\pi nk/N)\cos(2\pi mk/N) + \sum_{k=0}^{N-1} \sin(2\pi nk/N)\cos(2\pi mk/N)$$

$$+ \sum_{k=0}^{N-1} \cos(2\pi nk/N)\sin(2\pi mk/N) + \sum_{k=0}^{N-1} \sin(2\pi nk/N)\sin(2\pi mk/N).$$

Collecting the terms and combining them using the familiar trigonometric identities, we get

$$\sum_{k=0}^{N-1} \text{cas}(2\pi nk/N)\text{cas}(2\pi mk/N) = \sum_{k=0}^{N-1} \cos[2\pi(n - m)k/N] + \sum_{k=0}^{N-1} \sin[2\pi(n + m)k/N].$$

However, both the summations on the right hand side are zero.

Therefore

$$\sum_{k=0}^{N-1} \text{cas}(2\pi mk/N)\text{cas}(2\pi nk/N) = 0.$$

Let m = n in (3.17). Then it becomes

$$\begin{aligned}\sum_{k=0}^{N-1} \text{cas}(2\pi nk/N)\text{cas}(2\pi nk/N) &= \sum_{k=0}^{N-1} [\text{cas}(2\pi nk/N)]^2 \\ &= \sum_{k=0}^{N-1} [\cos(2\pi nk/N) + \sin(2\pi nk/N)]^2 \\ &= \sum_{k=0}^{N-1} \cos^2(2\pi nk/N) + \sin^2(2\pi nk/N) \\ &\quad + 2\cos(2\pi nk/N)\sin(2\pi nk/N). \\ &= \sum_{k=0}^{N-1} 1 + 2 \sum_{k=0}^{N-1} \cos(2\pi nk/N)\sin(2\pi nk/N) \\ &= N + 0 \text{ (Since sine and cosine or orthogonal} \\ &\qquad\qquad \text{to each other over one full cycle)} \\ &= N.\end{aligned}$$

Therefore, we have proved that

$$\sum_{k=0}^{N-1} \text{cas}(2\pi mk/N)\text{cas}(2\pi nk/N) = 0, \text{ if } m \text{ is not equal to } n \text{ and}$$

$$= N, \text{ if } m = n.$$

Hence, the kernel is orthogonal and the DHT is an orthogonal transform. The DFT also is an orthogonal transform but the kernel is complex in its case.

3.4.4 Linearity

The DHT is a linear transform like the DFT. Let us consider two N-point sequences $\{x_1(n)\}$ and $\{x_2(n)\}$. Then the DHT of the summation is given by

$$H(k) = (1/N) \sum_{n=0}^{N-1} [x_1(n) + x_2(n)]\text{cas}(2\pi nk/N), \; k = 0, 1, \ldots, N-1.$$

However, we can split the above summation into two summations as shown

$$H(k) = (1/N) \sum_{n=0}^{N-1} x_1(n)\text{cas}(2\pi nk/N) + (1/N) \sum_{n=0}^{N-1} x_2(n)\text{cas}(2\pi nk/N),$$

$$k = 0, 1, \ldots, N-1.$$

$$= H_1(k) + H_2(k)$$

Where, $H_1(k)$ and $H_2(k)$ are the DHTs of the sequences $\{x_1(n)\}$ and $\{x_2(n)\}$ respectively. Therefore, the DHT of a sum of sequences is the the sum of the DHTs of the sequences taken independently. If lengths of the sequences are not identical, we make them so by adding zeros to the data sequence having less number of data points before applying this theorem.

3.4.5 Reversal

Let us take an N-point data sequence $\{x(n)\}$ and arrange all the elements in the reverse order to get the sequence $\{x(N - n)\}$. The DHT of $\{x(N -)\}$ is simply a reversed version of the DHT $\{H(k)\}$ of $\{x(n)\}$, except for the first element. That is, if $\{x(0), x(1), \ldots, x(7)\}$ has the DHT sequence $\{H(0), H(1), \ldots, H(7)\}$, then the DHT of the sequence $\{x(0), x(7), x(6), \ldots, x(1)\}$ is $\{H(0), H(7), H(6), \ldots, H(1)\}$.

3.4.6 Transforms of Even and Odd Sequences

We know that the DHT of a sequence $\{x(n)\}$ is given by

$$H(k) = (1/N) \sum_{n=0}^{N-1} x(n)\mathrm{cas}(2\pi nk/N), \quad k = 0, 1, \ldots, N - 1.$$

For a data sequence that is even, we can replace $x(n)$ by $x(-n)$ in the above equation.

$$H(k) = (1/N) \sum_{n=0}^{N-1} x(-n)\mathrm{cas}(2\pi nk/N), \quad k = 0, 1, \ldots, N - 1.$$

By the reversal theorem, the right hand side of the above equation is $H(-k)$.

Therefore

$$H(k) = H(-k).$$

The above result shows that the DHT of an even real-valued sequence is also even.

The above result can also be proved via the DFT. From (3.15) we know that X(k) is related to H(k) by

$$X(k) = H_e(k) - jH_o(k).$$

Since the DFT of a real and even-sequence is real and, as such, does not have any imaginary part, from the above equation we conclude that the DHT of the data sequence does not have any odd part.

Now, let us consider a data sequence that is odd. i.e.,

$$x(n) = - x(-n).$$

The DHT of {x(n)} is given by,

$$H(k) = (1/N) \sum_{n=0}^{N-1} x(n)\mathrm{cas}(2\pi nk/N),\ k = 0,\ 1,\ \ldots,\ N - 1.$$

Since the data sequence is odd, we replace x(n) by -x(-n) in the above equation.

$$H(k) = (1/N) \sum_{n=0}^{N-1} -x(-n)\mathrm{cas}(2\pi nk/N).$$

By the reversal theorem, the right hand side is -H(-k). Hence

$$H(k) = - H(-k).$$

We can reach the same conclusion from our knowledge that the DFT of a real and odd-sequence is imaginary. Applying that fact to Eqn. (3.15), we can conclude that the DHT of an odd and real sequence is also odd.

3.4.7 Shift Theorem

Let us consider a data sequence {x(n)} whose DHT is {H(k)}. Now, let us circularly shift the data sequence by m samples to the left.

When we do that, the samples that are pushed beyond the left edge of the sequence will reappear at its right end, thus always maintaining the length of the data sequence as N.

For example, consider an 8-point data sequence {x(n)} as shown below.

$$\{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}.$$

If we circularly shift {x(n)} by two samples to the left, we get the following sequence.

$$\{x(2), x(3), x(4), x(5), x(6), x(7), x(0), x(1)\}.$$

Observe that the length of the data sequence is same, that is eight, in both the cases.

Now, let us find the DHT of such a shifted sequence.

$$H[x(n + m)] = (1/N) \sum_{n=0}^{N-1} x(n + m)\mathrm{cas}(2\pi nk/N), \quad k = 0, 1, \ldots, N - 1. \quad (3.18)$$

where m is the amount of shift.

Let $n + m = t$. Then $n = t - m$.

Also, when $n = 0$, $t = m$ and when $n = N - 1$, $t = m + N - 1$.

Making the above substitutions for n and the limits of summation in (3.18), we get

$$H[x(n + m)] = (1/N) \sum_{t=m}^{N-1+m} x(t)\mathrm{cas}[2\pi(t - m)k/N], \quad k = 0, 1, \ldots, N - 1.$$

Expanding $\mathrm{cas}[2\pi(t - m)k/N]$ and splitting the above summation into two, we get

$$H[x(n + m)] = (1/N) \sum_{t=m}^{N-1+m} x(t)\cos\{2\pi(t - m)k/N\}$$

$$+ (1/N) \sum_{t=m} x(t)\sin\{2\pi(t - m)k/N\},\ k = 0,\ 1,\ \ldots,\ N - 1.$$

Using the familiar trigonometric identities we can rewrite the above summations as follows:

$$H[x(n + m)] = (1/N) \sum_{t=m}^{N-1+m} x(t)[\cos(2\pi tk/N)\cos(2\pi mk/N)$$

$$+ \sin(2\pi tk/N)\sin(2\pi mk/N)]$$

$$+ (1/N) \sum_{t=m}^{N-1+m} x(t)[\sin(2\pi tk/N)\cos(2\pi mk/N)$$

$$- \cos(2\pi tk/N)\sin(2\pi mk/N)].$$

Collecting the terms, we get

$$H[x(n + m)] = (1/N)\cos(2\pi mk/N) \sum_{t=m}^{N-m+1} x(t)[\cos(2\pi tk/N) + \sin(2\pi tk/N)]$$

$$- (1/N)\sin(2\pi mk/N) \sum_{t=m}^{N-m+1} x(t)[\cos(2\pi tk/N) - \sin(2\pi tk/N)].$$

Recognizing that

$$\cos(2\pi tk/N) + \sin(2\pi tk/N) = \mathrm{cas}(2\pi tk/N)$$

$$\cos(2\pi tk/N) - \sin(2\pi tk/N) = \mathrm{cas}\{2\pi\tau(-k)/N\},$$

we can rewrite the above equation as

$$H[x(n + m)] = \cos(2\pi mk/N)(1/N) \sum_{t=m}^{N-1+m} x(t)\mathrm{cas}(2\pi tk/N)$$

$$- \sin(2\pi mk/N)(1/N) \sum_{t=m}^{N-1+m} x(t)\mathrm{cas}\{2\pi t(-k)/N\}.$$

$$H[x(n + m)] = \cos(2\pi mk/N)H(k) - \sin(2\pi mk/N)H(-k)$$

$$= \cos(2\pi mk/N)H(k) - \sin(2\pi mk/N)H(N-k).$$

If $H(k)$ is even, then the above result becomes

$$H(k)[\mathrm{cas}\{2\pi m(-k)\}],$$

and if $H(k)$ is odd, it will be $H(k)\mathrm{cas}(2\pi mk/N)$.

3.4.8 Convolution

Circular convolution of two data sequences can be implemented easily using the DHT. To perform circular convolution of two sequences, both the sequences must be of identical length, unlike in linear convolution where differing lengths are allowed. Circular convolution involves time reversing one sequence, overlaying it on the other and then carrying out the convolution in the usual manner, remembering that whenever we shift an element, the shift is circular.

If we have two data sequences $\{x_1(n)\}$ and $\{x_2(n)\}$, each of length N, then their circular convolution is

$$y(n) = (1/N) \sum_{h=0}^{N-1} x_1(h)x_2(n - h), \; n = 0, 1, \ldots, N - 1.$$

If the DFTs of $\{x_1(n)\}$ and $\{x_2(n)\}$ are $X_1(k)$ and $X_2(k)$ respectively, then the DFT of the circular convolution of the two sequences is

$$F[x_1(n) * x_2(n)] = X_1(k)X_2(k).$$

By Eqn. (3.15), we rewrite the right hand side as

$$X_1(k)X_2(k) = [H_{1e}(k) - jH_{1o}(k)][H_{2e}(k) - jH_{2o}(k)]$$

Where $H_{1e}(k)$, $H_{1o}(k)$, $H_{2e}(k)$, $H_{2o}(k)$ are the even- and odd- parts respectively, of the DHTs $H_1(k)$ and $H_2(k)$ of the data sequences.

Carrying out the multiplication in the above equation, we get

$$X_1(k)X_2(k) = [H_{1e}(k)H_{2e}(k) - H_{1o}(k)H_{2o}(k)]$$
$$-j[H_{1e}(k)H_{2o}(k) + H_{1o}(k)H_{2e}(k)].$$

Using the result of (3.16) we obtain the DHT of the convolution as

$$H[x_1(n) * x_2(n)] = H_{1e}(k)[H_{2e}(k) + H_{2o}(k)] - H_{1o}(k)[H_{2o}(k) - H_{2e}(k)].$$

Writing $H_{1e}(k)$, $H_{2e}(k)$, $H_{1o}(k)$, $H_{2o}(k)$ in terms of $H_1(k)$, $H_1(-k)$, $H_2(k)$, and $H_2(-k)$, we get

$$H[x_1(n)*x_2(n)] = (1/2)[H_1(k)H_2(k) + H_1(-k)H_2(k) + H_1(k)H_2(-k)$$
$$- H_1(-k)H_2(-k)].$$

In some special cases we get simpler results. Following is a summary:

SYMMETRY	DHT OF THE CIRCULAR CONVOLUTION
$\{x_1(n)\}$ is even	$H_1(k)H_2(k)$
$\{x_2(n)\}$ is even	$H_1(k)H_2(k)$
Both sequences are even	$H_1(k)H_2(k)$
$\{x_1(n)\}$ is odd	$H_1(k)H_2(-k)$
$\{x_2(n)\}$ is odd	$H_1(-k)H_2(k)$
Both sequences are odd	$- H_1(k)H_2(k)$

Whenever we need to perform linear convolution of two sequences, $\{x_1(n)\}$ of length N_1 and $\{x_2(n)\}$ of length N_2, $N_2 > N_1$, we add a string of zeros to both the sequences until the length of each becomes at least equal to $(N_1 + N_2 - 1)$, and then perform circular convolution on the resulting sequences.

3.4.9 Product Theorem

Let us consider two data sequences $\{x_1(n)\}$ and $\{x_2(n)\}$, each of length N. If their DFT sequences are $\{X_1(k)\}$ and $\{X_2(k)\}$ respectively, then the DFT of their product is given by the circular convolution of $X_1(k)$ and $X_2(k)$. That is,

$$F[x_1(n)x_2(n)] = (1/N)\ [X_1(k) * X_2(k)]$$

Using the result of (3.15) we can rewrite the r.., t hand side of the above equation as

$$(1/N)[X_1(k) * X_2(k)] = (1/N)[H_{1e}(k) - jH_{1o}(k)] * [H_{2e}(k) - jH_{2o}(k)]$$

Carrying out the multiplication and rearranging the above expression, we get

$$(1/N)[X_1(k) * X_2(k)] = (1/N)[\{H_{1e}(k) * H_{2e}(k) - H_{1o}(k) * H_{2o}(k)\}$$
$$-j\{H_{1e}(k) * H_{2o}(k) + H_{1o}(k) * H_{2e}(k)\}].$$

Using the result of (3.16) we obtain the DHT of the product $\{x_1(n)x_2(n)\}$ as shown below.

$$H[x_1(n)x_2(n)] = (1/N)[H_{1e}(k)*\{H_{2e}(k) + H_{2o}(k)\}$$

$$- H_{1o}(k)*\{H_{2o}(k) - H_{2e}(k)\}].$$

Observing that

$$H_{1e}(k) = [H_1(k) + H_1(-k)]/2 \quad H_{1o}(k) = [H_1(k) - H_1(-k)]/2,$$

$$H_{2e}(k) = [H_2(k) + H_2(-k)]/2 \quad H_{2o}(k) = [H_2(k) - H_2(-k)]/2$$

we get

$$H[x_1(n)x_2(n)] = (1/2N)[H_1(k) * H_2(k) + H_1(-k) * H_2(k)$$
$$+ H_1(k) * H_2(-k) - H_1(-k) * H_2(-k)]$$

If there is any symmetry in one or both of the multiplying sequences, the above result simplifies. Following is a summary.

<u>SYMMETRY</u>	<u>DHT OF THE PRODUCT</u>
$\{x_1(n)$ is even$\}$	$(1/N)[H_1(k)H_2(k)]$
$\{x_2(n)$ is even$\}$	$(1/N)[H_1(k)H_2(k)]$
Both sequences are even	$(1/N)[H_1(k)H_2(k)]$
$\{x_1(n)$ is odd$\}$	$(1/N)[H_1(k)H_2(-k)]$
$\{x_2(n)$ is odd$\}$	$(1/N)[H_1(-k)H_2(k)]$
Both sequences are odd	$(1/N)[-H_1(k)H_2(k)]$

3.4.10 Cross-Correlation Theorem

Circular cross-correlation of two data sequences $\{x_1(n)\}$ and $\{x_2(n)\}$ is given by

$$y(n) = (1/N) \sum_{h=0}^{N-1} x_1(h)x_2(n+h), \; n = 0, 1, \ldots, N-1. \qquad (3.19)$$

Collecting the terms and combining them using the familiar trigonometric identities, we get

$$\sum_{k=0}^{N-1} \text{cas}(2\pi nk/N)\text{cas}(2\pi mk/N) = \sum_{k=0}^{N-1} \cos[2\pi(n - m)k/N] + \sum_{k=0}^{N-1} \sin[2\pi(n + m)k/N].$$

However, both the summations on the right hand side are zero.

Therefore

$$\sum_{k=0}^{N-1} \text{cas}(2\pi mk/N)\text{cas}(2\pi nk/N) = 0.$$

Let m = n in (3.17). Then it becomes

$$\begin{aligned}\sum_{k=0}^{N-1} \text{cas}(2\pi nk/N)\text{cas}(2\pi nk/N) &= \sum_{k=0}^{N-1} [\text{cas}(2\pi nk/N)]^2 \\ &= \sum_{k=0}^{N-1} [\cos(2\pi nk/N) + \sin(2\pi nk/N)]^2 \\ &= \sum_{k=0}^{N-1} \cos^2(2\pi nk/N) + \sin^2(2\pi nk/N) \\ &\quad + 2\cos(2\pi nk/N)\sin(2\pi nk/N). \\ &= \sum_{k=0}^{N-1} 1 + 2 \sum_{k=0}^{N-1} \cos(2\pi nk/N)\sin(2\pi nk/N) \\ &= N + 0 \text{ (Since sine and cosine or orthogonal} \\ &\qquad \text{to each other over one full cycle)} \\ &= N.\end{aligned}$$

Therefore, we have proved that

$$\sum_{k=0}^{N-1} \text{cas}(2\pi mk/N)\text{cas}(2\pi nk/N) = 0, \text{ if } m \text{ is not equal to } n \text{ and}$$

$$= N, \text{ if } m = n.$$

Hence, the kernel is orthogonal and the DHT is an orthogonal transform. The DFT also is an orthogonal transform but the kernel is complex in its case.

3.4.4 Linearity

The DHT is a linear transform like the DFT. Let us consider two N-point sequences $\{x_1(n)\}$ and $\{x_2(n)\}$. Then the DHT of the summation is given by

$$H(k) = (1/N) \sum_{n=0}^{N-1} [x_1(n) + x_2(n)]\text{cas}(2\pi nk/N), \; k = 0, 1, \ldots, N - 1.$$

However, we can split the above summation into two summations as shown

$$H(k) = (1/N) \sum_{n=0}^{N-1} x_1(n)\text{cas}(2\pi nk/N) + (1/N) \sum_{n=0}^{N-1} x_2(n)\text{cas}(2\pi nk/N),$$

$$k = 0, 1, \ldots, N - 1.$$

$$= H_1(k) + H_2(k)$$

Where, $H_1(k)$ and $H_2(k)$ are the DHTs of the sequences $\{x_1(n)\}$ and $\{x_2(n)\}$ respectively. Therefore, the DHT of a sum of sequences is the the sum of the DHTs of the sequences taken independently. If lengths of the sequences are not identical, we make them so by adding zeros to the data sequence having less number of data points before applying this theorem.

3.4.5 Reversal

Let us take an N-point data sequence {x(n)} and arrange all the elements in the reverse order to get the sequence {x(N - n)}. The DHT of {x(N -)} is simply a reversed version of the DHT {H(k)} of {x(n)}, except for the first element. That is, if {x(0), x(1), ..., x(7)} has the DHT sequence {H(0), H(1), ..., H(7)}, then the DHT of the sequence {x(0), x(7), x(6), ..., x(1)} is {H(0), H(7), H(6), ..., H(1)}.

3.4.6 Transforms of Even and Odd Sequences

We know that the DHT of a sequence {x(n)} is given by

$$H(k) = (1/N) \sum_{n=0}^{N-1} x(n)\text{cas}(2\pi nk/N), \quad k = 0, 1, \ldots, N - 1.$$

For a data sequence that is even, we can replace x(n) by x(-n) in the above equation.

$$H(k) = (1/N) \sum_{n=0}^{N-1} x(-n)\text{cas}(2\pi nk/N), \quad k = 0, 1, \ldots, N - 1.$$

By the reversal theorem, the right hand side of the above equation is H(-k).

Therefore

$$H(k) = H(-k).$$

The above result shows that the DHT of an even real-valued sequence is also even.

The above result can also be proved via the DFT. From (3.15) we know that X(k) is related to H(k) by

$$X(k) = H_e(k) - jH_o(k).$$

Since the DFT of a real and even-sequence is real and, as such, does not have any imaginary part, from the above equation we conclude that the DHT of the data sequence does not have any odd part.

Now, let us consider a data sequence that is odd. i.e.,

$$x(n) = - x(-n).$$

The DHT of {x(n)} is given by,

$$H(k) = (1/N) \sum_{n=0}^{N-1} x(n)\mathrm{cas}(2\pi nk/N), \; k = 0, 1, \ldots, N - 1.$$

Since the data sequence is odd, we replace x(n) by -x(-n) in the above equation.

$$H(k) = (1/N) \sum_{n=0}^{N-1} -x(-n)\mathrm{cas}(2\pi nk/N).$$

By the reversal theorem, the right hand side is -H(-k). Hence

$$H(k) = - H(-k).$$

We can reach the same conclusion from our knowledge that the DFT of a real and odd-sequence is imaginary. Applying that fact to Eqn. (3.15), we can conclude that the DHT of an odd and real sequence is also odd.

3.4.7 Shift Theorem

Let us consider a data sequence {x(n)} whose DHT is {H(k)}. Now, let us circularly shift the data sequence by m samples to the left.

When we do that, the samples that are pushed beyond the left edge of the sequence will reappear at its right end, thus always maintaining the length of the data sequence as N.

For example, consider an 8-point data sequence {x(n)} as shown below.

$$\{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}.$$

If we circularly shift {x(n)} by two samples to the left, we get the following sequence.

$$\{x(2), x(3), x(4), x(5), x(6), x(7), x(0), x(1)\}.$$

Observe that the length of the data sequence is same, that is eight, in both the cases.

Now, let us find the DHT of such a shifted sequence.

$$H[x(n + m)] = (1/N) \sum_{n=0}^{N-1} x(n + m)\operatorname{cas}(2\pi nk/N), \ k = 0, 1, \ldots, N - 1. \quad (3.18)$$

where m is the amount of shift.

Let $n + m = t$. Then $n = t - m$.

Also, when $n = 0$, $t = m$ and when $n = N - 1$, $t = m + N - 1$.

Making the above substitutions for n and the limits of summation in (3.18), we get

$$H[x(n + m)] = (1/N) \sum_{t=m}^{N-1+m} x(t)\operatorname{cas}[2\pi(t - m)k/N], \ k = 0, 1, \ldots, N - 1.$$

Expanding $\operatorname{cas}[2\pi(t - m)k/N]$ and splitting the above summation into two, we get

$$H[x(n + m)] = (1/N) \sum_{t=m}^{N-1+m} x(t)\cos\{2\pi(t - m)k/N\}$$

$$+ (1/N) \sum_{t=m} x(t)\sin\{2\pi(t - m)k/N\}, \quad k = 0, 1, \dots, N - 1.$$

Using the familiar trigonometric identities we can rewrite the above summations as follows:

$$H[x(n + m)] = (1/N) \sum_{t=m}^{N-1+m} x(t)[\cos(2\pi tk/N)\cos(2\pi mk/N) + \sin(2\pi tk/N)\sin(2\pi mk/N)]$$

$$+ (1/N) \sum_{t=m}^{N-1+m} x(t)[\sin(2\pi tk/N)\cos(2\pi mk/N) - \cos(2\pi tk/N)\sin(2\pi mk/N)].$$

Collecting the terms, we get

$$H[x(n + m)] = (1/N)\cos(2\pi mk/N) \sum_{t=m}^{N-m+1} x(t)[\cos(2\pi tk/N) + \sin(2\pi tk/N)]$$

$$- (1/N)\sin(2\pi mk/N) \sum_{t=m}^{N-m+1} x(t)[\cos(2\pi tk/N) - \sin(2\pi tk/N)].$$

Recognizing that

$\cos(2\pi tk/N) + \sin(2\pi tk/N) = \mathrm{cas}(2\pi tk/N)$

$\cos(2\pi tk/N) - \sin(2\pi tk/N) = \mathrm{cas}\{2\pi\tau(-k)/N\}$,

we can rewrite the above equation as

$$H[x(n + m)] = \cos(2\pi mk/N)(1/N) \sum_{t=m}^{N-1+m} x(t)\mathrm{cas}(2\pi tk/N)$$

$$- \sin(2\pi mk/N)(1/N) \sum_{t=m}^{N-1+m} x(t)\mathrm{cas}\{2\pi t(-k)/N\}.$$

$$H[x(n + m)] = \cos(2\pi mk/N)H(k) - \sin(2\pi mk/N)H(-k)$$

$$= \cos(2\pi mk/N)H(k) - \sin(2\pi mk/N)H(N-k).$$

If $H(k)$ is even, then the above result becomes

$$H(k)[\mathrm{cas}\{2\pi m(-k)\}],$$

and if $H(k)$ is odd, it will be $H(k)\mathrm{cas}(2\pi mk/N)$.

