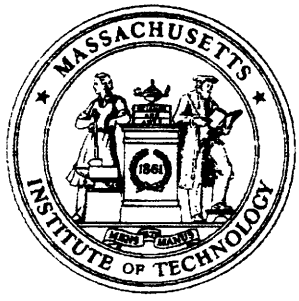


Error Properties of Hartley Transform Algorithms

RLE Technical Report No. 511

October 1985

Avideh Zakhor

Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, MA 02139 USA

This work was supported in part by the Advanced Research Projects Agency monitored by ONR under contract N00014-81-K-0742 NR-049-506 and in part by the National Science Foundation under Grant