3.4.8 Convolution

Circular convolution of two data sequences can be implemented easily using the DHT. To perform circular convolution of two sequences, both the sequences must be of identical length, unlike in linear convolution where differing lengths are allowed. Circular convolution involves time reversing one sequence, overlaying it on the other and then carrying out the convolution in the usual manner, remembering that whenever we shift an element, the shift is circular.

If we have two data sequences $\{x_1(n)\}$ and $\{x_2(n)\}$, each of length N, then their circular convolution is

$$y(n) = (1/N) \sum_{h=0}^{N-1} x_1(h)x_2(n - h), \; n = 0, 1, \ldots, N - 1.$$

If the DFTs of $\{x_1(n)\}$ and $\{x_2(n)\}$ are $X_1(k)$ and $X_2(k)$ respectively, then the DFT of the circular convolution of the two sequences is $F[x_1(n) * x_2(n)] = X_1(k)X_2(k)$.

By Eqn. (3.15), we rewrite the right hand side as

$$X_1(k)X_2(k) = [H_{1e}(k) - jH_{1o}(k)][H_{2e}(k) - jH_{2o}(k)]$$

Where $H_{1e}(k)$, $H_{1o}(k)$, $H_{2e}(k)$, $H_{2o}(k)$ are the even- and odd- parts respectively, of the DHTs $H_1(k)$ and $H_2(k)$ of the data sequences.

Carrying out the multiplication in the above equation, we get

$$X_1(k)X_2(k) = [H_{1e}(k)H_{2e}(k) - H_{1o}(k)H_{2o}(k)]$$
$$-j[H_{1e}(k)H_{2o}(k) + H_{1o}(k)H_{2e}(k)].$$

Using the result of (3.16) we obtain the DHT of the convolution as

$$H[x_1(n) * x_2(n)] = H_{1e}(k)[H_{2e}(k) + H_{2o}(k)] - H_{1o}(k)[H_{2o}(k) - H_{2e}(k)].$$

Writing $H_{1e}(k)$, $H_{2e}(k)$, $H_{1o}(k)$, $H_{2o}(k)$ in terms of $H_1(k)$, $H_1(-k)$, $H_2(k)$, and $H_2(-k)$, we get

$$H[x_1(n)*x_2(n)] = (1/2)[H_1(k)H_2(k) + H_1(-k)H_2(k) + H_1(k)H_2(-k)$$
$$- H_1(-k)H_2(-k)].$$

In some special cases we get simpler results. Following is a summary:

SYMMETRY	DHT OF THE CIRCULAR CONVOLUTION
$\{x_1(n)\}$ is even	$H_1(k)H_2(k)$
$\{x_2(n)\}$ is even	$H_1(k)H_2(k)$
Both sequences are even	$H_1(k)H_2(k)$
$\{x_1(n)\}$ is odd	$H_1(k)H_2(-k)$
$\{x_2(n)\}$ is odd	$H_1(-k)H_2(k)$
Both sequences are odd	$- H_1(k)H_2(k)$

Whenever we need to perform linear convolution of two sequences, $\{x_1(n)\}$ of length N_1 and $\{x_2(n)\}$ of length N_2, $N_2 > N_1$, we add a string of zeros to both the sequences until the length of each becomes at least equal to $(N_1 + N_2 - 1)$, and then perform circular convolution on the resulting sequences.

3.4.9 Product Theorem

Let us consider two data sequences $\{x_1(n)\}$ and $\{x_2(n)\}$, each of length N. If their DFT sequences are $\{X_1(k)\}$ and $\{X_2(k)\}$ respectively, then the DFT of their product is given by the circular convolution of $X_1(k)$ and $X_2(k)$. That is,

$$F[x_1(n)x_2(n)] = (1/N)\ [X_1(k) * X_2(k)]$$

Using the result of (3.15) we can rewrite the r.., t hand side of the above equation as

$$(1/N)[X_1(k) * X_2(k)] = (1/N)[H_{1e}(k) - jH_{1o}(k)] * [H_{2e}(k) - jH_{2o}(k)]$$

Carrying out the multiplication and rearranging the above expression, we get

$$\begin{aligned}(1/N)[X_1(k) * X_2(k)] = (1/N)[&\{H_{1e}(k) * H_{2e}(k) - H_{1o}(k) * H_{2o}(k)\} \\ &-j\{H_{1e}(k) * H_{2o}(k) + H_{1o}(k) * H_{2e}(k)\}].\end{aligned}$$

Using the result of (3.16) we obtain the DHT of the product $\{x_1(n)x_2(n)\}$ as shown below.

$$H[x_1(n)x_2(n)] = (1/N)[H_{1e}(k)*\{H_{2e}(k) + H_{2o}(k)\}$$

$$- H_{1o}(k)*\{H_{2o}(k) - H_{2e}(k)\}].$$

Observing that

$$H_{1e}(k) = [H_1(k) + H_1(-k)]/2 \qquad H_{1o}(k) = [H_1(k) - H_1(-k)]/2,$$

$$H_{2e}(k) = [H_2(k) + H_2(-k)]/2 \qquad H_{2o}(k) = [H_2(k) - H_2(-k)]/2$$

we get

$$H[x_1(n)x_2(n)] = (1/2N)[H_1(k) * H_2(k) + H_1(-k) * H_2(k)$$
$$+ H_1(k) * H_2(-k) - H_1(-k) * H_2(-k)]$$

If there is any symmetry in one or both of the multiplying sequences, the above result simplifies. Following is a summary.

SYMMETRY	DHT OF THE PRODUCT
$\{x_1(n)$ is even$\}$	$(1/N)[H_1(k)H_2(k)]$
$\{x_2(n)$ is even$\}$	$(1/N)[H_1(k)H_2(k)]$
Both sequences are even	$(1/N)[H_1(k)H_2(k)]$
$\{x_1(n)$ is odd$\}$	$(1/N)[H_1(k)H_2(-k)]$
$\{x_2(n)$ is odd$\}$	$(1/N)[H_1(-k)H_2(k)]$
Both sequences are odd	$(1/N)[-H_1(k)H_2(k)]$

3.4.10 Cross-Correlation Theorem

Circular cross-correlation of two data sequences $\{x_1(n)\}$ and $\{x_2(n)\}$ is given by

$$y(n) = (1/N) \sum_{h=0}^{N-1} x_1(h)x_2(n + h), \quad n = 0, 1, \ldots, N - 1. \qquad (3.19)$$

Note the similarity between the circular cross-correlation and the circular convolution of two data sequences. The difference is that in the convolution operation, we flip one of the data sequences before carrying out the operation, whereas in cross-correlation we do not.
If the DFT sequences of $\{x_1(n)\}$ and $\{x_2(n)\}$ are $\{X_1(k)\}$ and $\{X_2(k)\}$ respectively, then the DFT of their cross-correlation is

$$F[\text{Cross correlation of } x_1(n) \text{ and } x_2(n)] = X_1^*(k)X_2(k)$$

Where $X_1^*(k)$ is the complex conjugate of $X_1(k)$.

Using the result of (3.15), we rewrite $X_1^*(k)X_2(k)$ in terms of the DHT of the cross correlation H(k).

$$X_1^*(k)X_2(k) = [H_{1e}(k) + jH_{1o}(k)][H_{2e}(k) - jH_{2o}(k)]$$

Where $H_{1e}(k)$, $H_{1o}(k)$, $H_{2e}(k)$, $H_{2o}(k)$ are the even- and odd- parts of the DHTs $H_1(k)$ and $H_2(k)$.

Expanding the right hand side, we get

$$X_1^*(k)X_2(k) = [H_{1e}(k)H_{2e}(k) + H_{1o}(k)H_{2o}(k)]$$
$$- j[H_{1e}(k)H_{2o}(k) - H_{1o}(k)H_{2e}(k)].$$

Using the result of (3.16), we obtain the DHT of the correlation as

$H[\text{cross-correlation of } x_1(n) \text{ and } x_2(n)]$

$$= H_{1e}(k)H_{2e}(k) + H_{1o}(k)H_{2o}(k) + H_{1e}(k)H_{2o}(k) - H_{1o}(k)H_{2e}(k).$$

Substituting for $H_{1e}(k)$, $H_{1o}(k)$, $H_{2e}(k)$, $H_{2o}(k)$ in terms of $H_1(k)$, $H_1(-k)$, $H_2(k)$, $H_2(-k)$ and then simplifying, we get

H[cross correlation of $x_1(n)$ and $x_2(n)$]

$$= (1/2)[H_1(k)H_2(k) + H_1(-k)H_2(k) - H_1(k)H_2(-k) + H_1(-k)H_2(-k)] \quad (3.20)$$

Symmetry in one or both of the sequences correlated simplifies the above result. Following is a summary:

SYMMETRY	DHT OF THE CROSS CORRELATION
$\{x_1(n)\}$ is even	$H_1(k)H_2(k)$
$\{x_2(n)\}$ is even	$H_1(-k)H_2(k)$
Both sequences are even	$H_1(k)H_2(k)$
$\{x_1(n)\}$ is odd	$-H_1(k)H_2(-k)$
$\{x_2(n)\}$ is odd	$H_1(k)H_2(k)$
Both sequences are odd	$H_1(k)H_2(k)$

When we need to perform linear cross-correlation of two data sequences, $\{x_1(n)\}$ of length N_1 and $\{x_2(n)\}$ of length N_2, we add a string of zeros to the data sequences until the length of each becomes atleast $(N_1 + N_2 - 1)$, and then perform a circular cross correlation as shown above.

3.4.11 Autocorrelation Theorem

Given an N-point sequence $\{x_1(n)\}$ its autocorrelation R_{xx} is defined as

$$R_{xx}(n) = (1/N) \sum_{h=0}^{N-1} x_1(h)x_1(n + h), \; n = 0, 1, \ldots, N - 1.$$

Observe that if $x_1(n) = x_2(n)$ in (3.19), we get the above equation for autocorrelation. Therefore, taking the result for the DHT of the cross-correlation for two sequences in (2.20), and making the substitution $H_1(k) = H_2(k)$, we obtain the DHT of autocorrelation of $\{x_1(n)\}$.

Therefore,

$$R_{xx}(k) = (1/2)[H_1^2(k) + H_1^2(-k)]. \qquad (3.21)$$

From the above equation it is clear that the DHT of the autocorrelation of a real sequence is a non-negative even function. From the properties of the DHT we know that if $x_1(n)$ is even, $H_1(k) = H_1(-k)$ and if $x_1(n)$ is odd, $H_1(k) = - H_1(-k)$. In either case (3.21) reduces to $H_1^2(k)$.

If we need to perform the linear autocorrelation of $x_1(n)$, we add zeros to the data sequence until its length becomes atleast $2N - 1$ and then perform the circular autocorrelation as described above on the resulting sequence.

3.4.12 Parseval's Theorem

If $\{x(p)\}$ is an N-point real-valued sequence, we know that its DHT is given by

$$H(k) = (1/N) \sum_{p=0}^{N-1} x(p)\mathrm{cas}(2\pi pk/N), \; k = 0, 1, \ldots, N - 1. \qquad (3.22)$$

By changing the index in the data array from p to q, we can again write the DHT of the data sequence as

$$H(k) = (1/N) \sum_{q=0}^{N-1} x(q)\text{cas}(2\pi qk/N), \quad k = 0, 1, \ldots, N-1. \quad (3.23)$$

Multiplying (3.22) and (3.23), we get

$$H^2(k) = (1/N)^2 \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} x(p)x(q)\text{cas}(2\pi pk/N)\text{cas}(2\pi qk/N),$$

$$k = 0, 1, \ldots, N-1.$$

Now, let us sum both sides of the above equation over N points.

$$\sum_{k=0}^{N-1} H^2(k) = (1/N)^2 \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} x(p)x(q) \sum_{k=0}^{N-1} \text{cas}(2\pi pk/N)\text{cas}(2\pi qk/N)$$

By the orthogonality property of the kernel, we have

$$\sum_{n=0}^{N-1} \text{cas}(2\pi pn/N)\text{cas}(2\pi qn/N) = N, \quad p = q$$

$$= 0, \quad \text{otherwise}$$

Hence

$$\sum_{k=0}^{N-1} H^2(k) = (1/N) \sum_{p=0}^{N-1} x^2(p).$$

Changing the index on the right hand side, we get

$$\sum_{k=0}^{N-1} H^2(k) = (1/N) \sum_{n=0}^{N-1} x^2(n).$$

The above equation is the Parseval's theorem for the DHT. Table 3.1 provides a summary of the properties of the DHT.

TABLE 3.1 PROPERTIES OF THE DHT

1.	Kernel = $\mathrm{cas}(2\pi mn/N)$		Orthogonal and periodic on N
2.	(Effect of periodicity of the kernel)		(Imposes periodicity on the data sequence and its DHT)
3.	DHT$\{x(n)\}$	=	$(\mathrm{Re}\{X(k)\} - \mathrm{Im}\{X(k)\}$, where $X(k)$ is the DFT of $x(n))$
4.	DFT$\{x(n)\}$	=	$(H_e(k) - jH_o(k)$, where $H(k)$ is the DHT of $x(n))$
5.	$x_1(n)$		$H_1(k)$
6.	$x_2(n)$		$H_2(k)$
7.	$x_1(n) = x_1(-n)$		$H_1(k) = H_1(-k)$
8.	$x_1(n) = - x_1(-n)$		$H_1(k) = - H_1(-k)$
9.	$x_1(n) + x_2(n)$		$H_1(k) + H_2(k)$
10.	$x_1(n + m)$		$[\cos(2\pi mk/N)H_1(k) - \sin(2\pi mk/N)H_1(-k)]$
11.	$[x_1(n)*x_2(n)]$		$(1/2)[H_1(k)H_2(k)+H_1(-k)H_2(k)+H_1(k)H_2(-k)-H_1(-k)H_2(-k)]$
12.	$x_1(n)x_2(n)$		$(1/2)[H_1(k)*H_2(k) + H_1(-k)*H_2(k) + H_1(k)*H_2(-k) + H_1(-k)*H_2(-k)]$
13.	{Cross-correlation of $x_1(n)$ and $x_2(n)$}		$(1/2)[H_1(k)H_2(k) + H_1(-k)H_2(k) - H_1(k)H_2(-k) + H_1(-k)H_2(-k)]$
14.	$R_{xx}\ \{x_1(n)\}$		$(1/2)[H_1^2(k) + H_1^2(-k)]$
15.	$(1/N) \sum_{n=0}^{N-1} x^2(n)$	=	$\sum_{k=0}^{N-1} H^2(k)$

4. RADIX-2 DECIMATION-IN-TIME FAST HARTLEY TRANSFORM ALGORITHM

4.1 Introduction

We know that the DHT of an N-point real sequence {x(n)} is given by

$$H(k) = (1/N) \sum_{n=0}^{N-1} x(n)\mathrm{cas}(2\pi nk/N), \quad k = 0, 1, \ldots, N - 1.$$

However, if we compute the DHT of the sequence directly, we need N multiplications and N additions to compute each coefficient in the DHT sequence, and a total of N^2 multiplications and additions to compute the N-point sequence. This is an extremely tall order for long data sequences. As such, the fast algorithms are a practical way out of this computational explosion. Bracewell [7] was the first to introduce a radix-2 decimation-in-time (DIT) fast algorithm to compute the DHT, but he did not exploit many inherent symmetries in the algorithm. Kwong and Shiu [8] proposed a more refined version of Bracewell's algorithm, which significantly reduced the number of multiplications and additions required to compute the DHT. Sorensen <u>et</u> <u>al</u>. [9] derived the same algorithm by an index mapping approach. This algorithm is the most commonly used, and also computationally the least complex of all the FHT algorithms.

However, the radix-2 algorithm can only be used with data sequences whose length is an integer power of two, because the algorithm involves successively splitting the data sequence into two equal half-length sequences. Any fast algorithm that computes the DHT of a given length of data sequence from smaller length DHTs by splitting the data into smaller sequences is called a DIT algorithm.

4.2 The Decomposition Formula

By exploiting the symmetry and the periodicity of the kernel cas($2\pi nk/N$), the DHT computation can be decomposed into successively smaller DHT computations.

Let us consider the DHT equation of an N-point real-valued sequence.

$$H(k) = (1/N) \sum_{n=0}^{N-1} x(n)\mathrm{cas}(2\pi nk/N), \quad k = 0, 1, \ldots, N - 1. \qquad (4.1)$$

Since N is an even integer (which is a essential for this algorithm), we can compute H(k) by separating {x(n)} into two (N/2)-point sequences, one consisting of the even-indexed samples and the other containing odd-indexed samples in {x(n)}. Stated mathematically,

$$H(k) = \sum_{n \text{ even}} x(n)\mathrm{cas}(2\pi nk/N) + \sum_{n \text{ odd}} x(n)\mathrm{cas}(2\pi nk/N)$$

By substituting the variables n = 2r for even n and n = 2r + 1 for odd n, we get

$$H(k) = \sum_{r=0}^{(N/2)-1} x(2r)\mathrm{cas}\{2\pi(2r)k/N\}$$

$$+ \sum_{r=0}^{(N/2)-1} x(2r + 1)\mathrm{cas}\{2\pi(2r + 1)k/N\}. \qquad (4.2)$$

However,

$$\begin{aligned}\mathrm{cas}\{2\pi(2r + 1)k/N\} &= \mathrm{cas}\{2\pi(2r)k/N + 2\pi k/N\}\\ &= \cos\{2\pi(2r)k/N + 2\pi k/N\} + \sin\{2\pi(2r)k/N + 2\pi k/N\}\\ &= \cos\{2\pi(2r)k/N\}\cos(2\pi k/N)\\ &- \sin\{2\pi(2r)k/N\}\sin(2\pi k/N)\\ &+ \sin\{2\pi(2r)k/N\}\cos(2\pi k/N)\\ &+ \cos\{2\pi(2r)k/N\}\sin(2\pi k/N).\end{aligned}$$

Rearranging the terms, we get

$$\begin{aligned}\mathrm{cas}\{2\pi(2r + 1)k/N\} &= \cos(2\pi k/N)\ \{\cos[2\pi(2r)k/N] + \sin[2\pi(2r)k/N]\}\\ &+ \sin(2\pi k/N)\ \{\cos[2\pi(2k)r/N] - \sin[2\pi(2k)r/N]\}.\\ &= \cos(2\pi k/N)\mathrm{cas}\{2\pi(2r)k/N\}\\ &+ \sin(2\pi k/N)\mathrm{cas}\{2\pi(2k)(-r)/N\}\\ &= \cos(2\pi k/N)\mathrm{cas}\{2\pi rk/(N/2)\}\\ &+ \sin(2\pi k/N)\mathrm{cas}\{2\pi k(-r)/(N/2)\}\end{aligned}$$

Substituting the above expression for cas{2π(2r + 1)k/N} in (4.2), we get

$$\begin{aligned}H(k) &= \sum_{r=0}^{(N/2-1)} x(2r)\mathrm{cas}\{2\pi rk/(N/2)\}\\ &+ \cos(2\pi k/N) \sum_{r=0}^{(N/2)-1} x(2r + 1)\mathrm{cas}\{2\pi rk/(N/2)\}.\\ &+ \sin(2\pi k/N) \sum_{r=0}^{(N/2)-1} x(2r + 1)\mathrm{cas}\{2\pi k(-r)/(N/2)\}.\end{aligned}$$

Recognizing that each of the above summations is an (N/2)-point DHT, we can rewrite the above equation as

$$\begin{aligned} H(k) &= H_{2r}(k) + \cos(2\pi k/N)H_{2r+1}(k) + \sin(2\pi k/N)H_{2r+1}(-k) \\ &= H_{2r}(k) + \cos(2\pi k/N)H_{2r+1}(k) + \sin(2\pi k/N)H_{2r+1}\{(N/2) - k\}, \\ & \qquad k = 0, 1, \ldots, (N-1). \end{aligned} \tag{4.3}$$

where

$H_{2r}(k)$ is the DHT of the (N/2)-point even-indexed samples and

$H_{2r+1}(k)$ is the DHT of the (N/2)-point odd-indexed samples. Since all of the above are (N/2)-point DHTs, they are all periodic on N/2.

Therefore, the steps involved in the computation of an N-point DHT are:

1. Split the data sequence into two (N/2)-point sequences, one consisting only of the even-indexed samples and the other consisting only of the odd-indexed samples.
2. Compute the DHTs of the two sequences separately. Call the DHT of the even-indexed samples $H_{2r}(k)$ and the DHT of the odd-indexed samples $H_{2r+1}(k)$.
3. Compute the sum in (4.3) to obtain the DHT of the N-point sequence.

However we can repeatedly apply Step 1 in performing Step 2, dividing each (N/2)-point sequence into two (N/4)-point sequences, and each (N/4)-point sequence into two (N/8)-point sequences and so on, until we reach the stage where we need to compute only 2-point DHTs, which require no multiplications. In the next section, the working of the algorithm is explained with application to a 16-point DHT.

4.3 Example Description of the Algorithm

Let us consider a 16-point data sequence {x(0), x(1), x(2), ..., x(15)}. Now, as Step 1 suggests in Sec. 4.2, let us rearrange the data points such that the first half of the array consists of the even-indexed samples and the second half consists of the odd-indexed samples. Before computing the DHTs of the two 8-point sequences above, let us continue splitting them, as in Step 1, until we reach the 2-point sequences. Table 4.1 on the next page explains the procedure.

The last column in Table 4.1 is the result of repeatedly applying Step 1 to the data sequence. If we observe closely, the index of any sample in the last column is obtained by taking the binary representation of the data sample index in the first column of the same row, and then reversing the bit string. For example, let us consider x(12) in the first column. The number 12 is represented as 1100 in binary form. If we reverse the digits, we obtain 0011, whose decimal equivalent is 3, and, as such, x(3) is the sample point found in the last column for the row containing x(12). Similarly, the number 14 is represented as 1110 in binary form, and, reversing it we obtain 0111, which is the binary equivalent of the number 7. That explains why we find x(14) in the first column, and x(7) in the last column of the same row in the above table.

Therefore, in the first step we scramble the data array such that it contains the samples in the bit reversed order. The scrambled data is shown as the input to the algorithm in Fig. (4.1).

From Eqn. (4.1), we see that the DHT of a two-point sequence {x(0), x(1)} is obtained by once adding them, and next subtracting one from the other i.e.,

$$H(0) = x(0) + x(1),$$

and

$$H(1) = x(0) - x(1).$$

TABLE 4.1 SCRAMBLING THE DATA FOR THE RADIX-2 DIT ALGORITHM

Original array	1st scramble	2nd scramble	3rd scramble
x(0)	x(0)	x(0)	x(0)
x(1)	x(2)	x(4)	x(8)
x(2)	x(4)	x(8)	x(4)
x(3)	x(6)	x(12)	x(12)
x(4)	x(8)	x(2)	x(2)
x(5)	x(10)	x(6)	x(10)
x(6)	x(12)	x(10)	x(6)
x(7)	x(14)	x(14)	x(14)
x(8)	x(1)	x(1)	x(1)
x(9)	x(3)	x(5)	x(9)
x(10)	x(5)	x(9)	x(5)
x(11)	x(7)	x(13)	x(13)
x(12)	x(9)	x(3)	x(3)
x(13)	x(11)	x(7)	x(11)
x(14)	x(13)	x(11)	x(7)
x(15)	x(15)	x(15)	x(15)

Computation of the 2-point DHTs is shown in Fig. (4.1). The broken lines in the figure indicate subtraction and the solid lines indicate addition. From these two-point DHTs, we compute the four-point DHTs in the second stage. Computation of a four-point DHT also does not need any multiplications.

In the third stage, we compute the 8-point DHTs from the 4-point DHTs using (4.3), bearing in mind that the top 4-point sequence always acts as $H_{2r}(k)$ and the bottom one as $H_{2r+1}(k)$. The internal structure of the boxes marked T_3 in Fig. (4.1) is shown separately in Fig. (4.2). The input to the boxes labeled T_3 is always the bottom sequence $H_{2r+1}(k)$. The boxes implement the summation $\cos(2\pi k/N)H_{2r+1}(k) + \sin(2\pi k/N)H_{2r+1}\{(N/2)-k)$ with $N = 8$, $k = 0, 1, \ldots, 3$.

Finally the 16-point sequence is obtained from the two 8-point sequences, with the top 8-point sequence acting as $H_{2r}(k)$ and the bottom 8-point sequence acting as $H_{2r+1}(k)$ in Eqn. (4.3). The box T_4 implements the summation

$\cos(2\pi k/N)H_{2r+1}(k) + \sin(2\pi k/N)H_{2r+1}\{(N/2)-k\}$, $N = 16$, $k = 0, \ldots, 7$

4.4 Computational Cost

Let us consider Eqn. (4.3).

$$H(k) = H_{2r}(k) + \cos(2\pi k/N)H_{2r+1}(k) + \sin(2\pi k/N)H_{2r+1}\{(N/2) - k\}$$

From the above equation we can form the following three equations.

$$H\{(N/2)-k\} = H_{2r}\{(N/2)-k\} - \cos(2\pi k/N)H_{2r+1}\{(N/2)-k\} + \sin(2\pi k/N)H_{2r+1}(k) \qquad (4.4)$$

$$H(k+N/2) = H_{2r}(k) - [\cos(2\pi k/N)H_{2r+1}(k) + \sin(2\pi k/N)H_{2r+1}\{(N/2)-k\}], \qquad (4.5)$$

$$H(N-k) = H_{2r}\{(N/2)-k\} + [\cos(2\pi k/N)H_{2r+1}\{(N/2)-k\} - \sin(2\pi k/N)H_{2r+1}(k). \qquad (4.6)$$

In arriving at the above equations, we used the fact that $H_{2r}(k)$ and $H_{2r+1}(k)$ are periodic on N/2.

By referring to (4.3), (4.4), (4.5), (4.6) we see that the basic products involved in those equations are only those that are shown in Table 4.2

TABLE 4.2 PRODUCTS REQUIRED TO COMPUTE AN N-POINT DHT USING THE RADIX-2 DIT ALGORITHM

$P1 = \cos(2\pi k/N)[H_{2r+1}(k)]$	$P2 = \cos(2\pi k/N)[H_{2r+1}\{(N/2)-k\}]$
$P3 = \sin(2\pi k/N)[H_{2r+1}(k)]$	$P4 = \sin(2\pi k/N)[H_{2r+1}\{(N/2)-k\}]$

By forming the above products, we observe that for each k we can compute a total of four points H(k), H{(N/2) - k}, H{(N/2) + k} and H(N - k). Thus, once we form the products necessary to compute H(0), H(1), ..., H{(N/4)}, we can compute the remaining (3N/4) - 1 coefficients in the DHT sequence without any further multiplications. Table 4.3 explains the fact clearly for a 16-point sequence.

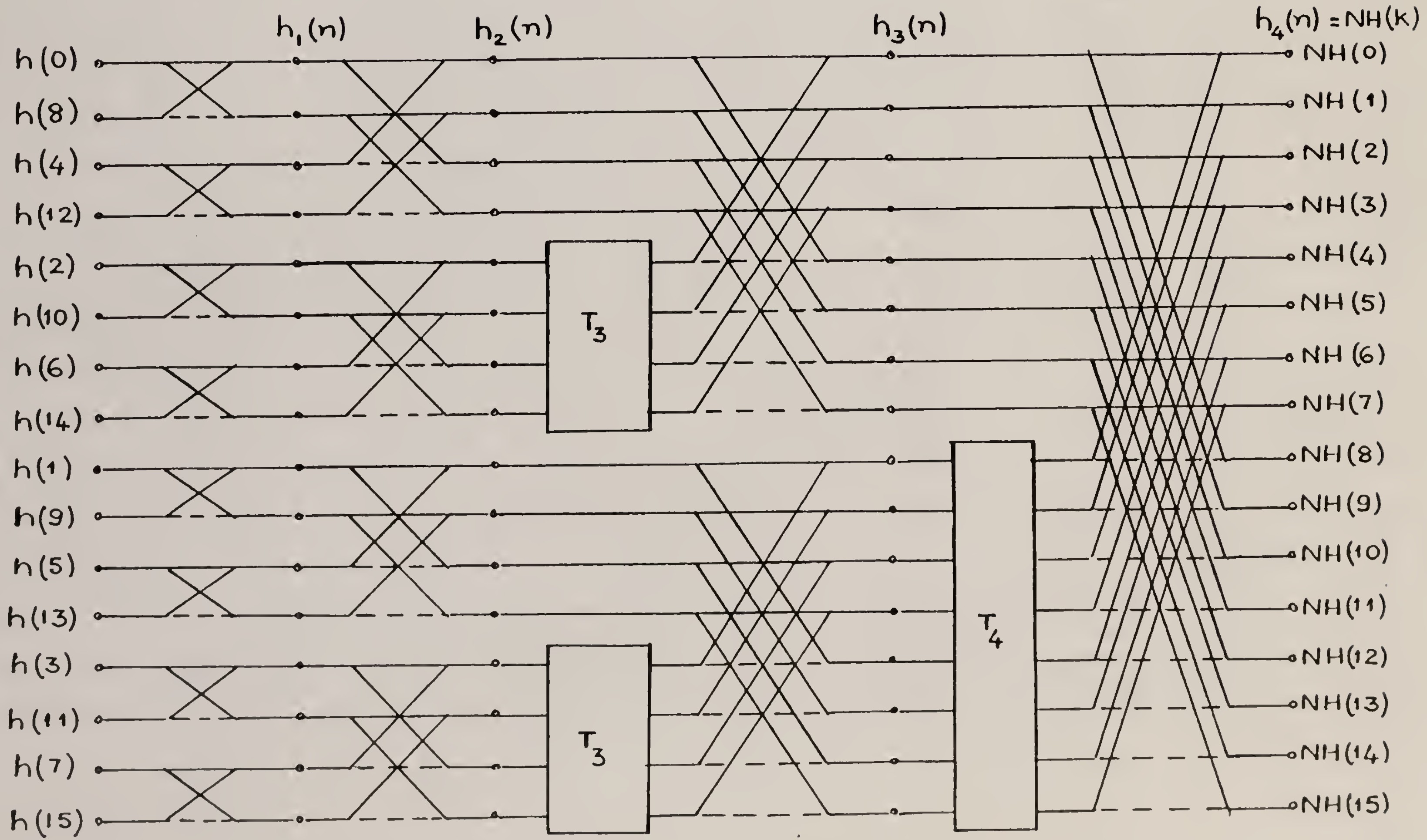

Fig. 4.1 Flowgraph of a 16-point radix-2 decimation-in-time FHT.

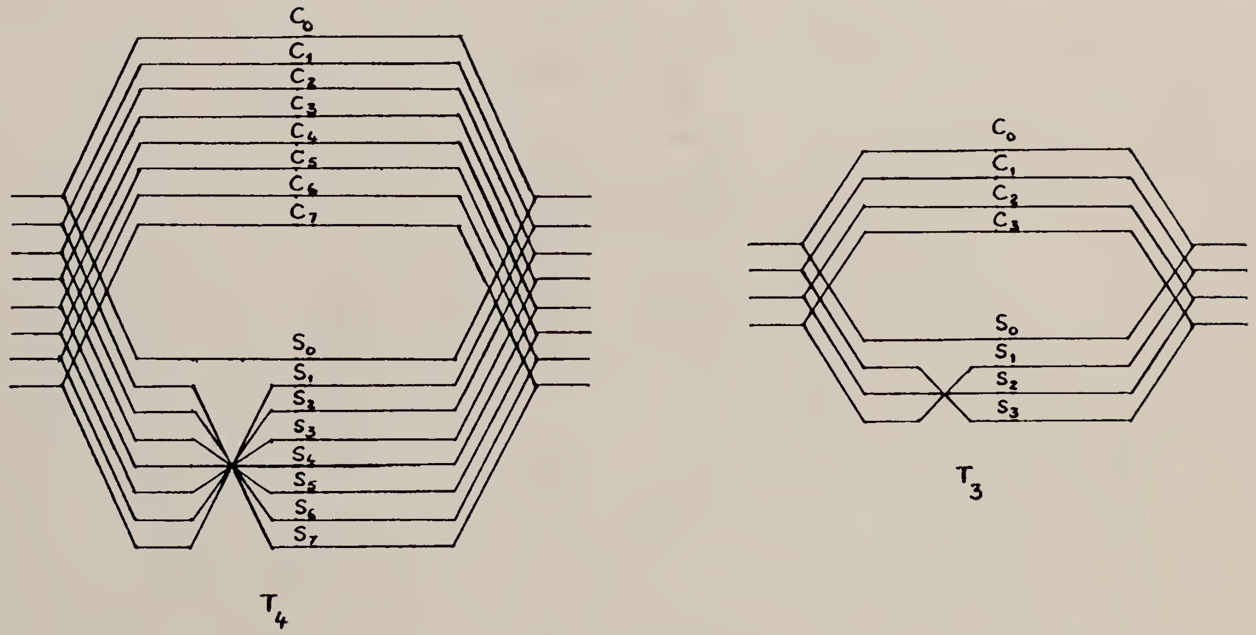

Fig. 4.2 Internal structure of the boxes T3 and T4.

TABLE 4.3 WORKING OF THE RADIX-2 DIT AGORITHM FOR A 16-POINT DHT

k	(N/2) - k	(N/2) + k	N - k
0		8	
1	7	9	15
2	6	10	14
3	5	11	13
4		12	

Thus, for a 16-point data we can compute the entire DHT sequence by forming the following products

$$P1,\ P2,\ P3,\ P4, \quad k = 0, 1, \ldots, 4$$

in Table 4.2, and then using them in Eqns. (4.4) - (4.6), instead of directly computing the coefficients using Eqn. (4.3).

From Table 4.2 it is clear that whenever $k = 0$ or $k = N/4$, the trigonometric terms in products P1 through P4 become either zero or one and as such, we do not need any multiplications for those two special cases. Avoiding these two special cases, we need to compute the four products

$$P1,\ P2,\ P3,\ P4, \quad k = 0, 1, \ldots, (N/4) - 1.$$

So for an N-point sequence we need $4[(N/4) - 1] = N - 4$ products.

Since we repeatedly apply Eqn. (4.3) in the algorithm by computing (N/2)-point DHTs from two (N/4)-point DHTs, and (N/4)-point DHTs from two (N/8)-point DHTs etc., we can reap the above mentioned advantages in multiplications at every stage. Therefore the total number of multiplications is given by

$M = (N - 4) + 2[(N/2) - 4] + 2^2[(N/2^2) - 4] + \dots + 2^{P-3}[N/(2^{P-3}) - 4]$

where

$P = \log_2 N.$

Simplifying the above expression, we get

$$\begin{aligned} M &= (P - 2)N - 4(1 + 2 + 2^2 + 2^3 + \dots + 2^{P-3}) \\ &= (P - 2)N - 4(2^{P-2} - 1) \\ &= (P - 2)N - N + 4 \\ &= PN - 3N + 4. \end{aligned}$$

Hence total number of multiplications is given by

$M = N \log_2 N - 3N + 4.$

<u>Additions</u>

Having formed the products

$$P1,\ P2,\ P3,\ P4, \quad k = 0, 1, \dots, (N/4) - 1,$$

let us form the following two summations

$$S1 = P1 + P4 \quad \text{and}$$

$$S2 = P3 - P2$$

Then (4.3), (4.4), (4.5), (4.6) can be rewritten as

$$\begin{aligned} H(k) &= H_{2r}(k) + S1, \\ H\{(N/2) + k\} &= H_{2r}(k) - S1, \\ H\{(N/2) - k\} &= H_{2r}\{(N/2) - k\} - S2, \\ H(N - k) &= H_{2r}\{(N/2) - k\} + S2. \end{aligned}$$

Therefore, in addition to the two summations S1 and S2, we need the above four summations, with k taking values from 1 through (N/4) - 1), to compute the entire N-point DHT sequence. Thus a total of $6[(N/4) - 1] = (3N/2) - 6$ additions are needed. But for the two special cases of $k = 0$ and $k = N/4$, we need only two additions each instead of the usual six as explained below.

Substituting $k = 0$ and $k = (N/4)$ in (4.3), (4.4), (4.5) and (4.6), we get

$$H(0) = H_{2r}(0) + H_{2r+1}(0)$$

and

$$H(N/2) = H_{2r}(0) - H_{2r+1}(0),$$

$$H(N/4) = H_{2r}(N/4) + H_{2r+1}(N/4),$$

and

$$H(3N/4) = H_{2r}(N/4) - H_{2r+1}(N/4).$$

Therefore, total number of additions A is given by

$$A = (3N/2) - 6 + 4 = (3N/2) - 2.$$

Since we continue splitting the data sequence into half-length sequences until we reach 2-point sequences, we can obtain the above savings in additions at every stage. Therefore, total number of additions A is given by

$$A = [(3N/2) - 2] + 2[(3/2)(N/2) - 2] + \dots + 2^{P-2}[(3/2)(N/2^{P-2}) - 2]$$

where

$$P = \log_2 N.$$

$$A = (3N/2)(P - 1) - 2[1 + 2 + 2^2 + \dots + 2^{P-2}]$$

$$= (3N/2)(P - 1) - 2[2^{P-1} - 1]$$

$$= (3N/2)(P - 1) - N + 2.$$

The above is the number of additions needed until we reach 4-point sequences. In addition, we have (N/2) two-point sequences, which need N more additions

$$A = (3N/2)P - (3N/2) - N + 2 + N$$

$$= (3N/2)\log_2 N - (3N/2) + 2.$$

5. RADIX-2 DECIMATION-IN-FREQUENCY FAST HARTLEY TRANSFORM ALGORITHM

5.1 Introduction

Decimation-in-time (DIT) and decimation-in-frequency (DIF) algorithms differ in one significant aspect. DIT algorithms compute the DHTs from short length DHTs and DIF algorithms compute the DHTs as short length DHTs. That is, when we compute the DHT of a 16-point sequence using DIT algorithm, we first compute the 2-point, 4-point, 8-point DHTs and finally by combining the two 8-point DHTs, we obtain the 16-point DHT. In a DIF algorithm, the task of computing a 16-point DHT is progrssively reduced to that of computing only 2-point DHTs, without ever actually computing the intermediate 8-point and 4-point DHTs. Computationally, a DIF algorithm does not have any advantage over the corresponding DIT algorithm. A radix-2 DIF algorithm can be used only with data sequences whose length is an integral power of two. Meckelburg and Lipka [10] first introduced the radix-2 DIF FHT algorithm. Sorensen et al. [9] derived the same algorithm by an index mapping approach.

5.2 The Decomposition Formula

The DHT H(k) of an N-point real data sequence {x(n)} is given by

$$H(k) = (1/N) \sum_{n=0}^{N-1} x(n)\mathrm{cas}(2\pi nk/N), \quad k = 0, 1, \ldots, N-1. \qquad (5.1)$$

We can divide {x(n)} into two half-length sequences and rearrange the above equation as

$$H(k) = (1/N) \sum_{n=0}^{(N/2)-1} x(n)\text{cas}(2\pi nk/N)$$

$$+ (1/N) \sum_{n=N/2}^{(N/2)-1} x(n)\text{cas}(2\pi nk/N), \quad k = 0, 1, \ldots, N - 1.$$

or

$$H(k) = (1/N) \sum_{n=0}^{(N/2)-1} x(n)\text{cas}(2\pi nk/N)$$

$$+ (1/N) \sum_{n=0}^{(N/2)-1} x(n + N/2)\text{cas}\{2\pi k(n + N/2)/N\}, \quad k = 0, 1, \ldots, N - 1.$$

$$H(k) = (1/N) \sum_{n=0}^{(N/2)-1} x(n)\text{cas}(2\pi nk/N)$$

$$+ (1/N) \sum_{n=0}^{(N/2)-1} x(n + N/2)\text{cas}[k\pi + 2\pi nk/N), \quad k = 0, 1, \ldots, N - 1. \quad (5.2)$$

Now, let us compute the even- and odd- indexed DHT samples separately. Substituting k = 2r in (5.2), we obtain the even-indexed samples

$$H(2r) = (1/N) \sum_{n=0}^{(N/2)-1} x(n)\text{cas}\{2\pi n(2r)/N\}$$

$$+ (1/N) \sum_{n=0}^{(N/2)-1} x(n + N/2)\text{cas}[2r\pi + 2\pi n(2r)/N], \quad r = 0, 1, \ldots, (N/2) - 1.$$

Recognizing that

$\text{cas}[2r\pi + 2\pi n(2r)/N] = \text{cas}(2\pi n(2r)/N)$

we can rewrite the above equation as

$$H(2r) = (1/N) \sum_{n=0}^{(N/2)-1} [x(n) + x(n + N/2)]\text{cas}\{2\pi n(2r)/N)\}$$

$$= (1/N) \sum_{n=0}^{(N/2)-1} [x(n) + x(n + N/2)]\text{cas}[2\pi nr/(N/2)],$$

$$r = 0, 1, \ldots, (N/2) - 1. \qquad (5.3)$$

Since the above is the expression for an (N/2)-point DHT, we conclude that the even indexed samples of the DHT can be computed as an (N/2)-point DHT.

Substituting $k = 2r + 1$ in (5.2), we obtain the odd-indexed samples of the DHT sequence.

$$H(2r + 1) = (1/N) \sum_{n=0}^{(N/2)-1} x(n)\text{cas}[2\pi n(2r + 1)/N]$$

$$+ (1/N) \sum_{n=0}^{(N/2)-1} x(n + N/2)\text{cas}[(2r + 1)\pi + 2\pi n(2r + 1)/N]. \qquad (5.4)$$

Using the fact that

$$\text{cas}[(2r + 1)\pi + 2\pi n(2r + 1)/N] = \text{cas}[\pi + 2\pi n(2r + 1)/N]$$
$$= - \text{cas}[2\pi n(2r + 1)/N]$$

we can rewrite (5.4) as

$$H(2r + 1) = (1/N) \sum_{n=0}^{(N/2)-1} x(n)\text{cas}[2\pi n(2r + 1)/N]$$

$$- (1/N) \sum_{n=0}^{(N/2)-1} x(n + N/2)\text{cas}[2\pi n(2r + 1)/N],$$

$$r = 0, 1, \ldots, (N/2) - 1. \qquad (5.5)$$

$$= (1/N) \sum_{n=0}^{(N/2)-1} [x(n) - x(n + N/2)]\mathrm{cas}[2\pi n(2r + 1)/N]. \qquad (5.6)$$

However

$$\mathrm{cas}[2\pi n(2r + 1)/N] = \cos(2\pi n/N)\mathrm{cas}(2\pi n(2r)/N) + \sin(2\pi n/N)\mathrm{cas}[2\pi(2r)(-n)/N]$$

Substituting the above result in (5.6), we get

$$H(2r + 1) = (1/N) \sum_{n=0}^{(N/2)-1} [x(n) - x(n + N/2)]\cos(2\pi n/N)\mathrm{cas}(2\pi nr/(N/2)$$

$$+ (1/N) \sum_{n=0}^{(N/2)-1} [x(n) - x(n+N/2)]\sin(2\pi n/N)\mathrm{cas}[2\pi r(-n)/(N/2)] \qquad (5.7)$$

Reversing the direction of the second summation in (5.7), we get

$$H(2r+1) = (1/N) \sum_{n=0}^{(N/2)-1} \{[x(n) - x(n + N/2)]\cos(2\pi n/N)$$

$$+ [x\{(N/2) - n\} - x(N - n)]\sin(2\pi n/N)\}\mathrm{cas}\{2\pi nr/(N/2)\} \qquad (5.8)$$

The above is an (N/2)-point DHT to compute the odd-indexed samples of the DHT.

Hence from (5.3) and (5.8) we see that an N-point DHT can be computed as two (N/2)-point DHTs. The following are the steps involved in the computation of the DHT of an N-point sequence $\{x(n)\}$.

1. First, form the N/2-point sequences $\{g(n)\}$ and $\{h(n)\}$, where $g(n)$ and $h(n)$ are given by

   $g(n) = [x(n) + x(n + N/2)]$, $n = 0, 1, \ldots, (N/2) - 1$,

   $h(n) = [x(n) - x(n + N/2)]$, $n = 0, 1, \ldots, (N/2) - 1$.

2. Then compute the (N/2)-point sequence

$$f(n) = h(n)\cos(2\pi n/N) + h\{(N/2) - n\}\sin(2\pi n/N),$$
$$n = 0, 1, \ldots, (N/2) - 1.$$

3. Compute the (N/2)-point DHTs of the sequences g(n) and f(n).

   The DHT of g(n) gives the even-indexed samples, and the DHT of f(n) gives the odd-indexed samples of H(k).

However, in performing Step 3 we can apply Steps 1 and 2 to the computation of those (N/2)-point DHTs repeatedly, until we need to compute only 2-point DHTs. It is this repeated application of Steps 1 and 2 that makes the computation of the DHT faster. The working of the algorithm for an 8-point data sequence is explained in the next section.

5.3 Example Description of the Algorithm

Fig. (5.1) shows the flow diagram for an 8-point FHT. Observe that in Fig. (5.1), the data on the input side of the flow diagram is in the normal order. We do not need to scramble the data in the beginning as we did with the DIT radix-2 algorithm. In the diagram, whenevr two solid lines meet at a point, it corresponds to an addition and when a solid line and a broken line meet, it defines subtraction.

As suggested in Step 1 in the previous section, we first form the 4-point sequences {g(n)} and {h(n)} as follows:

$$g(n) = x(n) + x(n + 4),\ n = 0, 1, \ldots, 3$$

$$h(n) = x(n) - x(n + 4),\ n = 0, 1, \ldots, 3$$

The box marked T_3 achieves the computation of f(n). The internal structure of T_3 is shown separately. The box T_3 takes the sequence h(n) as its input and computes f(n) as

$$f(n) = h(n)\cos(2\pi n/N) + h(4 - n)\sin(2\pi n/N),\ N = 8,\ n = 0, \ldots, 3.$$

According to Step 3 in the previous section, we now compute the DHTs of g(n) and f(n). But we can apply Steps 1 and 2 to those two sequences one more time, leading us to the computation of only 2-point DHTs. We observe that the DHT coefficients are in the bit reversed order. We have to perform a bit reversal operation, as described in the previous chapter, to get the coefficients back in the normal order. In this example, we have only three stages involved, reduction of an 8-point computation to two 4-point DHTs, and 4-point DHTs to 2-point DHTs and finally the computation of 2-point DHTs. In general, for a data sequence of length $N = 2^P$, we will have P iterations in the DHT computation process. Like the DIT radix-2 algorithm in the last chapter, this DIF algorithm also is an in-place algorithm. Therefore, once we compute g(n) and f(n) from the original 8-point data in the first iteration, we can replace the data elements with g(n) and f(n) in the data array. We keep doing this in all the iterations until we compute the required 2-point DHTs in the final iteration at which point the original data elements in the array will have been replaced by the DHT coefficients.

5.4 Computational Cost

We stated earlier that from the N data elements in {x(n)}, we form the (N/2)-point sequences {g(n)} and {h(n)}, and from h(n) the sequence f(n). The sequence f(n) is given by

$$f(n) = h(n)\cos(2\pi n/N) + h\{(N/2) - n\}\sin(2\pi n/N),\ n = 0, \ldots, (N/2) - 1.$$

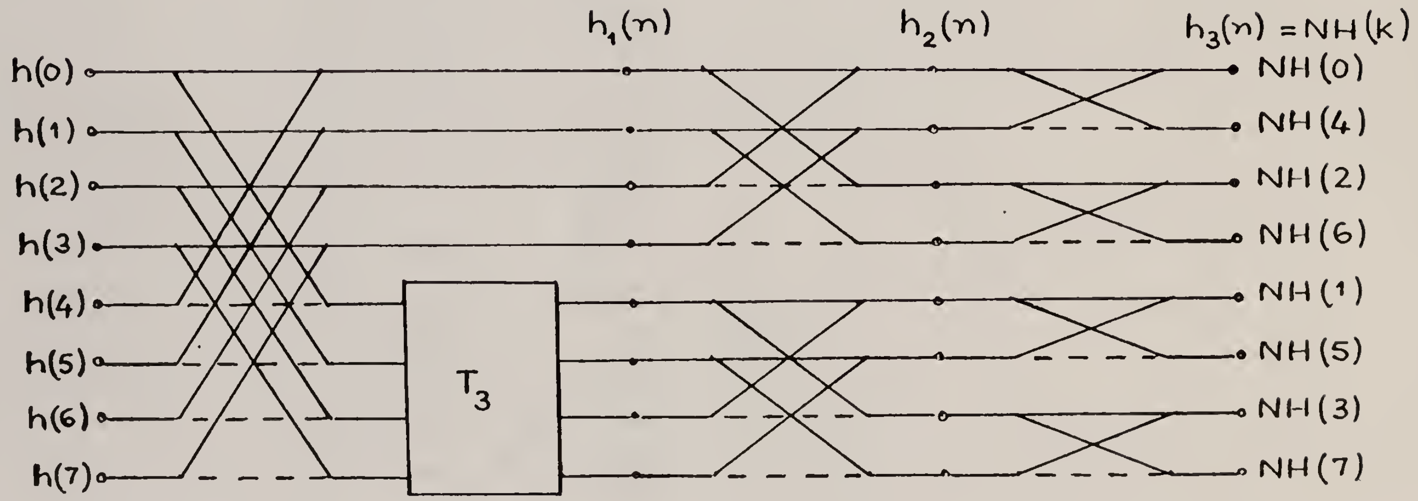

Fig. 5.1 Flow diagram of a 8-point decimation-in-frequency FHT.

Thus, to obtain a point $f(n)$ we need two multiplications

1. $h(n)\cos(2\pi n/N)$
2. $h\{(N/2)-n\}\sin(2\pi n/N)$.

Since $\{f(n)\}$ contains a total of $(N/2)$ points, we thus need a total of $2(N/2) = N$ multiplications. However, whenever $n = 0$ or $n = N/4$, the trigonometric functions in the above two multiplications reduce to either zero or one, and, as such, we can avoid those multiplications. Therefore, those two special coefficients save us a total of four multiplications. Hence, the number of multiplications required to obtain the $(N/2)$-point sequence $\{f(n)\}$ from the N-point data is $(N-4)$. These savings in multiplications can be obtained in all the iterations. Adding all such multiplications until we reach 4-point DHTs (wherefrom we do not need any multiplications), the total count M is given by

$$M = (N-4) + 2[(N/2) - 4] + 2^2[(N/2^2) - 4] + \ldots + 2^{P-3}[(N/2^{P-3}) - 4]$$

where $P = \log_2 N$.

Simplifying further, we get

$$\begin{aligned} M &= (P - 2)N - 4[1 + 2 + 2^2 + \ldots + 2^{P-3}] \\ &= (P - 2)N - 4[2^{P-2}-1] \\ &= (P - 2)N + 4 - N \qquad (\text{since } 2^P = N) \\ &= PN - 3N + 4 \\ M &= N\log_2 N - 3N + 4. \end{aligned}$$

Let us consider the number of additions. From the original data sequence $\{x(n)\}$, we first form $g(n)$ and $h(n)$ as shown below.

$g(n) = [x(n) + x(N + N/2)],\ n = 0, 1, \ldots, (N/2) - 1$

$h(n) = [x(n) - x(n + N/2)], \; n = 0, 1, \ldots, (N/2) - 1.$

We need (N/2) additions to form $\{g(n)\}$ and (N/2) additions to form $\{h(n)\}$, totaling N additions. Besides, in the computation of f(n), we come across additions of the following type:

$f(n) = h(n)\cos(2\pi n/N) + h[(N/2) - n]\sin(2\pi n/N), \; n = 0, \ldots, (N/2) - 1.$

We thus need (N/2) more additions to compute $\{f(n)\}$. However, when $n = 0$ or $n = (N/4)$, the trigonometric functions in the above addition become either zero or one and, as such, we do not need any additions in evaluating f(0) and f(N/4), thus saving us two additions in computing $\{f(n)\}$. Therefore, to compute $\{f(n)\}$ we need $[(N/2) - 2]$ additions. Hence, the total number of additions in computing $\{g(n)\}$, $\{h(n)\}$, $\{f(n)\}$ are given by

$$\begin{aligned} A1 &= (N/2) + (N/2) + (N/2) - 2 \\ &= (3N/2) - 2 \end{aligned}$$

Summing such additions over all the iterations until we reach 2-point DHTs, the total number of additions A is given by

$$\begin{aligned} A &= [(3N/2) - 2] + 2[(3/2)(N/2) - 2] \\ &\quad + 2^2[(3/2)(N/2^2) - 2] + \ldots + 2^{P-2}[(3/2)(N/2^{P-2}) - 2]. \\ &= (3N/2)(P - 1) - 2[1 + 2 + 2^2 + \ldots + 2^{P-2}] \\ &= (3N/2)(P - 1) - 2[2^{P-1} - 1] \\ &= (3N/2)(P - 1) - N + 2. \qquad (\text{since } 2^P = N) \end{aligned}$$

In addition, we will have (N/2) two-point DHTs to compute, thus requiring $2(N/2) = N$ more additions.

Thus, number of additions is given by

$$A = (3N/2)(P - 1) - N + 2 + N$$

$$= (3N/2)P - 3N/2 + 2$$

$$= (3N/2)\log_2 N - 3N/2 + 2.$$

The above count for additions and multiplications is the same as that for the DIT radix-2 algorithm discussed in the previous chapter.

6. RADIX-4 DECIMATION-IN-TIME FAST HARTLEY TRANSFORM ALGORITHM

6.1 Introduction

In the previous two chapters, we discussed the radix-2 algorithms which can be used to compute the DHT of a sequence whose length is an integer power of two. In addition to being even, if the length of a sequence is an integer power of four, we can use a radix-4 algorithm to compute its DHT more efficiently. In a DIT radix-4 algorithm, we split the data sequence into four equal-length smaller data sequences, compute the DHTs of those sequences, and finally combine them to obtain the DHT of the original data. Sorensen _et al_. [9] derived a radix-4 algorithm by an index mapping approach. Prado [11] also presented a radix-4 algorithm, which takes more number of operations than that of Sorensen _et al_. In this chapter, we derive the radix-4 algorithm from the definition of the DHT, without relying on the index mapping approach, describe its working with an example, and finally obtain its computational cost.

6.2 The Decomposition Formula

The DHT of an N-point real data sequence $\{x(n)\}$ is given by

$$H(k) = (1/N) \sum_{n=0}^{N-1} x(n)\mathrm{cas}(2\pi nk/N), \quad k = 0, 1, \ldots, N-1. \qquad (6.1)$$

Since the length of $\{x(n)\}$ is an integer power of four (which is a precondition for using this algorithm), we can split the summation in the above equation into four equal-length smaller summations as shown below. Since the factor (1/N) is only a constant outside the summation, we need not carry it in the derivation process until we reach the final step.

$$H(k) = \sum_{n=0}^{(N/4)-1} x(4n)\text{cas}[2\pi(4n)k/N] + \sum_{n=0}^{(N/4)-1} x(4n+1)\text{cas}[2\pi(4n+1)k/N]$$

$$+ \sum_{n=0}^{(N/4)-1} x(4n+2)\text{cas}[2\pi(4n+2)k/N]$$

$$+ \sum_{n=0}^{(N/4)-1} x(4n+3)\text{cas}[2\pi(4n+3)k/N]. \qquad (6.2)$$

Observing that

$$\text{cas}[2\pi(4n+1)k/N] = \cos[2\pi(4n+1)k/N] + \sin[2\pi(4n+1)k/N],$$

and then using the familiar trigonometric identities, we get

$$\text{cas}[2\pi(4n+1)k/N] = \cos(2\pi 1k/N)\text{cas}[2\pi nk/(N/4)]$$
$$+ \sin(2\pi 1k/N)\text{cas}[2\pi n(-k)/(N/4)]. \qquad (6.3)$$

Using the result of (6.3) in (6.2) with 1 = 1, 2, 3, we get

$$H(k) = \sum_{n=0}^{(N/4)-1} x(4n)\text{cas}[2\pi nk/(N/4)]$$

$$+ \cos(2\pi k/N) \sum_{n=0}^{(N/4)-1} x(4n+1)\text{cas}[2\pi nk/(N/4)]$$

$$+ \sin(2\pi k/N) \sum_{n=0}^{(N/4)-1} x(4n+1)\text{cas}[2\pi n(-k)/(N/4)]$$

$$+ \cos[2\pi(2k)/N] \sum_{n=0}^{(N/4)-1} x(4n+2)\text{cas}[2\pi nk/(N/4)]$$

$$+ \sin[2\pi(2k)/N] \sum_{n=0}^{(N/4)-1} x(4n+2)\text{cas}[2\pi n(-k)/(N/4)]$$

$$+ \cos[2\pi(3k)/N] \sum_{n=0}^{(N/4)-1} x(4n+3)\text{cas}[2\pi nk/(N/4)]$$

$$+ \sin[2\pi(3k)/N] \sum_{n=0}^{(N/4)-1} x(4n+3)\text{cas}[2\pi n(-k)/(N/4)],$$

$$k = 0, 1, \ldots, (N/4) - 1. \qquad (6.4)$$

Bringing back the (1/N) factor and recognizing that

$$(1/N) \sum_{n=0}^{(N/4)-1} x(4n+1)\text{cas}[2\pi n(-k)/(N/4)], \quad k = 0, 1, \ldots, (N/4) - 1$$

is an (N/4)-point DHT, and indicating it as H_{4n+1}, we can rewrite (6.4) as

$$H(k) = [H_{4n}(k) + \cos(2\pi k/N)H_{4n+1}(k) + \sin(2\pi k/N)H_{4n+1}(-k)$$
$$+ \cos[2\pi(2k)/N)H_{4n+2}(k) + \sin[2\pi(2k)/N)H_{4n+2}(-k)$$
$$+ \cos[2\pi(3k)/N)H_{4n+3}(k) + \sin[2\pi(3k)/N)H_{4n+3}(-k)] \qquad (6.5)$$

Eqn. (6.5) is the decomposition formula for the DIT radix-4 algorithm. Thus, H(k) is computed from four (N/4)-point DHTs - $H_{4n}(k)$, $H_{4n+1}(k)$, $H_{4n+2}(k)$ and $H_{4n+3}(k)$. The following are the steps involved in the algorithm.

1. Split the data sequence {x(n)} into four (N/4)-point sequences

   {x(4n)}, {x(4n + 1)}, {x(4n + 2)}, {x(4n + 3)},

   $n = 0, 1, \ldots, (N/4) - 1.$

2. Compute independently the (N/4)-point DHTs

   $H_{4n}(k)$, $H_{4n+1}(k)$, $H_{4n+2}(k)$ and $H_{4n+3}(k)$, $k = 0, 1, \ldots, (N/4) - 1$

   respectively, of the sequences formed in Step 1.

3. Using (6.5) combine the above (N/4)-point DHTs to obtain the N-point DHT.

6.3 Example Description of the Working of the Algorithm

Let us consider a 64-point sequence {x(n)}. As Step 1 suggests in Sec. 6.2, let us divide it into four 16-point sequences as shown below.

{x(4n)} = {x(0), x(4), x(8), ..., x(60)}

{x(4n + 1)} = {x(1), x(5), x(9), ..., x(61)}

{x(4n + 2)} = {x(2), x(6), x(10), ..., x(62)}

{x(4n + 3)} = {x(3), x(7), x(11), ..., x(63)}.

Now, according to Step 2, we have to compute the DHTs of the above 16-point sequences. But in doing so, we can again apply Step 1, thus splitting each 16-point sequence into four 4-point sequences. The division of {x(4n)} results in the following:

{x(16n)} = {x(0), x(16), x(32), x(48)}

{x(16n + 4)} = {x(4), x(20), x(36), x(52)}

{x(16n + 8)} = {x(8), x(24), x(40), x(56)}

{x(16n + 12)} = {x(12), x(28), x(44), x(60)}

Similarly when we split the second 16-point sequence $\{x(4n + 1)\}$, we obtain the following four-point sequences.

$\{x(16n + 1)\} = \{x(1), x(17), x(33), x(49)\}$

$\{x(16n + 5)\} = \{x(5), x(21), x(37), x(53)\}$

$\{x(16n + 9)\} = \{x(9), x(25), x(41), x(57)\}$

$\{x(16n + 13)\} = \{x(13), x(29), x(45), x(61)\}$.

When we split the third 16-point sequence $\{x(4n + 2)\}$ into 4-point sequences, we obtain

$\{x(16n + 2)\} = \{x(2), x(18), x(34), x(50)\}$

$\{x(16n + 6)\} = \{x(6), x(22), x(38), x(54)\}$

$\{x(16n + 10)\} = \{x(10), x(26), x(42), x(58)\}$

$\{x(16n + 14)\} = \{x(14), x(30), x(46), x(62)\}$

And finally the division of the last 16-point sequence $\{x(4n+3)\}$ results in the following four sequences.

$\{x(16n + 3)\} = \{x(3), x(19), x(35), x(51)\}$

$\{x(16n + 7)\} = \{x(7), x(23), x(39), x(55)\}$

$\{x(16n + 11)\} = \{x(11), x(27), x(43), x(59)\}$

$\{x(16n + 15)\} = \{x(15), x(31), x(47), x(63)\}$.

Therefore, the first thing that needs to be done is to scramble the data array such that it contains the data elements in the above order of rows. Some kind of scrambling of the input data is the characteristic of any DIT algorithm.

Once we scrambled the data array, the first stage in the computation process is to compute the DHTs of all the above 4-point sequences, and this does not require any multiplications.

Now that we have all the necessary sixteen 4-point DHTs, we combine them using (6.5) with $N = 4^2$ and $k = 0, 1, \ldots, 15$ to obtain the four 16-point DHTs in the second stage.

Once we have the four 16-point DHTs, we compute the 64-point DHT from them using (6.5), with $N = 4^3$ and $k = 0, 1, \ldots, 63$ in the third and final stage.

Observe that in the second and third stages above, the value of N in (6.5) is always an integer power of four, and the value of the exponent is same as the iteration number. In total, we have three iterations - computation of 4-point, 16_point and 64-point DHTs - in the evaluation of a 64-point DHT. For a data sequence of length 4^P, we will have P stages in the evaluation of the DHT as shown in Fig. (6.1), where $P = 4$.

6.4 Computational Cost

From (6.5) we generate the following seven equations.

$$\begin{aligned} H(k + N/4) = {} & H_{4n}(k) - \sin(2\pi k/N)H_{4n+1}(k) + \cos(2\pi k/N)H_{4n+1}(-k) \\ & - \cos[2\pi(2k)/N]H_{4n+2}(k) - \sin[2\pi(2k)/N]H_{4n+2}(-k) \\ & + \sin[2\pi(3k)/N]H_{4n+3}(k) - \cos[2\pi(3k)/N]H_{4n+3}(-k) \end{aligned} \quad (6.6)$$

$$\begin{aligned} H(k + N/2) = {} & H_{4n}(k) - \cos(2\pi k/N)H_{4n+1}(k) - \sin(2\pi k/N)H_{4n+1}(-k) \\ & + \cos[2\pi(2k)/N]H_{4n+2}(k) + \sin[2\pi(2k)/N]H_{4n+2}(-k) \\ & - \cos[2\pi(3k)/N]H_{4n+3}(k) - \sin[2\pi(3k)/N]H_{4n+3}(-k) \end{aligned} \quad (6.7)$$

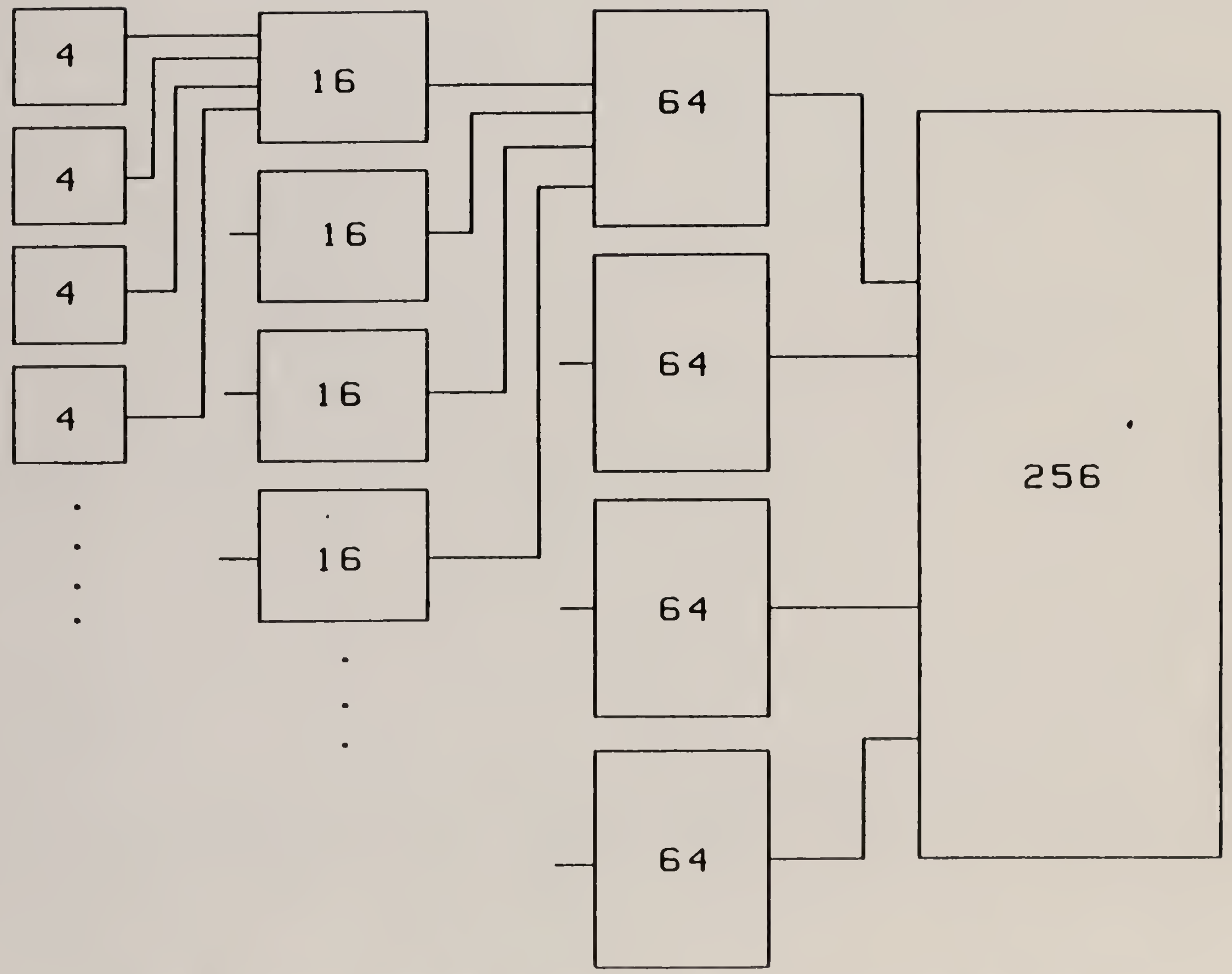

Fig. 6.1 Block diagram of the working of a 256-point radix-4 FHT.

$$\begin{aligned} H(k + 3N/4) = {} & H_{4n}(k) + \sin(2\pi k/N)H_{4n+1}(k) - \cos(2\pi k/N)H_{4n+1}(-k) \\ & - \cos[2\pi(2k)/N]H_{4n+2}(k) - \sin[2\pi(2k)/N]H_{4n+2}(-k) \\ & - \sin[2\pi(3k/N]H_{4n+3}(k) + \cos[2\pi(3k)/N]H_{4n+3}(-k) \end{aligned} \tag{6.8}$$

$$\begin{aligned} H[(N/4) - k] = {} & H_{4n}(-k) + \cos(2\pi k/N)H_{4n+1}(k) + \sin(2\pi k/N)H_{4n+1}(-k) \\ & + \sin[2\pi(2k)/N]H_{4n+2}(k) - \cos[2\pi(2k)/N]H_{4n+2}(-k) \\ & - \cos[2\pi(3k)/N]H_{4n+3}(k) - \sin[2\pi(3k)/N]H_{4n+3}(-k) \end{aligned} \tag{6.9}$$

$$\begin{aligned} H[(N/2) - k] = {} & H_{4n}(-k) + \sin(2\pi k/N)H_{4n+1}(k) - \cos(2\pi k/N)H_{4n+1}(-k) \\ & - \sin[2\pi(2k)/N]H_{4n+2}(k) + \cos[2\pi(2k)/N]H_{4n+2}(-k) \\ & + \sin[2\pi(3k)/N]H_{4n+3}(k) - \cos[2\pi(3k)/N]H_{4n+3}(-k) \end{aligned} \tag{6.10}$$

$$\begin{aligned} H[(3N/4) - k] = {} & H_{4n}(-k) - \cos(2\pi k/N)H_{4n+1}(k) - \sin(2\pi k/N)H_{4n+1}(-k) \\ & + \sin[2\pi(2k)/N]H_{4n+2}(k) - \cos[2\pi(2k)/N]H_{4n+2}(-k) \\ & + \cos[2\pi(3k)/N]H_{4n+3}(k) + \sin[2\pi(3k)/N]H_{4n+3}(-k) \end{aligned} \tag{6.11}$$

$$\begin{aligned} H(N - k) = {} & H_{4n}(-k) + \cos(2\pi k/N)H_{4n+1}(-k) - \sin(2\pi k/N)H_{4n+1}(k) \\ & - \sin[2\pi(2k)/N]H_{4n+2}(k) + \cos[2\pi(2k)/N]H_{4n+2}(-k) \\ & - \sin[2\pi(3k)/N]H_{4n+3}(k) + \cos[2\pi(3k)/N]H_{4n+3}(-k), \\ & k = 0, 1, \ldots, (N/8) - 1. \end{aligned} \tag{6.12}$$

In deriving the above equations, we repeatedly used the fact that $H_{4n}(k)$, $H_{4n+1}(k)$, $H_{4n+2}(k)$ and $H_{4n+3}(k)$ are all periodic on (N/4). In the above Eqns. (6.5) - (6.12)

$$H_{4n}(-k) = H_{4n}[(N/4) - k],$$

$$H_{4n+1}(-k) = H_{4n+1}[(N/2) - k],$$

$$H_{4n+2}(-k) = H_{4n+2}[(3N/4) - k],$$

$$H_{4n+3}(-k) = H_{4n+3}(N - k).$$

In Eqns. (6.5) - (6.12) the only products involved are those shown in Table 6.1

TABLE 6.1 PRODUCTS REQUIRED TO COMPUTE AN N-POINT DHT USING THE RADIX-4 DIT ALGORITHM

1.	$\cos(2\pi k/N)H_{4n+1}(k)$	2.	$\sin(2\pi k/N)H_{4n+1}(-k)$
3.	$\cos[2\pi(2k)/N]H_{4n+2}(k)$	4.	$\sin[2\pi(2k)/N]H_{4n+2}(-k)$
5.	$\cos[2\pi(3k)/N]H_{4n+3}(k)$	6.	$\sin[2\pi(3k)/N]H_{4n+3}(-k)$
7.	$\sin(2\pi k/N)H_{4n+1}(k)$	8.	$\cos(2\pi k/N)H_{4n+1}(-k)$
9.	$\sin[2\pi(2k)/N]H_{4n+2}(k)$	10.	$\cos[2\pi(2k)/N]H_{4n+2}(-k)$
11.	$\sin[2\pi(3k)/N]H_{4n+3}(k)$	12.	$\cos[2\pi(3k)/N]H_{4n+3}(-k)$

$k = 0, 1, \ldots, (N/8) - 1.$

From (6.5) - (6.12), we see that once we form the above products, we can compute the entire DHT sequence. The advantage is that instead of using (6.5) N times to evaluate the N DHT coefficients, we could as well use it only (N/8) times to compute the first (N/8) coefficients, and still obtain the entire N-point sequence with the help of Eqns. (6.6) - (6.12). Table 6.2 explains the above fact for a 64-point sequence.

From Table 6.2 we observe that whenever we compute

$H(k)$, $k = 0, 1, \ldots, 8$ (values in the left most column) using (6.5), with the help of the twelve products in Table 6.1, we can simultaneously obtain all the other coefficients in that row. For example, when we compute H(3) we also obtain H(13), H(19), H(29), H(35), H(45), H(51) and H(61) at the same time without ever actually evaluating them from (6.5), the basic DIT equation. This extraordinary symmetry is not exploited if we employ any radix-2 algorithm for a data sequence whose length is an integer power of four.

TABLE 6.2 WORKING OF THE RADIX-4 DIT ALGORITHM FOR A 64-POINT DHT

k	(N/4)-k	k+N/4	(N/2)-k	k+N/2	(3N/4)-k	k+3N/4	N-k
0		16		32		48	
1	15	17	31	33	47	49	63
2	14	18	30	34	46	50	62
3	13	19	29	35	45	51	61
4	12	20	28	36	44	52	60
5	11	21	27	37	43	53	59
6	10	22	26	38	42	54	58
7	9	23	25	39	41	55	57
8		24		40		56	

Therefore we need $12[(N/8) + 1] = (3N/2) + 2$ multiplications to compute the N-point DHT from four (N/4)-point DHTs. However, whenever $k = 0$ or $k = (N/8)$ we have some savings in the number of multiplications. When $k = 0$, there are no multiplications at all since all the trigonometric functions in the twelve products in Table 6.1 are either zero or one. When $k = (N/8)$ we note that

$k = (N/4) - k \qquad k + (N/4) = (N/2) - k$

$k + (N/2) = (3N/4) - k \qquad k + (3N/4) = (N - k)$

Therefore for this case, Eqns. (6.9) - (6.12) are same as Eqns. (6.5) - (6.8) and, as such, we obtain only four coefficients instead of the usual eight. The following are the equations we obtain when k = N/8.

$$H(N/8) = H_{4n}(N/8) + 2H_{4n+1}(N/8) + H_{4n+2}(N/8) \tag{6.13}$$

$$H(3N/8) = H_{4n}(N/8) - H_{4n+2}(N/8) + 2H_{4n+3}(N/8) \tag{6.14}$$

$$H(5N/8) = H_{4n}(N/8) - 2H_{4n+1}(N/8) + H_{4n+2}(N/8) \tag{6.15}$$

$$H(7N/8) = H_{4n}(N/8) - H_{4n+2}(N/8) - 2H_{4n+3}(N/8). \tag{6.16}$$

In arriving at the above equations, we used the fact

$$\cos(\pi/4) + \sin(\pi/4) = 2$$

We see that the only products involved in the above equations are

$$2[H_{4n+1}(N/8)] \text{ and } 2[H_{4n+3}(N/8)]$$

So, when k = N/8, we need to compute only two products instead of the usual twelve. Hence for the special cases of k = 0 and k = N/8 combined, we have a total of only two products instead of the normal twenty four. Hence the number of multiplications M is given by

$$M = 3N/2 + 12 - 22 = (3N/2) - 10.$$

Combining such multiplications over all the iterations, we get

$$M = [(3N/2) - 10] + 4[(3/2)(N/4) - 10] + 4^2[(3/2)(N/4^2) - 10] + \dots + 4^{P-2}[(3/2)(N/4^{P-2}) - 10],$$

where $P = \log_4 N$.

Simplifying the above expression further, we get

$$M = (3N/2)(P - 1) - 10[1 + 4 + 4^2 + \ldots + 4^{P-2}]$$

$$= (3N/2)P - (3N/2) - 10[4^{P-1}-1]/3$$

$$M = (3N/2)P - (7N/3) + 10/3.$$

Additions

Once we form the twelve products in Table 6.1, let us form the summations given in Table 6.3.

TABLE 6.3 PRIMARY SUMMATIONS REQUIRED TO COMPUTE AN N-POINT DHT USING THE RADIX-4 DIT ALGORITHM

SUM1 = PROD1 + PROD2	SUM2 = PROD3 + PROD4
SUM3 = PROD5 + PROD6	SUM4 = PROD7 + PROD8
SUM5 = PROD9 + PROD10	SUM6 = PROD11 + PROD12

where PROD1, PROD2, etc. are the products in Table 6.1. Thus we have a total of six summations to compute. Now, again referring to Eqns. (6.5) - (6.12), we note that they can be implemented as shown below.

$$H(k) = [H_{4n}(k) + SUM2] + [SUM1 + SUM3]$$

$$H(k + N/4) = [H_{4n}(k) - SUM2] + [SUM6 - SUM4]$$

$$H(k + N/2) = [H_{4n}(k) + SUM2] - [SUM1 + SUM3]$$

$$H(k + 3N/4) = [H_{4n}(k) - SUM2] - [SUM6 - SUM4]$$

$$H[(N/4) - k] = [H_{4n}(-k) + SUM5] + [SUM1 - SUM3]$$

$H[(N/2) - k] = [H_{4n}(-k) - SUM5] + [SUM4 + SUM6]$

$H[(3N/4) - k] = [H_{4n}(-k) + SUM5] - [SUM1 - SUM3]$

$H(N - k) = [H_{4n}(-k) - SUM5] - [SUM4 + SUM6]$

If we do the above summations directly, we need three additions for each coefficient, thus totaling 24 additions. If we observe the above summations carefully, we notice that each parenthesized sum occurs twice. So instead of repeating an addition already performed, we store it in a temporory register once it is computed, as shown in Table 6.4

TABLE 6.4 INTERMEDIATE SUMMATIONS

$T1 = H_{4n}(k) + SUM2$	$T3 = SUM1 + SUM3$
$T2 = H_{4n}(k) - SUM2$	$T4 = SUM6 - SUM4$

Then, (6.5) - (6.8) can be implemented as given in Table 6.5.

TABLE 6.5 ADDITIONS TO GENERATE THE FIRST FOUR DHT COEFFICIENTS

$H(k) = T1 + T3$	$H(k + N/2) = T1 - T3$
$H(k + N/4) = T2 + T4$	$H(k + 3N/4) = T2 - T4$

Once $H(k)$, $H(k + N/4)$, $H(k + N/2)$ and $H(k + 3N/4)$ are formed the sums stored in T1 through T4 are no longer useful. To save memory space, we can use the same registers this time to form the summations given in Table 6.6.

TABLE 6.6 ADDITIONAL INTERMEDIATE SUMMATIONS

$T1 = H_{4n}(-k) + SUM5$	$T3 = SUM1 - SUM3$
$T2 = H_{4n}(-k) - SUM5$	$T4 = SUM4 + SUM6$

Then, Eqns. (6.9) - (6.12) are implemented as given in Table 6.7.

TABLE 6.7 ADDITIONS TO COMPUTE THE LAST FOUR DHT COEFFICIENTS

$H[(N/4) - k] = T1 + T3$	$H[(N/2) - k] = T2 + T4$
$H[(3N/4) - k] = T1 - T3$	$H(N - k) = T2 - T4$

If we add all the summations in Tables 6.3 through 6.7, we have a total of 22 additions. Normally, we will have to carry out such additions for $k = 0$ through $k = N/8$ resulting in a total of

$[(N/8) + 1]22 = (11/4)N + 22$ additions. However, for the special cases of $k = 0$ and $k = (N/8)$ we have some savings in the number of additions. For $k=0$, Eqns. (6.9) through (6.12) are only repetitions of Eqns. (6.5) through (6.8) and, as such, we obtain only four distinct coefficients. To compute them we form the four sums given below.

$$T1 = H_{4n}(0) + H_{4n+1}(0) \qquad T3 = H_{4n+2}(0) + H_{4n+3}(0)$$

$$T2 = H_{4n}(0) - H_{4n+1}(0) \qquad T4 = H_{4n+2}(0) - H_{4n+3}(0)$$

Then the four coefficients are obtained as given below.

$$H(0) = T1 + T3 \qquad H(N/2) = T2 + T4$$

$$H(N/4) = T1 - T3 \qquad H(3N/4) = T2 - 4$$

Thus only eight additions are needed when $k = 0$

When $k = (N/8)$, to obtain the four DHT coefficients given by Eqns. (6.13) through (6.16), we first form the following four summations.

$$T1 = H_{4n}(N/8) \qquad T2 = 2[H_{4n+1}(N/8)]$$

$$T3 = H_{4n+2}(N/8) \qquad T4 = 2[H_{4n+3}(N/8)]$$

Then those equations can be implemented as given below

$$H(N/8) = T1 + T2 + T3$$

$$H(3N/8) = T1 - T3 + T4$$

$$H(5N/8) = T1 - T2 + T3$$

$$H(7N/8) = T1 - T3 - T4.$$

As described above, we thus need only 16 additions for the special cases $k = 0$ and $k = (N/8)$ combined, instead of the usual 44. Hence the total number of additions required to compute the N-point DHT from four (N/4)-point DHTs is given by

$$A = (11N/4) + 22 - 44 + 16 = (11N/4) - 6$$

Summing all such additions over all the iterations until we reach 16-point DHTs, we get

$$A = [(11N/4) - 6] + 4[(11/4)(N/4) - 6] + 4^2[(11/4)(N/4^2) - 6]$$
$$+ \ldots + 4^{P-2}[(11/4)(N/4^{P-2}) - 6]$$

where $P = \log_4 N$.

$$= [illegible] /4)NP - 11(N/4) - (N/2) + 2.$$

In addition, we also have $8[4^{P-1}] = 2N$ additions to compute all the necessary 4-point DHTs. Hence the total number of additions is

$$A = (11N/4)P - 5N/4 + 2.$$

7. SPLIT-RADIX FAST HARTLEY TRANSFORM ALGORITHM

7.1 Introduction

In the previous chapters two different types of radix-2 algorithms were described to compute the DHT of data sequences whose length is an integer power of two. In the last chapter a radix-4 algorithm was introduced which computes the DHT of a data sequence more efficiently if its length is an integer power of four. In this chapter, a new algorithm which involves the application of both the radix-2 and radix-4 algorithms is described. The new algorithm, known as the split-radix algorithm, is computationally more efficient than either the radix-2 or the radix-4 algorithm. The idea of the split-radix algorithm was first proposed in 1984 by Duhamel [12] to compute the DFT of a data sequence. A similar algorithm was proposed by Soo-chang Pei and Ja-ling Wu [13] to compute the DHT of a real-valued sequence. Using the index mapping approach Sorensen _et_ _al_. [9] derived a more efficient split-radix algorithm to compute the DHT. In this chapter the decomposition formula for the algorithm is derived from the definition of the DHT without taking recourse to the index mapping approach, and its working is described.

7.2 The Decomposition Formula

The split-radix algorithm applies the radix-2 DIF decomposition to the even-indexed samples, and the radix-4 DIF decomposition to the odd-indexed samples of the data sequence.

The DHT of an N-point real-valued data sequence is given by

$$H(k) = (1/N) \sum_{n=0}^{N-1} x(n)\mathrm{cas}(2\pi nk/N), \quad k = 0, 1, \ldots, N-1. \quad (7.1)$$

We can rewrite the above summation as shown below.

$$H(k) = (1/N) \sum_{n=0}^{(N/2)-1} x(n)\mathrm{cas}(2\pi nk/N) + (1/N) \sum_{n=N/2}^{N-1} x(n)\mathrm{cas}(2\pi nk/N)$$

$$= (1/N)\left[\sum_{n=0}^{(N/2)-1} x(n)\mathrm{cas}(2\pi nk/N) + \sum_{n=0}^{(N/2)-1} x(n+N/2)\mathrm{cas}\{2\pi k(n + N/2)/N\}\right],$$

$$k = 0, 1, \ldots, N-1. \quad (7.2)$$

In the rest of the derivation the factor (1/N) is not carried through, but is restored in the final step of the derivation.

Let us consider the even- and odd- indexed DHT coefficients separately.

Let k = 2m in Eqn. (7.2). Then

$$H(2m) = \sum_{n=0}^{(N/2)-1} x(n)\mathrm{cas}[2\pi n(2m)/N] + \sum_{n=0}^{(N/2)-1} x(n + N/2)\mathrm{cas}[2\pi(n + N/2)(2m)/N]$$

$$= \sum_{n=0}^{(N/2)-1} x(n)\mathrm{cas}[2\pi n(2m)/N] + \sum_{n=0}^{(N/2)-1} x(n + N/2)\mathrm{cas}[2\pi n(2m)/N]$$

$$= \sum_{n=0}^{(N/2)-1} [x(n) + x(n + N/2)]\mathrm{cas}\{2\pi nm/(N/2)\},$$

$$m = 0, 1, \ldots, (N/2) - 1. \quad (7.3)$$

The above is an (N/2)-point DHT. Thus the even-indexed coefficients H(2m) are obtained by repeatedly applying the above radix-2 DIF formula.

Now let us consider the odd-indexed samples in the DHT sequence. Let k = 2p + 1.

Then we can rewrite Eqn. (7.2) as

$$H(2p+1) = \sum_{n=0}^{(N/2)-1} x(n)\mathrm{cas}[2\pi n(2p+1)/N]$$

$$+ \sum_{n=0}^{(N/2)-1} x(n+N/2)\mathrm{cas}[2\pi(n+N/2)(2p+1)/N]$$

Since

$$\mathrm{cas}[(2p+1)\pi + x] = \mathrm{cas}[\pi + x] = -\mathrm{cas}x,$$

the above equation becomes

$$H(2p+1) = \sum_{n=0}^{(N/2)-1} x(n)\mathrm{cas}[2\pi(2p+1)n/N]$$

$$- \sum_{n=0}^{(N/2)-1} x(n+N/2)\mathrm{cas}[2\pi(2p+1)n/N].$$

$$= \sum_{n=0}^{(N/2)-1} [x(n) - x(n+N/2)]\mathrm{cas}\{2\pi n(2p+1)/N\},$$

$$p = 0, 1, \ldots, (N/2) - 1. \quad (7.4)$$

However,

$$\mathrm{cas}[2\pi n(2p+1)/N] = \cos(2\pi n/N)\mathrm{cas}(4\pi np/N) + \sin(2\pi n/N)\mathrm{cas}\{4\pi n(-p)\}.$$

Using the above result in Eqn. (7.4), we get

$$H(2p+1) = \sum_{n=0}^{(N/2)-1} [x(n) - x(n+N/2)]\cos(2\pi n/N)\mathrm{cas}(4\pi np/N)$$

$$+ \sum_{n=0}^{(N/2)-1} [x(n) - x(n + N/2)]\sin(2\pi n/N)\mathrm{cas}\{4\pi n(-p)/N\}.$$

If the direction of summation in the second sum is reversed, we get

$$H(2p + 1) = \sum_{n=0}^{(N/2)-1} [\{x(n) - x(n + N/2)\}\cos(2\pi n/N)$$

$$+ \{x(N/2 - n) - x(N - n)\}\sin(2\pi n/N)]\mathrm{cas}(4\pi np/N),$$

$$p = 0, 1, \ldots,(N/2) - 1. \qquad (7.5)$$

The above is the radix-2 DIF decomposition formula to obtain the odd-indexed coefficients of the DHT. If we were interested in only a radix-2 algorithm, we would have stopped at this point. However, in the split-radix algorithm we once again apply a radix-2 decomposition to the formula in Eqn. (7.5). We observe that this additional radix-2 decomposition is applied only in evaluating the odd-indexed coefficients, but not in evaluating the even-indexed samples. If we had done so, we would be effectively deriving the usual radix-4 DIF algorithm.

We can split the (N/2)-point summation in Eqn. (7.5) into two (N/4)-point summations as follows.

$$(H(2p + 1) = \sum_{n=0}^{(N/4)-1} [\{x(n) - x(n + N/2)\}\cos(2\pi n/N)$$

$$+ \{x(N/2 - n) - x(N - n)\}\sin(2\pi n/N)]\mathrm{cas}(4\pi np/N)$$

$$+ \sum_{n=0}^{(N/4)-1} [\{x(n + N/4) - x(n + 3N/4)\}\cos\{2\pi(n + N/4)/N\}$$

$$+ \{x(N/4 - n) - x(3N/4 - n)\}\sin\{2\pi(n + N/4)/N\}]\mathrm{cas}\{4\pi p(n + N/4)/N\}$$

Using the familiar trigonometric identities, we can rewrite the above equation as

$$H(2p + 1) = \sum_{n=0}^{(N/4)-1} [\{x(n) - x(n + N/2)\}\cos(2\pi n/N)$$

$$+ \{x(N/2 - n) - x(N - n)\}\sin(2\pi n/N)]\text{cas}(4\pi np/N)$$

$$+ \sum_{n=0}^{(N/4)-1} [\{x(N/4 - n) - x(3N/4 - n)\}\cos(2\pi n/N)$$

$$- \{x(n + N/4) - x(n + 3N/4)\}\sin(2\pi n/N)]\text{cas}\{\pi p + 4\pi np/N\} \quad (7.6)$$

If we let p = 2m in the above relation, we obtain the 'even' samples H(4m + 1) of the odd-indexed DHT sequence. After further simplification we finally obtain

$$H(4m + 1) = \sum_{n=0}^{(N/4)-1} [\{x(n) - x(n + N/2) + x(N/4 - n)-x(3N/4-n)\}\cos(2\pi n/N)$$

$$+ \{x(N/2-n) - x(N-n) + x(n+3N/4) - x(n+N/4)\}\sin(2\pi n/N)]\text{cas}\{2\pi nm/(N/4)\},$$

$$m = 0, 1, \ldots, (N/4) - 1. \quad (7.7)$$

The above equation is the (N/4)-point DHT that computes the odd DHT coefficients H(4m+1), m = 0, 1, ..., (N/4)-1.

If we let p = 2m + 1 in Eqn. (7.6), we obtain the 'odd' coefficients H(4m+3) of the DHT.

$$H(4m + 3) = \sum_{n0}^{(N/4)-1} [\{x(n) - x(n + N/2)\}\cos(2\pi n/N)$$

$$+ \{x(N/2 - n) - x(N - n)\}\sin(2\pi n/N)]\text{cas}\{2\pi n(2m + 1)/(N/2)\}$$

$$+ \sum_{n=0}^{(N/4)-1} [\{x(n + 3N/4) - x(n + N/4)\}\sin(2\pi n/N)$$

$$- \{x(N/4 - n) - x(3N/4 - n)\}\cos(2\pi n/N)]\mathrm{cas}\{2\pi n(2m + 1)/(N/2)\}$$

Collecting the terms, we get

$$H(4m + 3) = \sum_{n=0}^{(N/4)-1} \{x(n) - x(n+N/2) - x(N/4 - n) + x(3N/4-n)\}\cos(2\pi n/N)$$

$$\mathrm{cas}\{2\pi n(2m + 1)/(N/2)\}$$

$$+ \sum_{n=0}^{(N/4)-1} \{x(N/2 - n) - x(N - n) - x(n + 3N/4) + x(n + N/4)\}\sin(2\pi n/N)]$$

$$\mathrm{cas}(2\pi n(2m +)/(N/2). \qquad (7.8)$$

However,

$$\mathrm{cas}[2\pi n(2m + 1)/(N/2)] = \cos\{2\pi(2n)/N\}\mathrm{cas}\{2\pi[2nk/(N/2)]\}$$

$$+ \sin\{2\pi(2n)/N)\mathrm{cas}\{2\pi(2n)(-k)/(N/2)\}. \qquad (7.9)$$

Substituting the above expansion in the first sum of Eqn. (7.8) and calling it SUM1, we get

$$SUM1 = \sum_{n=0}^{(N/4)-1} \{x(n) - x(n + N/2) - x(N/4 - n) + x(3N/4 - n)\}\cos(2\pi n/N)$$

$$\cos\{2\pi(2n)/N)\mathrm{cas}\{2\pi(2nk)/(N/2)\}$$

$$+ \sum_{n=0}^{(N/4)-1} \{x(n) - x(n + N/2) - x(N/4 - n) + x(3N/4 - n)\}\cos(2\pi n/N)$$

$$\sin\{2\pi(2n)/N)\mathrm{cas}\{2\pi(2n)(-k)/(N/2)\}$$

If we reverse the direction of summation of the second sum in the above equation, i.e. substituting {(N/4) - n} in place of n, we get

$$\mathrm{SUM1} = \sum_{n=0}^{(N/4)-1} \{x(n) - x(n + N/2) - x(N/4 - n) + x(3N/4 - n)\}\cos(2\pi n/N)$$

$$\cos[2\pi(2n)/N]\mathrm{cas}\{2\pi(2nk)/(N/2)\}$$

$$+ \sum_{n=0}^{(N/4)-1} \{x(N/4 - n) - x(3N/4 - n) - x(n) + x(n + N/2)\}\sin(2\pi n/N)$$

$$\sin[2\pi(2n)/N]\ \mathrm{cas}\{2\pi(2nm)/(N/2)\}.$$

combining the above two summations using the trigonometric relation,

$$\cos(A + B) = \cos A\cos B - \sin A\sin B,$$

we get

$$\mathrm{SUM1} = \sum_{n=0}^{(N/4)-1} \{x(n) - x(n + N/2) + x(3N/4 - n) - x(N/4 - n)\}\cos\{2\pi(3n)/N\}$$

$$\mathrm{cas}\{2\pi nm/(N/4)\} \qquad (7.10)$$

Now, considering the second sum in Eqn. (7.8) with the result of Eqn. (7.9) substituted in it and calling it SUM2, we get

$$\mathrm{SUM2} = \sum_{n=0}^{(N/4)-1} \{x(N/2 - n) - x(N - n) - x(n + 3N/4) + x(n + N/4)\}$$

$$\sin(2\pi n/N)\cos\{2\pi(2n)/N)\ \mathrm{cas}\{2\pi(2nk)/(N/2)\}$$

$$+ \sum_{n=0}^{(N/4)-1} \{x(N/2 - n) - x(N - n) - x(n + 3N/4) + x(n + N/4)\}$$

$$\sin(2\pi n/N)\sin\{2\pi(2n)/N\}\ \mathrm{cas}\{2\pi(2n)(-m)/(N/2)\}$$

reversing the direction of summation in the second sum in the above equation, we get

$$\mathrm{SUM2} = \sum_{n=0}^{(N/4)-1} \{x(N/2 - n) - x(N - n) - x(n + 3N/4) + x(n + N/4)\}$$

$$\sin(2\pi n/N)\cos\{2\pi(2n)/N\}\ \mathrm{cas}\{2\pi(2nm)/(N/2)\}$$

$$+ \sum_{n=0}^{(N/4)-1} \{x(N/4 + n) - x(n + 3N/4) - x(N - n) + x(N/2 - n)\}$$

$$\cos(2\pi n/N)\sin\{2\pi(2n)/N\}\ \mathrm{cas}\{2\pi(2nm)/(N/2)\}.$$

Using the trigonometric relation

$$\sin(A + B) = \sin A\cos B + \cos A\sin B$$

and combining the two summations in the above equation we get

$$\mathrm{SUM2} = \sum_{n=0}^{(N/4)-1} \{x(N/2 - n) - x(N - n) - x(n + 3N/4) + x(n + N/4)\}$$

$$\sin\{2\pi(3n)/N)\cos\{2\pi(nm)/(N/4)\}. \qquad (7.11)$$

Combining (7.10) and (7.11), we get

$$H(4m + 3) = \sum_{n=0}^{(N/4)-1} [\{x(n) - x(n + N/2) + x(3N/4 - n) - x(N/4 - n)\}$$

$$\cos\{2\pi(3n)/N)$$

$$+ \{x(N/2 - n) - x(N - n) - x(n + N/4) + x(n + N/4)\}$$

$$\sin\{2\pi(3n)/N\}]\mathrm{cas}(2\pi nm/(N/4),$$

$$m = 0, 1, \ldots, (N/4) - 1. \qquad (7.12)$$

Table 7.1 gives a summary of the above discussion.

7.3 Working of the Algorithm

Fig. (7.1) describes the working of the split-radix algorithm in a block diagram for a 32-point data sequence. Observe the way the DHT is

TABLE 7.1 PROCEDURE TO COMPUTE AN N-POINT DHT USING THE SPLIT-RADIX ALGORITHM

Coefficients to be computed	DIF equation to be used
1. Even coefficients $H(2m)$, $m = 0, 1, \ldots, (N/2) - .$	(7.3)
2. Odd coefficints $H(4m + 1)$, $m = 0, 1, \ldots, (N/4) - 1$	(7.7)
3. Odd coefficients $H(4m + 3)$, $m = 0, 1, \ldots, (N/4) - 1.$	(7.12)

computed. The task is reduced to that of computing a half-length 16-point DHT and two quarter-length 8-point DHTs.

The half-length DHT computes the even-indexed coefficients H(0), H(2), H(4), H(6), H(8), H(10), H(12), H(14), H(16), H(18), H(20), H(22), H(24), H(26), H(28), H(30).

The first quarter-length DHT computes the odd-indexed coefficients H(1), H(5), H(9), H(13), H(17), H(21), H(25), H(29).

The second quarter-length DHT computes the following odd-indexed coefficients H(3), H(7), H(11), H(15), H(19), H(23), H(27), H(31). However the computation of the even- and odd- indexed coefficients is again reduced to three smaller length DHT computations until we reach the stage where we need to compute only 4-point or 2-point DHTs, both of which require no multiplications. The split-radix algorithm also is an in-place algorithm. It is applicable to all sequences whose length is an integer power of two, and is the most efficient of all in-place algorithms.

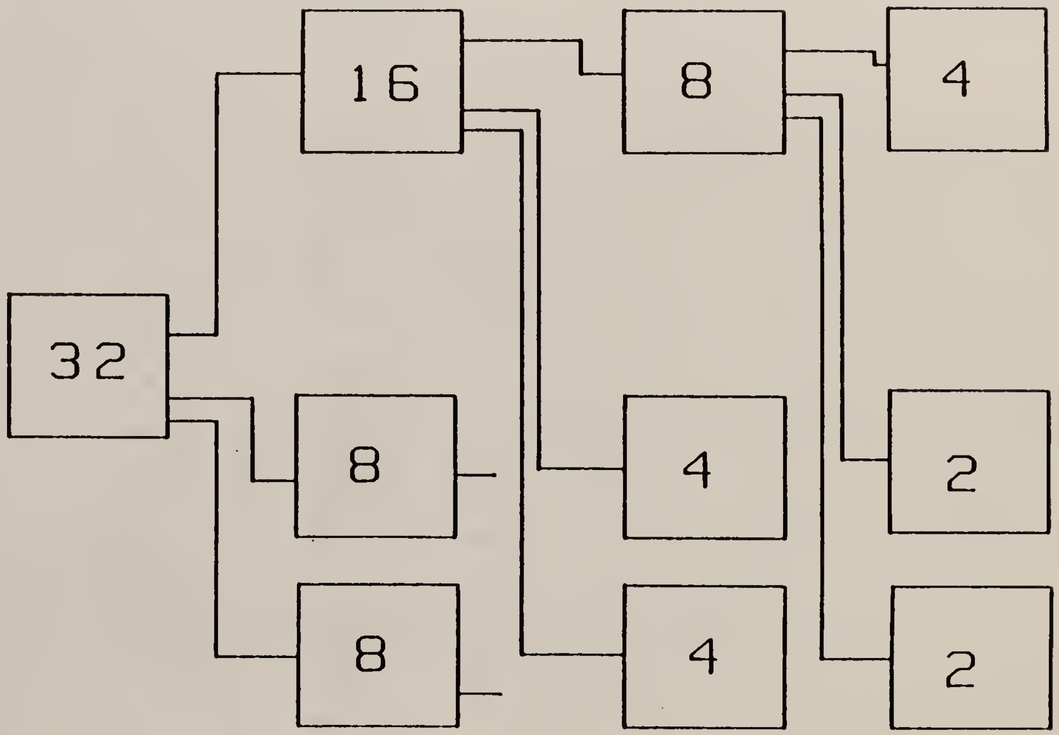

Fig. 7.1 Block diagram of the working of a 32-point split-radix FHT.

The number of multiplications M is given by

$$M = (2N/3)\log_2 N - 19(N/9) + 3 + (-1)^P/9$$

and the the number of additions A is given by

$$A = (4N/3)\log_2 N - (14N/9) + 3 + (-1)^P(5/9)$$

where $P = \log_2 N$.

Tables 7.2 through 7.4 give a summary of the operation counts for different algorithms. By operation counts we mean the total number of multiplications and additions. We can see from these operation counts that the split-radix algorithm uses less multiplications and less additions than either the radix-2 or the radix-4, but still has a fairly compact code. Therefore this is generally the most efficient algorithm for data sequences whose length is a power of two. The operation counts for these algorithms is less than those required to compute the DFT of a real-valued data sequence [14].

TABLE 7.2 OPERATION-COUNTS FOR RADIX-2 ALGORITHMS

Length	Mults	Adds	Mult + Add
RADIX-2			
4	0	8	8
8	4	26	30
16	20	74	94
32	68	194	262
64	196	482	678
128	516	1154	1670
256	1284	2690	3974
512	3076	6146	9222
1024	7172	13826	20998
2048	16388	30722	47110
4096	36868	67586	104454

TABLE 7.3 OPERATION-COUNTS FOR THE RADIX-4 ALGORITHM

Length	Mults	Adds	Mult + Add
4	0	8	8
16	14	70	84
64	142	450	594
256	942	2498	3440
1024	5294	12802	18096
4096	27310	62466	89776

TABLE 7.4 OPERATION-COUNTS FOR THE SPLIT-RADIX ALGORITHM

Length	Mults	Adds	Mult + Add
SPLIT-RADIX			
4	0	8	8
8	2	22	24
16	12	64	76
32	42	166	208
64	124	416	540
128	330	998	1328
256	828	2336	3164
512	1994	5350	7344
1024	4668	12064	16732
2048	10698	26854	37552
4096	24124	59168	83292

8. AN APPLICATION: RAMAN SPECTRA AND MATCHED FILTERING

8.1 Introduction

In this chapter an example application of the DHT in the analysis of Raman spectra is described, wherein it is employed to perform the fast convolution operation. We begin the chapter with a brief introduction to the origin of Raman spectra, and then we discuss the role of the FT and the DFT in the resolution of the spectra. The concept of matched filtering and its applicability in the enhancement of SNR of the spectra is presented. The implementation of the matched filter using both the DFT and the DHT is discussed next, and finally we close the chapter with a comparison of the performance of the DFT and the DHT in the application chosen.

8.2 Raman Spectra

The Raman effect, also known as the scattering of modified radiation, was first discovered in 1928 by C. V. Raman [15]. The phenomenon, first observed in the case of liquids, is universal in character and is a powerful technique in identifying different substances unambiguously. When visible light of a particular frequency is used to excite any material, the spectrum of the scattered light from the substance contains the signature of the material. This phenomenon is known as the Raman effect, and the spectrum of the scattered light is referred to as the Raman spectrum for that substance. For any given substance, the difference in frequency between the incident light and the scattered light is constant and is independent of the frequency of the incident light. This shift in frequency between the incident light and the scattered light corresponds to the frequency of oscillation of the chemically bonded atoms in the substance, which in turn depends on the geometry of the molecule. In terms of quantum theory of light, the difference in frequency between the spectrum of the incident light and the Raman spectrum is positive or negative, depending on whether the incident light is delivering energy to the target molecule or receiving energy from it.

One major drawback in analyzing the Raman spectra is that they are very feeble. Matched filtering is a very useful technique to enhance the SNR in such cases.

8.3 Matched Filtering

8.3.1 Preliminaries

Before describing the matched filter, some preliminary definitions and models for the signal and noise portion of the spectrometer output are presented.

Raman shift: When a substance is subjected to the Raman effect, the frequency of oscillation of the chemically bonded atoms in the substance alters. This shift in frequency is called the Raman shift and is measured in terms of wavenumbers (cm^{-1}).

signals: Spectroscopic signals are pulse-like in nature with wavenumber as the independent variable. For our study here a Lorentzian pulse is considered. It is defined by

$$s(v) = A(v_0)/ [1 + \{(v - v_0)/h\}^2]$$

where $A(v_0)$ is the peak amplitude of s(v) at the center frequency v_0 and h is the half-width at half-height (hwhh) of the pulse. The Lorentzian pulse is taken for study since it is the most difficult of all the peaks to detect for a given half-width at half-height.

Noise: The noise part of the spectrum is treated as a random process (rp) with white Gaussian statistics.

White noise: So named by analogy to the white light which contains all the visible frequencies, white noise is a zero mean rp whose power spectral density (PSD) is a constant at all the frequencies.

PSD: If x(t) is a wide-sense stationary (wss) rp, and its FT is given by X(f), then its PSD $S_{xx}(f)$ is given by

$$S_{xx}(f) = \lim_{T \to \infty} E\left[|X_T(f)|^2 / 2T \right].$$

where $X_T(f)$ is the FT of the signal x(t) restricted to an interval -T and T. For the white noise $S_{xx}(f)$ is a constant and is given by

$$S_{xx}(f) = N_0 / 2. \qquad (8.1)$$

Autocorrelation: If x(t) is a wss rp, its autocorrelation function $R_{xx}(t)$ is given by

$$R_{xx}(t) = FT^{-1}\left[S_{xx}(f) \right]. \qquad (8.2)$$

From Eqns. (8.1) and (8.2) we see that $R_{xx}(t)$ for the white noise is given by

$$R_{xx}(t) = [N_o/2]\delta(t),$$

where, $\delta(t)$ is the Dirac delta function. As is clear from the above equation, the white noise is uncorrelated to itself.

8.3.2 Theory of the Matched Filter

Matched filter is an optimal linear system which maximizes the SNR at the output of the filter.

We assume that the input to the filter consists of the sum of a deterministic signal s(t) and a random noise process n(t). The signal and noise at the output of the filter are denoted as $s_o(t)$ and $n_o(t)$, respectively. The instantaneous signal power at the output is given by $s_o^2(t)$ and the average noise power as $\bar{N}_o$. If the filter's impulse response is indicated as h(t) and its Fourier transform as H(f), then the output signal $s_o(T)$ at $t = T$ is given by

$$s_o(T) = \int_{-\infty}^{\infty} S(f)H(f)e^{j2\pi fT}\,df,$$

where S(f) is the FT of the signal s(t).

Similarly the output average noise power is

$$\bar{N}_o = \int_{-\infty}^{\infty} S_{NN}(f)|H(f)|^2 df \qquad (8.3)$$

where $S_{NN}(f)$ is the PSD of the input noise process. Therefore the output SNR is

$$S_o/N_o = \left| \int_{-\infty}^{\infty} S(f)H(f)e^{j2\pi fT}\, df \right|^2 / \bar{N}_o$$

where $\bar{N}_o$ is given in Eqn. (8.3)

It can be shown [16] that the above ratio is maximum when

$$H(f) = H_{opt}(f) = C\,\frac{S^*(f)e^{-j2\pi fT}}{S_{NN}(f)} \qquad (8.4)$$

where C is a real constant and $S^*(f)$ is the complex conjugate of S(f).

8.3.3 Matched Filter for White Noise

We know that if the input noise process is white, its PSD $S_{NN}(f) = N_o/2$. Substituting the above value for $S_{NN}(f)$ in Eqn. (8.4), we get

$$H_{opt}(f) = KS^*(f)e^{-j2\pi fT}$$

where $K = 2C/N_o$ is a real constant. The impulse response of the filter is given by

$$h_{opt}(t) = \int_{-\infty}^{\infty} H_{opt}(f)e^{j2\pi ft}\, df$$

$$= Ks(T - t). \qquad (8.5)$$

From Eqn. (8.5) it is clear that the impulse response of the matched filter, when the input noise process is white, is equal to the input signal shifted by T units and then reversed. The delay T is a parameter under the control of the filter designer. If we choose $T = 0$ and $K = 1$ in Eqn. (8.5), then it becomes

$$h_{opt}(t) = s(-t). \qquad (8.6)$$

Since the transfer function of the filter is so closely tied to the input signal as shown in the above equation, the filter is referred to as the matched filter. This feature can sometimes be a problem too, in that it requires of us a knowledge of the exact shape of the input signal. However, we can often make good guesses about the input signal profile.

8.3.4 Matched Filter as a Correlator

The output of the matched filter is given by

$$y(t) = x(t)*h_{opt}(t)$$

$$= \int_{-\infty}^{\infty} x(\tau)h_{opt}(t - \tau)d\tau$$

where $x(t) = s(t) + n(t)$ and * denotes convolution. If $n(t)$ is a white noise process, the above equation becomes

$$y(t) = \int_{-\infty}^{\infty} x(\tau)s(-t + \tau)\, d\tau$$

$$= \int_{-\infty}^{\infty} x(t + \tau)s(\tau)d\tau \tag{8.7}$$

The above equation shows that y(t) is the cross correlation of the assumed input signal with the input signal-plus-noise.

8.3.5 Implementation Considerations

The spectrum to be filtered is treated as an N-point sequence

$$\{x(m)\}, \quad m = 0, 1, \ldots, N - 1.$$

and the sampled version $\{h_{opt}(m)\}$ of the filter's impulse response is also made the same length. Then the linear convolution of the input and the filter's impulse response is given by

$$y(k) = \sum_{m=0}^{N-1} x(m)h(k - m), \; k = 0, 1, \ldots, 2N - 2, \tag{8.8}$$

where y(k) is the output of the matched filter. The convolution equation above is the linear convolution of the input spectrum and the filter's impulse response. However, the linear convolution can be achieved by adding atleast N - 1 zeros to both {x(m)} and {h(m)} and then performing a circular convolution on the resulting sequences. The advantage of the approach lies in the fact that we could use the DFT or the DHT to compute the circular convolution. Fig. (8.1) shows the block diagram for implementing the matched filter using the DFT. Dyer and Hardin [17] used the above approach in analyzing simulated Raman spectra using the matched filter technique.

Implementation of the matched filter using the DHT approach is presented here.

Fig. (8.2) shows the block diagram of the implementation of fast linear convolution using the DHT. The N-point spectrum {x(m)} and the N-point digitized filter transfer function {h(m)} are both zero padded upto 2N points. Note that for the matched filter, {h(m)} is the reversed version of the input pulse. Since a Lorentzian pulse centered at zero is an even pulse, obeying the relation $h(m) = h(N - m)$, the DHT of the circular convolution of x(m) and h(m) is the product of the individual DHTs $H_1(k)$ and $H_2(k)$ of x(m) and h(m), respectively. Using this fact, we find the DHT of the circular convolution of the 2N-point sequences {x(m)} and {h(m)} as shown in Fig. 8.2. Since the DHT is a symmetric transform, by performing the 2N-point DHT on the resulting sequence, we get the linear convolution of x(m) and h(m). Since we are interested only in the enhancement of the N-point spectrum that we started with, we discard the last N points in the resulting sequence. The main advantages of the DHT approach are that it computes the linear convolution faster than the DFT technique and, that it is simpler to implement than the latter when one of the functions being convolved is even.

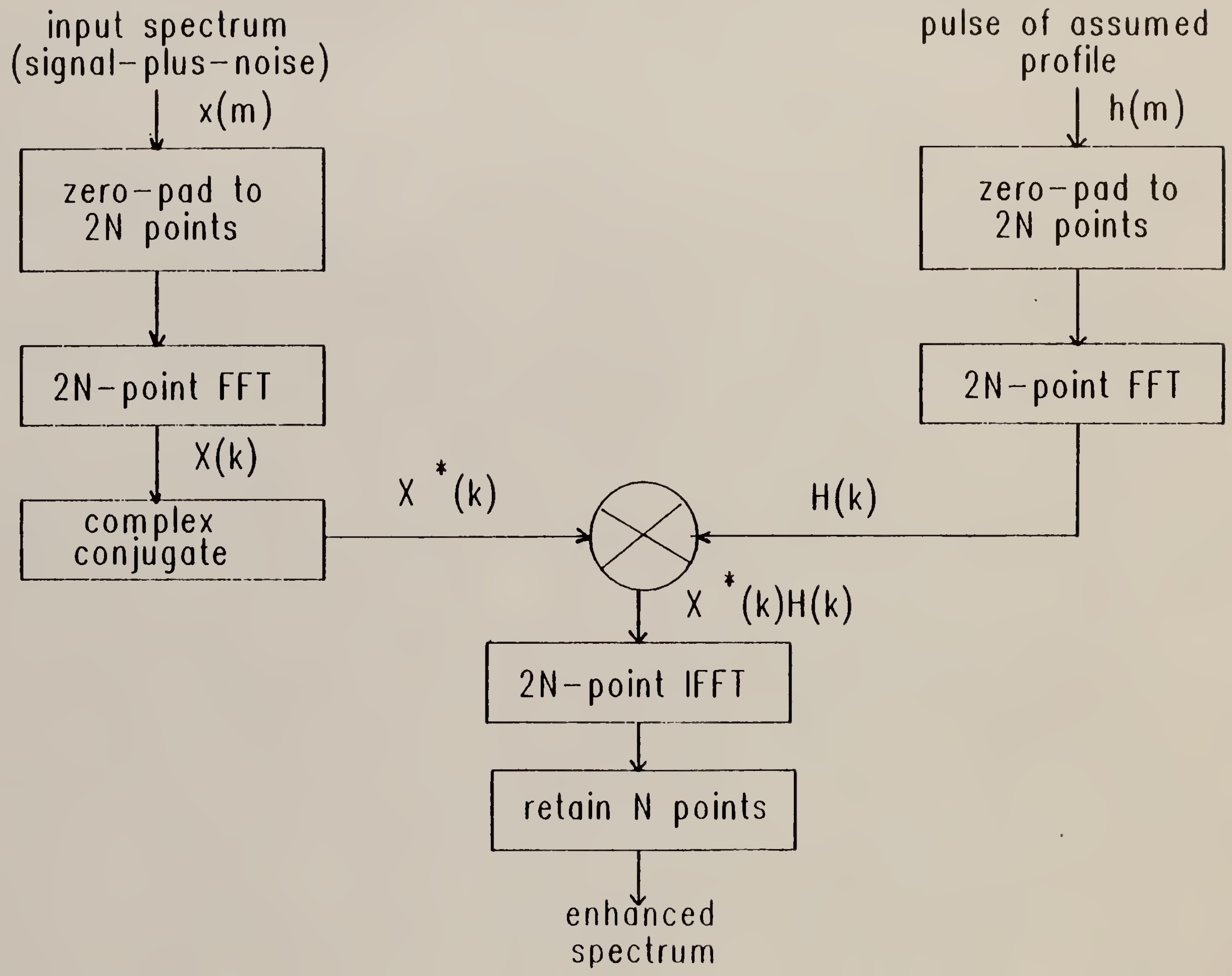

Fig. 8.1 Block diagram of an implementation of fast linear cross-correlation using the DFT.

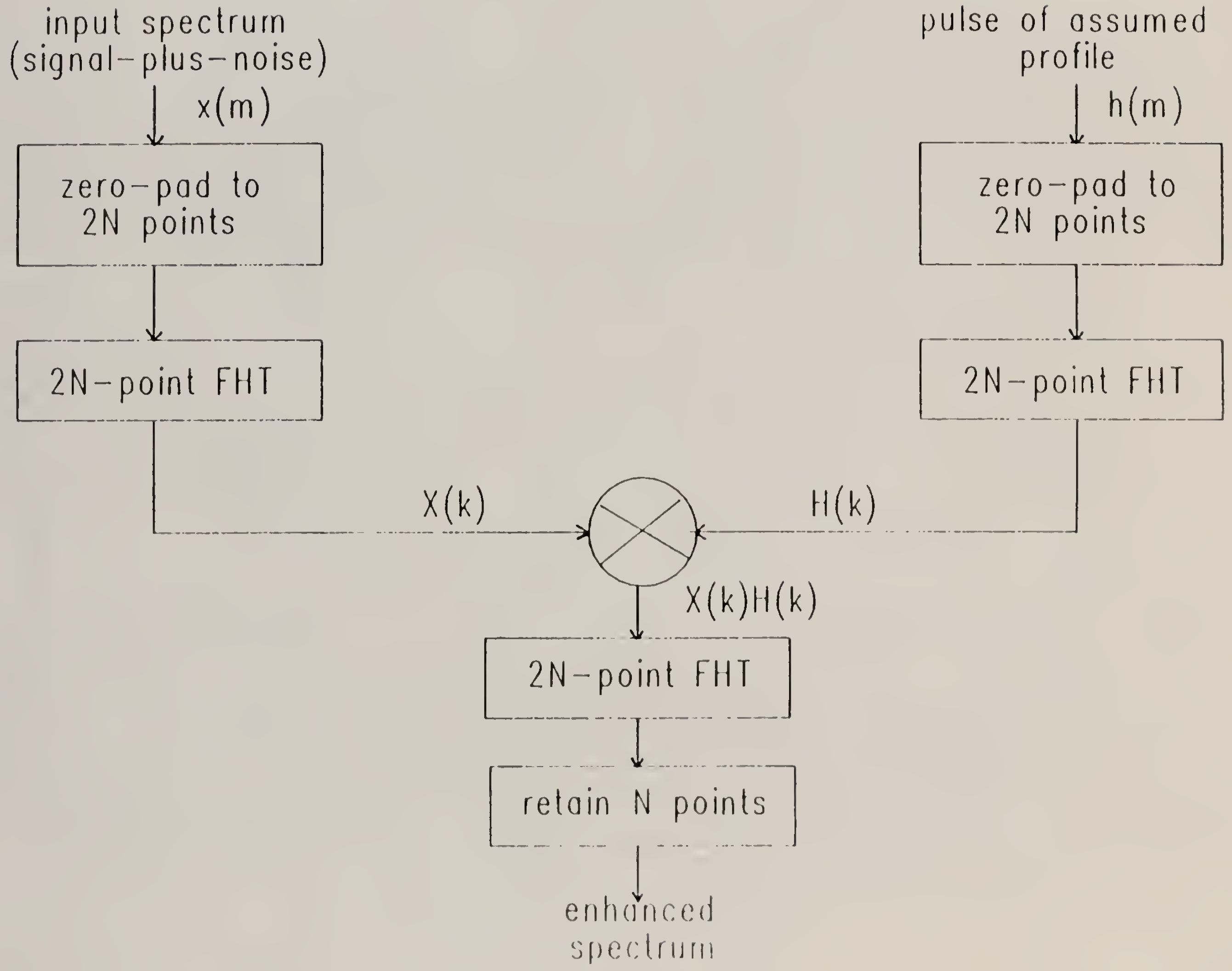

Fig. 8.2 Block diagram of an implementation of fast linear convolution using the DHT when one of the sequences is even.

8.4 Simulation Results

As an example application of the DHT, simulated Raman spectra were analyzed using matched filter. The signal taken for study consisted of two Lorentzian peaks, one centered at 250 cm^{-1} and the other at 650 cm^{-1}, and was shown in Fig. 8.3. The half-width at half-height (hwhh) of each peak was 20 cm^{-1}, and the sampling rate was 1024 samples/sec. White Gaussian noise having unity variance was added to the signal to generate the input spectrum. Fig. 8.4 represents the noise and Fig. 8.5 depicts the input spectrum.

The input spectrum consisted of 1024 points and the SNR (maximum amplitude of peak/ rms amplitude of noise) was 2.0. Fig. 8.6 presents the output of a matched filter having the spectrum of Fig. 8.5. as its input. The impulse response of the filter was a reversed Lorentzian peak having a hwhh of 20 cm^{-1}. The two spectral peaks are evident in the output spectrum, one located at 250 cm^{-1} and the other at 650 cm^{-1}.

Fig. 8.7 presents the output of the same matched filter imlemented using the DHT instead of the DFT. Again, the spectral peaks are clearly discernible from the noise at 250 cm^{-1} and 650 cm^{-1}.

As evident from the output spectrum, the filter output is insensitive to the choice of the transform. The analysis was carried out with two other filters also, one matched to a Lorentzian peak of hwhh = 5

cm^{-1} and the other to a Lorentzian peak with hwhh = 10 cm^{-1}. The filters were implemented with the DFT as well as the DHT. Table 8.1 shows a summary of the computation times for both the methods.

TABLE 8.1 COMPUTATION TIME FOR IMPLEMENTING THE MATCHED FILTER

HWHH of filter's response	Approach	CPU time in secs.
5 cm^{-1}	DHT	2.80
	DFT	3.06
10 cm^{-1}	DHT	2.83
	DFT	3.00
20 cm^{-1}	DHT	2.77
	DFT	3.05

On the average the DHT approach has taken 2.8 secs compared to 3.04 secs. for the DFT approach for a 1024 point spectrum.

Software for the matched filter was written in VAX-11 FORTRAN and was given in the Appendix. The white noise was generated using RALPH, a general-purpose DSP software developed at Kansas State University.

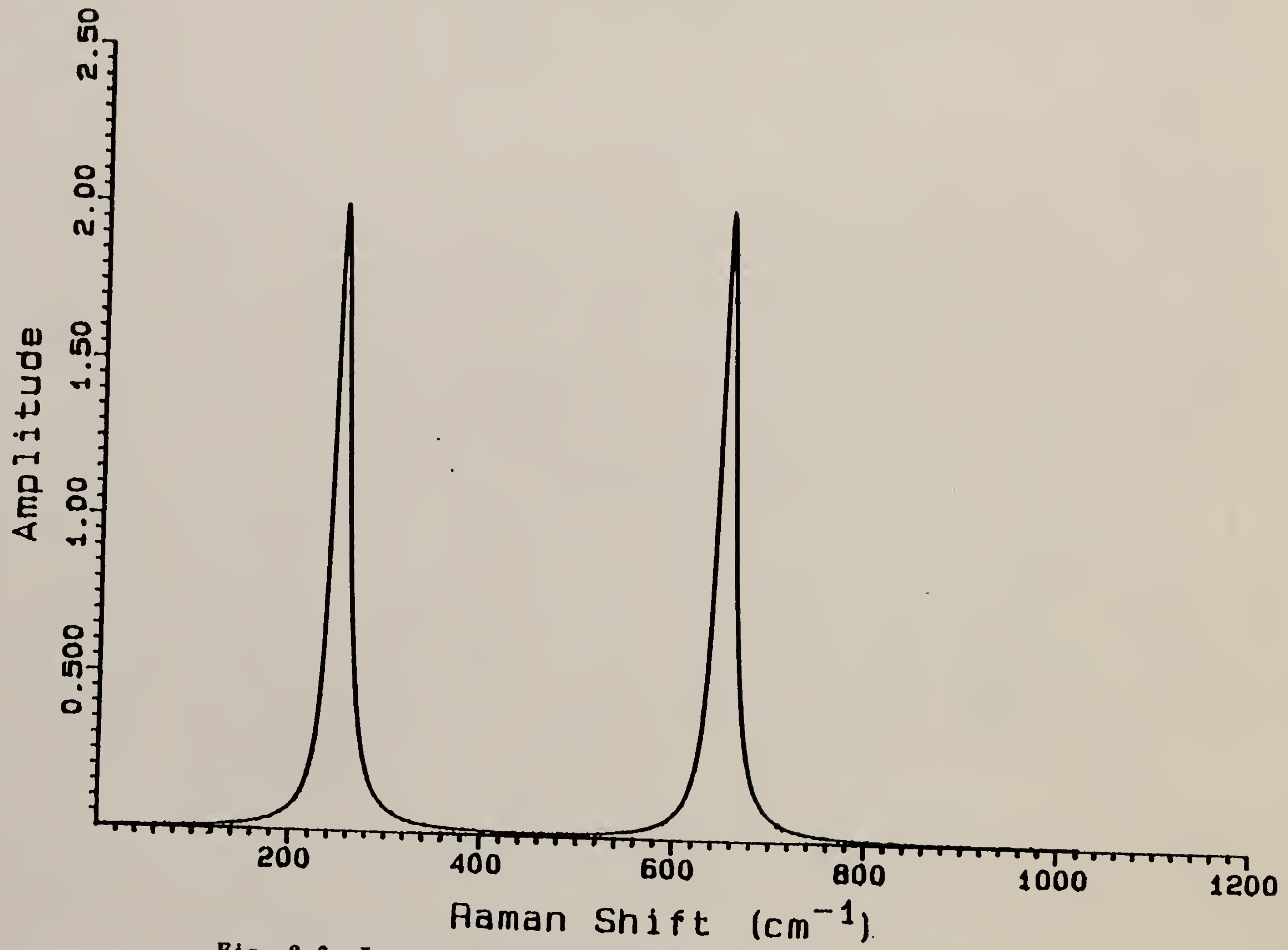

Fig. 8.3 Input signal consisting of two Lorentzian peaks, (ampliude = 2.0, hwhh = 20 cm^{-1}).

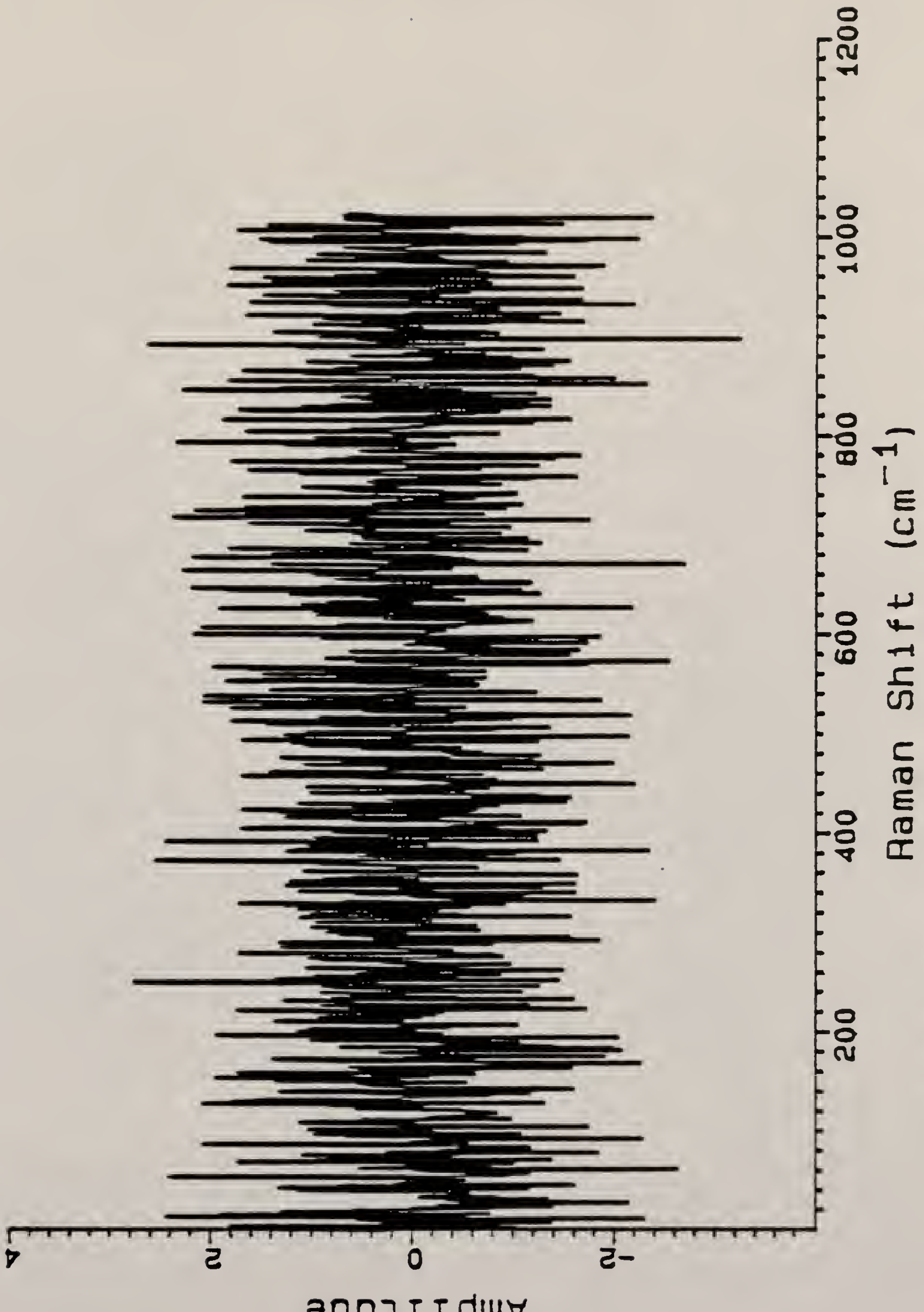

Fig. 8.4 Zero-mean unit-variance white Gaussian noise used in the simulation example.

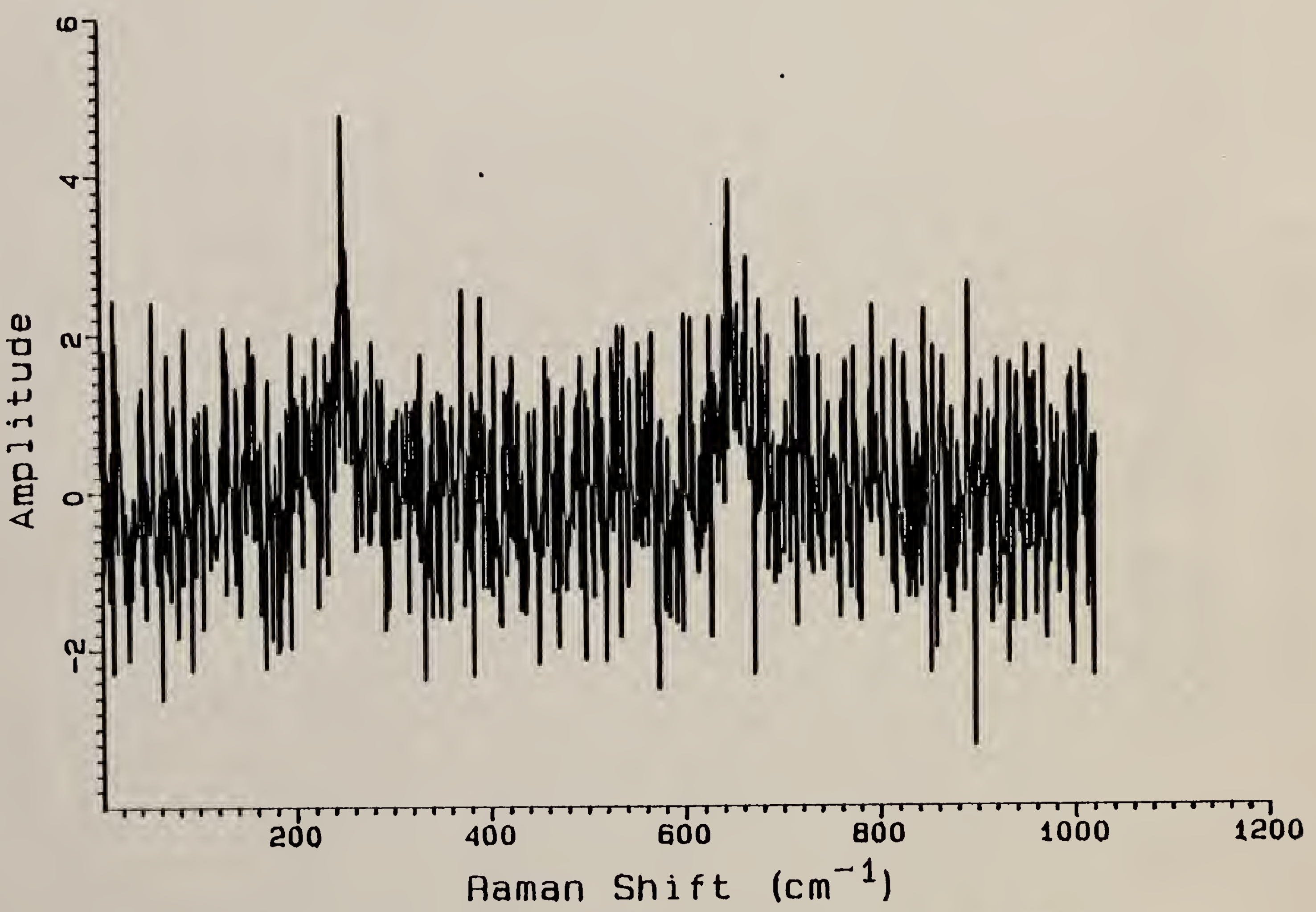

Fig. 8.5 Input spectrum consisting of the input signal and the additive noise at SNR = 2.0.

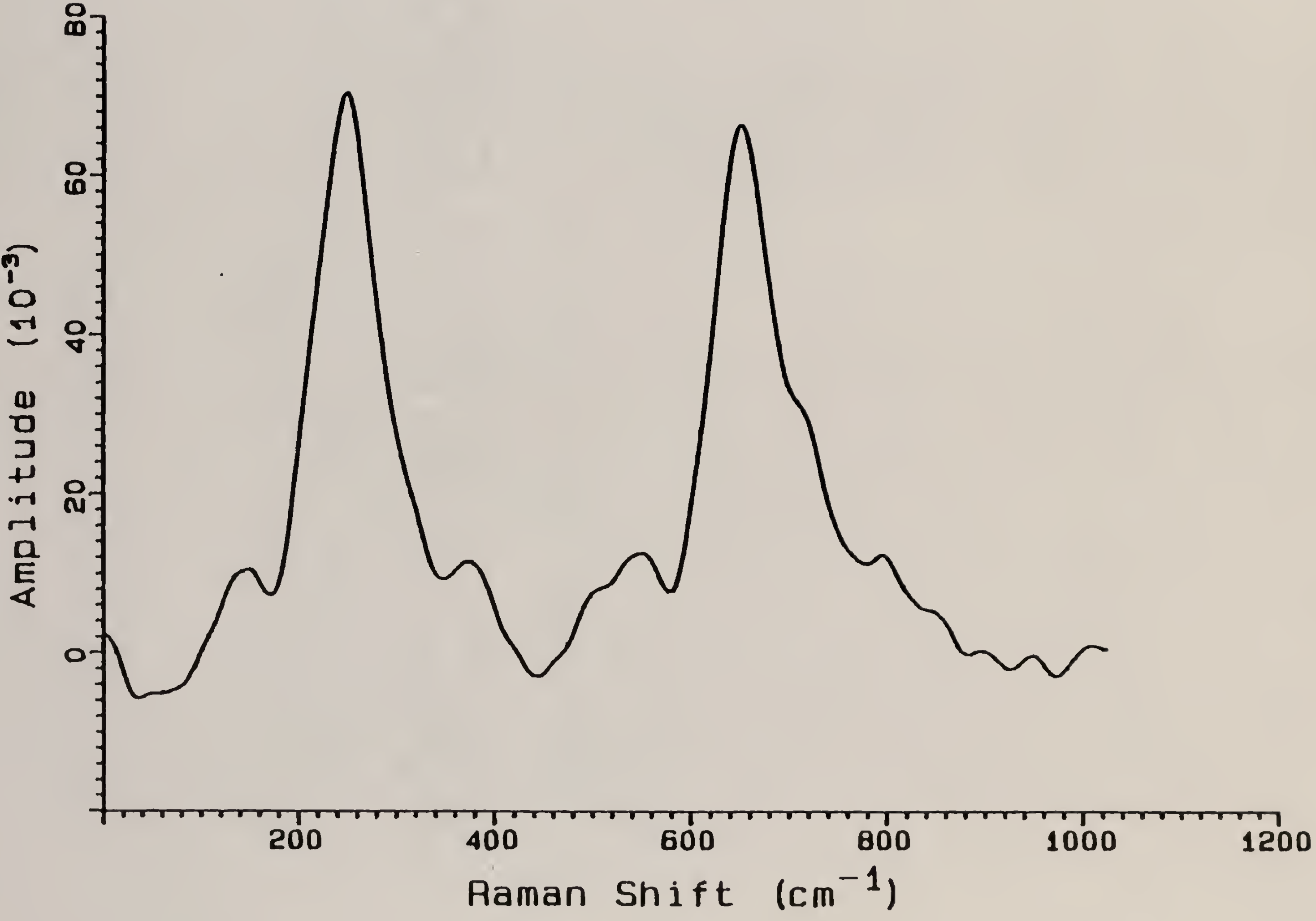

Fig. 8.6 Output of the matched filter for the input signal-plus-noise when the filter is implemented via the DFT.

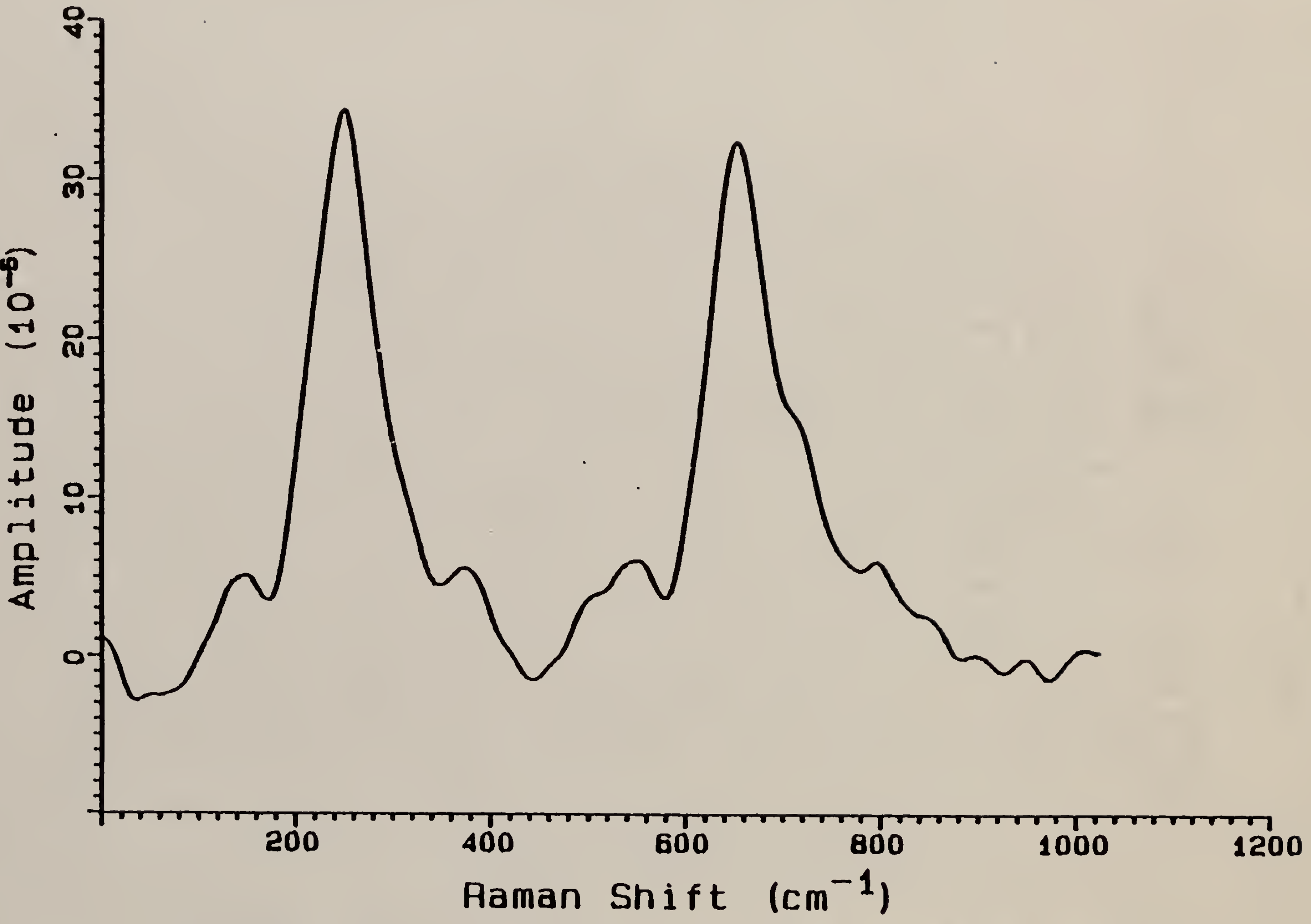

Fig. 8.7 Output of the matched filter for the input signal-plus-noise when the filter is implemented via the DHT.

9. CONCLUSIONS

Various properties of the HT for a real signal, and the DHT for a real data sequence were derived in the first two chapters. It was shown that all the properties of the FT and the DFT have a counterpart in the HT theory. These properties look very much identical, especially when the data sequence possesses either odd or even symmetry.

The DFT of a data sequence generates a complex-valued sequence. However, the DFT of a real-valued data has a useful property that its real part is even and its imaginary part is odd. General-purpose FFT algorithms written to deal with real- as well as complex-valued data do not take advantage of the above property. However those algorithms can be optimized for computing the DFT of real-valued data sequences more efficiently to save memory and computation time. The DHT helps us avoid taking recourse to such complicated methods of optimizing to achieve efficiency in dealing with real-valued data. Once the DHT of the real-valued data is computed, using a simple relation between the DFT and the DHT, we obtain the DFT of the sequence. This approach for computing the DFT has the advantage of speedier computation of the DHT, and it does not have the complexity of optimizing methods. This method is especially attractive in case of a DIF FFT for real-valued input, since there are no easy methods available to compute the DFT of a real data efficiently from the conventional methods. The operation-counts for radix-4 and split-radix algorithms to compute the DHT also are less than the number of operations for the corresponding FFT algorithms.

Finally an example application was given, in which simulated Raman spectra were analyzed using a matched filter. The DHT approach took less computational effort than the DFT method.

REFERENCES

1. Cooley, J. W., and Tukey, J. W., 'An algorithm for machine computation of complex Fourier series,' Math. Comput., Vol. 19, pp. 297-301, Apr. 1965.

2. Bracewell, R. N., 'The discrete Hartley transform,' J. Opt. Soc. Amer., Vol. 73, pp. 1832-1835, Dec. 1983.

3. Hartley, R. V. L., 'A more symmetrical Fourier analysis applied to transmission problems,' Proc. IEEE., Vol. 30, pp. 144-150, Mar. 1942.

4. Wang, Z., 'Harmonic analysis with a real frequency function, I. Aperiodic case,' Appl. Math. Comput., Vol. 9, pp. 53-73, 1981

5. Wang, Z. 'Harmonic analysis with a real frequency function, II. Periodic and bounded case,' Appl. Math. Comput., Vol. 9, pp. 153-163, 1981.

6. Wang, Z. 'Harmonic analysis with a real frequency function, III. Data sequence,' Appl. Math. Comput., Vol. 9, pp. 245-255, 1981.

7. Bracewell, R. N., 'The fast Hartley transform,' Proc. IEEE, Vol. 72, pp. 1010-1018, Aug. 1984.

8. Kwong, C. P., and Shiu, K. P., 'Structured fast Hartley transform algorithms,' IEEE Trans. Acoust., Speech, and Signal Processing, Vol. ASSP-34, Aug. 1986.

9. Sorensen, H. V., Jones, D. L., Burrus, C. S, and Heideman, M. T., 'On computing the discrete Hartley transform,' IEEE Trans.

Acoust., Speech, and Signal Processing, Vol. ASSP-32, pp. 1231-1238, Oct. 1985.

10. Meckelburg, H. J., and Lipka, D. 'Fast Hartley transform Algorithm,' Electron. Letters, Vol. 21, pp. 341-343, Apr. 1985.

11. Prado, J., 'Comments on the fast Hartley Transform,' Proc. IEEE, Vol. 73, no. 12, pp. 1862-1863, Dec. 1985.

12. Duhamel, P., and Hollman, H., 'Split Radix FFT algorithm,' Electron. Letters, Vol. 20, no. 1, pp. 14-16, Jan. 5, 1984.

13. Soo-Chang Pei and Ja-Ling Wu, 'Split Radix Fast Hartley Transform, Electronics Letters, Vol. 22, no.1, pp. 26-27, 2nd Jan., 1986.

14. Burrus, C. S., and Parks, T. W., DFT/FFT and Convolution Algorithms, Wiley, New York, 1985.

15. Krishnan, R. S., and Shankar, R. K., 'Raman Effect: History of the Discovery,' J. Raman Spectroscopy, Vol. 10, pp. 1-8, Sergio Porto Commemorative Issue, 1981.

16. Peebles, P. Z., 'Probability, Random Variables and Random Signal Principles,' 1st Ed., pp. 222-225, McGraw-Hill, New York, 1980.

17. Dyer, S. A., and Hardin, D. S., 'Enhancement of Raman Spectra Obtained at Low Signal-to-Noise Ratios: Matched Filtering and Adaptive Peak Detection,' Applied Spectroscopy, Vol. 39, no. 4, pp. 655-662, 1985.

APPENDIX

```
*****************************************************************
*
*       Department of Electrical and Computer Engineering
*               Kansas State University
*       Vax Fortran Source Filename: MATCHED_FILTER.FOR
*
*****************************************************************
*
*       ROUTINE:        Mainline
*
*       DESCRIPTION: It enhances the Signal to Noise Ratio
*                    (SNR) of the input spectrum using the
*                    Matched Filtering method.
*                    Discrete Hartley Transform (DHT)
*                    technique is used to implement the matched
*                    filter
*
*      DOCUMENTATION
*      FILES:        None
*
*      RETURN:       Not Used
*
*      ROUTINES
*      CALLED:       DHT2,NPLOT
*
*      AUTHOR:       CHANDRA C. VARANASI
*
*      DATE CREATED: 22nd March 1987,  Version 1.0
*
***************************************************************
*
*     The following are some of the key variables used in
*     the routine
*
*     SPECTRUM: (Real) Array containing the input spectrum,
*                      zero padded to twice its original
*                      length
*
*      PROFILE:  (Real) Array containing the signal profile
*                       pulse,zero padded to twice its original
*                       length
*
*      OUTPUT:   (Real) Array containing the spectrum with
*                       enhanced Signal to Noise Ratio (SNR),
*                       resulting as the output from the
*                       matched filter.Length of the array is
*                       same as the original input spectral
*                       data length
*
```

```
***********************************************************************
       IMPLICIT NONE

       REAL SPECTRUM(0:2047),PROFILE(0:2047),X_DATA(0:2047),
     +     OUTPUT(0:1023),Y_DATA(0:2047)

       INTEGER  NUMBER,I

* Read the input spectral data into the array SPECTRUM

       CALL SGOPEN(5,'READ','NOPROMPT','SPECT.DAT','REAL',NUMBER)
       CALL SGTRAN (5,'READ','REAL',SPECTRUM,NUMBER)

               DO I = 0,2047
                    Y_DATA(I)=SPECTRUM(I)
                    X_DATA(I)=FLOAT(I)
               END DO

* Read the input signal profile pulse into the array PROFILE

       CALL SGOPEN (6,'READ','NOPROMPT','PRO.DAT','REAL',NUMBER)
       CALL SGTRAN (6,'READ','REAL',PROFILE,NUMBER)

* Take the DHT of the SPECTRUM

       CALL LIB INIT_TIMER
       CALL DHT2 (SPECTRUM,2048,11)

* Take the DHT of the signal PROFILE

       CALL DHT2 (PROFILE,2048,11)

* Multiply the above two DHTs

               DO I = 0,2047
                  SPECTRUM(I) = SPECTRUM(I)*PROFILE(I)
               END DO

* Take the inverse DHT of the sequence.In fact,inverse DHT and
* forward DHT are one and the same

       CALL DHT2 (SPECTRUM,2048,11)
       CALL LIB SHOW_TIMER

* Now,retain the first 1024 points

               DO I = 0,1023
                  OUTPUT(I) = SPECTRUM(I)
                  X_DATA(I) = FLOAT(I)
               END DO
```

```
* Plot the matched filter OUTPUT

                CALL SIMPLE_PLOT(1024,' ',
     +            'LINEAR',X_DATA,OUTPUT,'cm|t-1t|',
     +            'Raman Shift',' ','Amplitude')

                END
```

```
******************************************************************
*    Department of Electrical and Computer Engineering
*            Kansas State University
*    Vax Fortran Source Filename: MATCHED_FILTER_DFT.FOR
*
******************************************************************
*
*    ROUTINE:        Mainline
*
*    DESCRIPTION:    It enhances the Signal to Noise Ratio
*                    (SNR) of the input spectrum using the
*                    Matched Filtering method.
*                    Discrete Fourier Transform (DFT) is used
*                    to implement the matched filter
*
*    DOCUMENTATION
*    FILES:          None
*
*    RETURN:         Not Used
*
*    ROUTINES
*    CALLED:         FFT,NPLOT
*
*    AUTHOR:         CHANDRA C. VARANASI
*
*    DATE CREATED: 22nd March 1987,    Version 1.0
*
*****************************************************************
*    The following are some of the key variables used in
*    the routine
*
*    SPECTRUM: (Complex) Array containing the input spectrum,
*                        (zero padded to twice its original
*                        length), as its real part.Imaginary
*                        part is zero.
*
*    PROFILE:  (Complex) Array containing the signal profile
*                        pulse,(zero padded to twice its
*                        original length), as its real part.
*                        Imaginary part is zero
*
*    OUTPUT:   (Real)    Array containing the spectrum with
*                        enhanced Signal to Noise Ratio (SNR),
*                        resulting as the output from the
*                        matched filter.Length of the array is
*                        same as the original spectral length
*
*****************************************************************
```

```
      IMPLICIT NONE

      COMPLEX  SPECTRUM(0:2047),PROFILE(0:2047)

      REAL OUTPUT(0:1023),X_DATA(0:1023),A(0:2047),B(0:2047)

      INTEGER   NUMBER,I

      CALL LIB INIT_TIMER

* Read the input spectral data into the real array A

      CALL SGOPEN(5,'READ','NOPROMPT','SPECT.DAT','REAL',NUMBER)
      CALL SGTRAN (5,'READ','REAL',A,NUMBER)

* Read the zero padded profile pulse into the real array B

      CALL SGOPEN (6,'READ','NOPROMPT','PRO.DAT','REAL',NUMBER)
      CALL SGTRAN (6,'READ','REAL',B,NUMBER)

* Transfer the data and profile into the complex arrays SPECTRUM
* and PROFILE respectively

      DO I = 0,2047
          PROFILE(I)= B(I)
          SPECTRUM(I) = A(I)
      END DO

* Take the DFT of the input SPECTRUM
      CALL LIB INIT_TIMER
      CALL FFT (SPECTRUM,2048,0)

* Take the conjugate of the transformed SPECTRUM

      DO I = 0,2047
         SPECTRUM(I) = CONJG(SPECTRUM(I))
      END DO

* Take the DFT of the signal PROFILE

      CALL FFT (PROFILE,2048,0)

* Multiply  X *(K) and H(K) where:

* X(K)  ---- DFT of the zero padded input spectrum
* H(K)  ---- DFT of the zero padded signal profile pulse

      DO I = 0,2047
         SPECTRUM(I) = SPECTRUM(I)*PROFILE(I)
      END DO
```

```
* Take the inverse DFT of the product

       CALL FFT (SPECTRUM,2048,1)

       CALL LIB SHOW_TIMER

* Now retain only 1024 points of the data, since that is the
* length of the original spectrum

      DO I = 1,1023
         OUTPUT(I) =REAL(SPECTRUM(2048-I))
         X_DATA(I) = REAL(I)
      END DO

* Plot the matched filter output

      CALL SIMPLE_PLOT(1024,'Matched Filter Output using DFT',
      +             'LINEAR',X_DATA,OUTPUT,'cm|t-1t|',
      +             'Raman Shift',' ','Amplitude')
      END
```

```
****************************************************************
*
*    Department of Electrical and Computer Engineering
*            Kansas State University
*    Vax Fortran Source Filename:  PROFILE.FOR
*
****************************************************************
*
*    ROUTINE:        PROFILE.FOR
*
*    DESCRIPTION:    It generates a Lorentzian pulse centered
*                    at zero.  After that, the array
*                    containing the pulse is zero padded
*                    to twice the pulse length.  Zeros are
*                    added in the middle of the data stream
*                    instead of at the end.  The zero padded
*                    pulse will be used as the profile in the
*                    convolution
*
*    DOCUMENTATION
*    FILES:          None
*
*    RETURN:         Not Used
*
*    ROUTINES
*    CALLED:         None
*
*    AUTHOR:         CHANDRA C. VARANASI
*
*    DATE CREATED: 30th March 1987    Version 1.0
*
****************************************************************
*
*    The following are the key variables used in the routine
*
*    AMPLITUDE:      (Real)  The amplitude of the Lorentzian
*                            peak
*
*    ALPHA:          (Real)  The half_width at half_height
*                            of the pulse
*
*    PULSE:          (Real)  Array containing the pulse
*
*    PROFILE:        (Real)  Array containing the zero padded
*                            pulse
*
****************************************************************
```

```
      IMPLICIT NONE

      REAL AMPLITUDE, ALPHA, PULSE(-512:511), PROFILE(0:2047),
      +    X_DATA(0:2047)

      INTEGER    I

      PARAMETER (AMPLITUDE = 2.0, ALPHA = 20.0)

*  Generate the Lorentzian pulse

      DO I = -512,511
            PULSE(I) = AMPLITUDE/(1+ (I/ALPHA)**2)
      END DO

*  Now, zero pad the profile upto twice the number of points
*  bringing the negative part of the pulse to the end of the
*  data stream and store in the array PROFILE

      DO I = -512,-1
        PROFILE(2048+I) = PULSE(I)
        PROFILE(I+512)  = PULSE(I+512)
      END DO

*  Write PROFILE to the disk

      CALL SGOPEN (5,'WRITE','NOPROMPT','PRO.DAT','REAL',2048)
      CALL SGTRAN (5,'WRITE','REAL',PROFILE,2048)

*  Plot the zero_padded profile.  For that set up the X-axis data

      DO I = 0,2047
           X_DATA(I) = REAL(I)
      END DO

*  Now plot

      CALL SIMPLE_PLOT(2048,'Zero Padded Profile','LINEAR',
      +              X_DATA,PROFILE,'cm|t-1t|','Raman Shift',
      +              ' ','Amplitude')

      END
```

```
*****************************************************************
*
*    Department of Electrical and Computer Engineering
*            Kansas State University
*    Vax Fortran Source Filename:  NOISE_PLOT.FOR
*
*****************************************************************
*
*    ROUTINE:        NOISE_PLOT.FOR
*
*    DESCRIPTION:    It plots the zero mean unit variance
*                    white Gaussian noise generated using
*                    RALPH, a general purpose signal
*                    processing software developed at the
*                    Kansas State University.
*
*    DOCUMENTATION
*    FILES:          None
*
*    RETURN:         Not Used
*
*    ROUTINES
*    CALLED:         SIMPLE_PLOT
*
*    AUTHOR:         CHANDRA C. VARANASI
*
*    DATE CREATED: 30th March 1987    Version 1.0
*
*****************************************************************
*
*    The following are the key variables used in the routine
*
*    GAUSSIAN NOISE:     (Real)Array containing the samples of
*                              the zero mean unit variance Gaussian
*                              noise.
*
*****************************************************************

     IMPLICIT NONE

     REAL       X_DATA(0:1023), GAUSSIAN NOISE(0:1023)

     INTEGER    I, NUMBER

     PARAMETER  (AMPLITUDE = 2.0, ALPHA = 20.0)

* Read the noise samples from the disk

     CALL SGOPEN (5, 'READ','NOPROMPT','NOISE.DAT','REAL',
     +               NUMBER)
```

```
      CALL SGTRAN (5,'READ','REAL',GAUSSIAN_NOISE,NUMBER)

      DO I = 0, 1023
            X_DATA(I) = FLOAT(I)
      END DO

* Plot the noise

      CALL SIMPLE_PLOT(1024,'White Gaussian noise','LINEAR',
      +                 X_DATA,GAUSSIAN_NOISE, 'Cm|t-1t|',
      +                 'Raman shift', ' ', 'Amplitude')

      END
```

```
**********************************************************************
*
*    Department of Electrical and Computer Engineering
*        Kansas State University
*    Vax Fortran Source Filename: SPECTRUM.FOR
*
**********************************************************************
*
*    ROUTINE:        SPECTRUM.FOR
*
*    DESCRIPTION:    It reads the input signal data file as
*                    well as the noise data file and then
*                    generates the input spectrum by mixing
*                    the above two files
*
*    DOCUMENTATION
*    FILES:          None
*
*    RETURN:         Not Used
*
*    ROUTINES
*    CALLED:         SIMPLE_PLOT
*
*    AUTHOR:         CHANDRA C. VARANASI
*
*    DATE CREATED: 1st April 1987     Version 1.0
*
**********************************************************************
*
*    The following are the key variables used in the routine
*
*    SPECTRUM:       (Real)  Array into which the input signal
*                            is read.  When the noise is added
*                            to it, it contains the simulated
*                            input spectrum to the matched
*                            filter
*
*    NOISE:          (Real)  Array into which the white
*                            Gaussian noise, generated using
*                            RALPH, is read
*
**********************************************************************

     IMPLICIT    NONE

     REAL SPECTRUM(0:2047),NOISE(0:2047),X_DATA(0:2047)

     INTEGER    NUMBER, I

* Read the input signal into SPECTRUM.  Input signal is a
* mixture of two Lorentzian pulses.

     CALL SGOPEN (7,'READ','NOPROMPT','SIGNAL.DAT','REAL',
     +              NUMBER)
     CALL SGTRAN (7,'READ','REAL',SPECTRUM,NUMBER)

* Read the zero mean unit variance white Gaussian noise generated
* using RALPH

     CALL SGOPEN (5,'READ','NOPROMPT','NOISE.DAT','REAL',
     +              NUMBER)
     CALL SGTRAN (5,'READ','REAL',NOISE,NUMBER)

* Add the two arrays SPECTRUM and NOISE to generate the input
* spectrum and place the result in SPECTRUM

     DO I = 0,2047
         SPECTRUM(I) = SPECTRUM(I)+NOISE(I)
         X_DATA(I) = REAL(I)
     END DO

* Write the input spectrum to the disk

     CALL SGOPEN (6,'WRITE','NOPROMPT','SPECT.DAT','REAL',2048)
     CALL SGTRAN (6,'WRITE','REAL',SPECTRUM,2048)

* Plot the input spectrum

     CALL SIMPLE_PLOT(2048,'Input Spectrum','LINEAR',X_DATA,
     +        SPECTRUM,'cm|t-1t|','Raman Shift',' ','Amplitude')


     END
```

```
****************************************************************
*
*    Department of Electrical and Computer Engineering
*                  Kansas State University
*       Vax Fortran Source Filename:      LORENTZ.FOR
*
****************************************************************
*
*    ROUTINE:          LORENTZ.FOR
*
*    DESCRIPTION:      It generates a lorentzian shaped pulse of
*                      desired amplitude and half width half
*                      height
*
*    DOCUMENTATION
*    FILES:            None
*
*    RETURN:           Not Used
*
*    ROUTINES
*    CALLED:           None
*
*    AUTHOR:           CHANDRA C.VARANASI
*
*    DATE CREATED:     23rd March, 1987    Version 1.0
*
****************************************************************
*
*    The following are some of the key variables used in the
*    routine
*
*    PEAK:                (Real) The amplitude of the
*                                lorentzian peak
*
*    ALPHA:               (Real) The half_width at half_height
*
*    CENTER_FREQUENCY:    (Real) The frequency at which the
*                                lorentzian pulse attains its
*                                peak
*
*    LORENTZ1,LORENTZ2:   (Real) Arrays that contain the
*                                lorentzian pulses at two
*                                distinct center frequencies
*
*    LORENTZ:             (Real) Array that contains the
*                                summation of LORENTZ1 and
*                                LORENTZ2
*
****************************************************************
```

```
        IMPLICIT NONE

        REAL         PEAK, ALPHA, CENTER_FREQUENCY1,
     +               CENTER_FREQUENCY2,LORENTZ1(0:1023),
     +               FREQUENCIES(0:1023), LORENTZ2(0:1023),
     +               LORENTZ(0:1023),
     +               X_DATA(0:1023)

        INTEGER      I

        PARAMETER    (PEAK = 2.0, ALPHA = 20.0,
     +               CENTER_FREQUENCY1 = 250.0,
     +               CENTER_FREQUENCY2 = 650.0)

****************************************************************

        DO I = 0,1023

            LORENTZ1(I) = PEAK/(1.0+((I - CENTER_FREQUENCY1)/
     +                        ALPHA)**2)
            LORENTZ2(I) = PEAK/(1.0+((I - CENTER_FREQUENCY2)/
     +                        ALPHA)**2)

        END DO

* Add LORENTZ1 and LORENTZ2  that is going to be the signal

        DO I = 0,1023
          LORENTZ(I) = LORENTZ1(I)+LORENTZ2(I)
          X_DATA(I)  = REAL(I)
          FREQUENCIES(I) = REAL(I)
        END DO

* Plot the signal

        CALL SIMPLE_PLOT(1024,'Input Signal','LINEAR',X_DATA,
     +        LORENTZ,'cm|t-1t|','Raman Shift',' ',
     +        'Amplitude')

* Write the signal to the disk
        CALL SGOPEN (6,'WRITE','NOPROMPT','SIGNAL.DAT','REAL',
     +             1024)
        CALL SGTRAN (6,'WRITE','REAL',LORENTZ,1024)

        END
```

```
**********************************************************************
*
*    Department of Electrical and Computer Engineering
*                Kansas State University
*    Vax Fortran source filename:      SIMPLE_PLOT.FOR
*
**********************************************************************
*
*    ROUTINE: SUBROUTINE
*             SIMPLE_PLOT (NUM_POINTS,PLOT_TITLE,
*             PLOT_TYPE,X_DATA,Y_DATA,X_AXIS_UNITS,
*             X_AXIS_TITLE,Y_AXIS_UNITS,Y_AXIS_TITLE)
*
*    DESCRIPTION:      Plots X_DATA and Y_DATA values as
*                      abscissae and ordinates respectively, and
*                      places a TITLE and UNITS on the corresp-
*                      onding axes.  PLOT_TYPE specifies the
*                      type of plot being generated,and finally
*                      the plot is given a PLOT_TITLE.
*                      The routine gives the user the choice of
*                      plotting on either the screen or the HP
*                      plotter.
*
*    DOCUMENTATION
*    FILES:            None
*
*    VARIABLES IN
*    THE ARGUMENT:
*
*      NUM_POINTS:     (input)  integer
*                      Number of data points to be plotted
*
*      PLOT_TITLE:     (input)  character*(*)
*                      Title to be placed on plot
*
*      PLOT_TYPE:      (input)  character*(*)
*                      Character string specifying the type of
*                      plot to be generated.  The following are
*                      valid:
*                      'LINEAR'        for linear-linear
*                      'LOG-LINEAR'    for log-linear
*                      'LINEAR-LOG'    for linear-log
*                      'LOG-LOG'       for log-log
*
*      X_DATA:         (input)   real
*                      Array of abscissae values to be plotted
*
*      Y_DATA:         (input)   real
*                      Array of ordinate values to be plotted
*
```

```
*      X_AXIS_UNITS: (input)   character*(*)
*                    Name to be given to the units associated
*                    with the X_AXIS
*
*      X_AXIS_TITLE: (input)   character*(*)
*                    Title to be placed on the X_AXIS
*
*      Y_AXIS_UNITS: (input)   character*(*)
*                    Name to be given to units associated with
*                    Y_axis
*
*      Y_AXIS_TITLE: (input)   character*(*)
*                    Title to be placed on the Y_axis
*
*    RETURN:         Not used
*
*    ROUTINES
*    CALLED:         PAXIS
*                    PCHRPL
*                    PCLOSP
*                    PINIT
*                    PLGAXS
*                    PLGLIN
*                    PLGLOG
*                    PLINE
*                    PLOGSC
*                    PLNLOG
*                    PORIG
*                    PPLOT
*                    PSCALE
*                    PSTCHR
*                    PSTVEL
*                    PTEXT
*                    PTXTLN
*                    PWIND
*
*    AUTHOR:         Chandra C. Varanasi
*
*    DATE CREATED:   2nd September 1986
*
*    REVISIONS:      8th September 1986
*                    Plotting log-log curve is added
*                    13th September 1986
*                    Offsetting the plot_title is added
******************************************************************
*
*    The subroutine SIMPLE_PLOT calls some routines from
*    THE P SYSTEM OF GENERALIZED PLOT SUBROUTINES.  The
*    following are the key variables required to call the
*    above routines.
*    CLENX:             (Output from subroutine PLOGSC)   real
*                       Length in cm. of one log cycle
*                       1.0 <= CLEN <= Length of X_axis
*
*    CLENY:             (Output from subroutine PLOGSC)
*                       Length in cm. of one log cycle
*                       1.0 <= CLEN <= Length of Y_Axis
*
*    DELTAX:            (output from subroutine PSCALE)   real
*                       Scale factor (increment in X between tic
*                       marks)
*
*    DELTAY:            (output from the subroutine PSCALE) real
*                       Scale factor (increment in Y between tic
*                       marks)
*
*    DIVLENX:           (output from subroutine PSCALE)    real
*                       Space between X_axis tic marks in user
*                       units
*
*    DIVLENY:           (output from subroutine PSCALE)   real
*                       Space between Y_axis tic marks in user
*                       units
*
*    FIRDEL:            (input to subroutine PLINE)    real
*                       A four element array containing the
*                       following:
*
*                       FIRSTX:      Starting value of X_data
*                       DELTAX:      Described above
*                       FIRSTY:      Starting value of Y
*                       DELTAY:      Described above
*
*    FIRLEN:            (input to subroutine PLINE)    real
*                       A four element containing the following:
*                       FIRSTX:      Starting value of X_DATA
*                       DELTAX:      Described above
*                       FIRSTY:      Starting value of Y_DATA
*                       CLENY:       Described above
*
*    LENSTR:            (output from subroutine PTXTLN)   integer
*                       Number of characters in PLOT_TITLE
*
*    NEGFLGX:           (output from subroutine PLOGSC)  integer
*                       Flag to warn that negative values were
*                       encountered in X_DATA
*                       0:  No negative values in X_DATA
*                       1:  Negative values in X_DATA
*    NEGFLGY:            (output from subroutine PLOGSC)  integer
*                        Flag to warn that negative values were
*                        encountered in Y_DATA
*
*    OFFSET:             (input to the routine PPLOT)   real
*                        Margin on the left side of the title)
*
****************************************************************

          SUBROUTINE        SIMPLE_PLOT(NUM_POINTS,PLOT_TITLE,
     +                      PLOT_TYPE,X_DATA,Y_DATA,
     +                      X_AXIS_UNITS,X_AXIS_TITLE,
     +                      Y_AXIS_UNITS,Y_AXIS_TITLE)

*    Declare the variables

          IMPLICIT        NONE

          INTEGER         CHOICE,I,NUM_POINTS,NEGFLGX,NEGFLGY,
     +                    LENSTR

          REAL            CLENX, CLENY, DELTAX, DELTAY, DIVLENX,
     +                    DIVLENY, FIRDEL(4), FIRLEN(4), FIRSTX,
     +                    FIRSTY, OFFSET, X_DATA(0:*), Y_DATA(0:*)

          CHARACTER       PLOT_TITLE*(*), PLOT_TYPE*(*),
     +                    X_AXIS_UNITS*(*),X_AXIS_TITLE*(*)
     +                    Y_AXIS_UNITS*(*), Y_AXIS_TITLE*(*)

****************************************************************

       PRINT*, ' SELECT THE PLOTTING DEVICE (ENTER 1 OR 2)'
       PRINT*
       PRINT*, ' 1.  Tektronix 4014 display'
       PRINT*,'  2.  HP 7475A Plotter'

* Read the user's choice of the plotting device

        READ*, CHOICE

* Initiate the corresponding device

        IF (CHOICE .EQ. 1) THEN
             CALL PINIT(4014,' ',1.0,'A')
        ELSE
             CALL PINIT(7475,' ',1.0,'A')
        END IF
```

```
* Set pen velocity

        CALL PSTVEL (3.0)

* Establish an origin

         CALL PORIG (4.54, 4.0)

* Establish the bounds of the plot

         CALL PWIND (0.0,0.0,0.0,0.0)

         IF (PLOT_TYPE .EQ. 'LINEAR') THEN

* Scale both X_DATA and Y_DATA

         CALL PSCALE (X_DATA,NUM_POINTS,18.0,FIRSTX,DELTAX,
     +                DIVLENX)
         CALL PSCALE (Y_DATA,NUM_POINTS,12.0,FIRSTY,DELTAY,
     +                DIVLENY)

* Fill in the array FIRDEL

         FIRDEL(1) = FIRSTX
         FIRDEL(2) = DELTAX
         FIRDEL(3) = FIRSTY
         FIRDEL(4) = DELTAY
*
* Draw the linear axes

         CALL PAXIS(0.0,0.0,X_AXIS_TITLE,X_AXIS_UNITS,220,
            2010,18.0,0.0,FIRSTX,DELTAX,DIVLENX)

         CALL PAXIS(0.0,0.0,Y_AXIS_TITLE,Y_AXIS_UNITS,120,
            1010,12.0,90.0,FIRSTY,DELTAY,DIVLENY)

* Draw the linear-linear curve

         CALL PLINE (X_DATA,Y_DATA,NUM_POINTS,FIRDEL,1,' ',
     +             DIVLENX,DIVLENY)

         END IF

         IF (PLOT_TYPE .EQ. 'LOG-LINEAR') THEN

* Scale the X and Y axes data

         CALL PSCALE(X_DATA,NUM_POINTS,18.0,FIRSTX,DELTAX,
     +               DIVLENX)
```

```
          CALL PLOGSC(Y_DATA,NUM_POINTS,12.0,FIRSTY,CLENY,NEGFLGY)

* Draw the logarithmic Y and linear X axes

          CALL PAXIS(0.0,0.0,X_AXIS_TITLE,X_AXIS_UNITS,220,2010,
     +               18.0,0.0,FIRSTX,DELTAX,DIVLENX)
          CALL PLGAXS(0.0,0.0,Y_AXIS_TITLE,Y_AXIS_UNITS,-1010,
     +                12.0,90.0,FIRSTY,CLENY)

* Fill in the array FIRLEN

        FIRDEL(1) = FIRSTX
        FIRDEL(2) = DELTAX
        FIRDEL(3) = FIRSTY
        FIRDEL(4) = CLENY

* Draw the log-linear curve

          CALL PLGLIN(X_DATA,Y_DATA,NUM_POINTS,FIRLEN,1,' ',
     +                  DIVLENX)

      END IF


      IF (PLOT_TYPE .EQ. 'LINEAR-LOG') THEN

* Scale the data
          CALL PLOGSC(Y_DATA,NUM_POINTS,18.0,FIRSTX,CLENX,NEGFLGX)
          CALL PSCALE(Y_DATA,NUM_POINTS,12.0,FIRSTY,DELTAY,
     +                   DIVLENY,)


* Draw the axes

          CALL PAXIS (0.0,0.0,Y_AXIS_TITLE,Y_AXIS_UNITS,120,1010,
     +              12.0,90.0,FIRSTY,DELTAY,DIVLENY)
          CALL PLGAXS(0.0,0.0,X_AXIS_TITLE,X_AXIS_UNITS,2010,18.0,
     +                0.0,FIRSTX,CLENX)

*
* Fill in the array FIRLEN

        FIRDEL(1) = FIRSTX
        FIRDEL(2) = CLENX
        FIRDEL(3) = FIRSTY
        FIRDEL(4) = DELTAY

* Draw the linear-log curve

               CALL PLNLOG(X_DATA,Y_DATA,NUM_POINTS,FIRLEN,1,' ',
      +               DIVLENY)

      END IF

      IF (PLOT_TYPE .EQ. 'LOG-LINEAR') THEN
*
* Scale the data

      CALL PLOGSC(X_DATA,NUM_POINTS,18.0,FIRSTX,CLENX,NEGFLGX)
      CALL PLOGSC(YX_DATA,NUM_POINTS,12.0,FIRSTY,CLENY,NEGFLGY)

* Draw the logarithmic axes
      CALL PLGAXS(0.0,0.0,X_AXIS_TITLE,X_AXIS_UNITS,2010,18.0,
      +               0.0,FIRSTX,CLENX)
      CALL PLGAXS(0.0,0.0,Y_AXIS_TITLE,Y_AXIS_UNITS,-1010,12.0,
      +               90.0,FIRSTY,CLENY)

* Fill in the array FIRLEN

        FIRDEL(1) = FIRSTX
        FIRDEL(2) = CLENX
        FIRDEL(3) = FIRSTY
        FIRDEL(4) = DELTAY

* Plot the log-log curve

        CALL PLGLOG(X_DATA,Y_DATA,NUM_POINTS,FIRLEN,1,' ')

      END IF

* Calculate the length of the plot_title

      CALL PTXTLN (PLOT_TITLE,LENSTR)

* Set the character size

      CALL PSTCHR (0.3,0.4,10.0)

* Set the offset

      OFFSET = (18.0 - LENSTR*0.3)1.5)/2.0

* Move the pen to the position where the plot_title is to start

      CALL PPLOT(OFFSET, 13.0, 0)

* Plot the title

      CALL PTEXT(PLOT_TITLE)
```

```
* Close the plotting device

      CALL PCLOSP
      END
```

THE DISCRETE HARTLEY TRANSFORM

by

CHANDRA CHUDA VARANASI

B.Tech., Jawaharlal Nehru Technological University, India

1983

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

Kansas State University
Manhattan, Kansas

1988

ABSTRACT

The discrete Fourier transform (DFT) is widely used in various applications in digital signal procesing (DSP). However, the DFT is a complex-valued transform and, as such, it transforms real-valued sequences into complex-valued transform sequences. Also, the inverse DFT is different from the forward DFT. In contrast, the discrete Hartley transform (DHT) is an inherently real-valued transform and is also symmetric.

Various properties of the Hartley transform (HT) and the DHT are derived in this study and compared with the well-known properties of the Fourier transform (FT) and the DFT. Decomposition formulas for different fast algorithms to compute the DHT are derived. The algorithms include the decimation-in-time (DIT) and decimation-in-frequency (DIF) radix-2, radix-4 and split-radix algorithms. Computational cost in terms of number of multiplications and additions is derived for all the algorithms and compared. Finally, an application of the DHT in the analysis of Raman spectra is given, wherein it is used to implement the matched filter. The matched filter was implemented via the DFT also. Both filter response and computation time were compared for the two methods. The DHT approach proved faster in terms of computation time, but the filter response was insensitive to the choice of the transform.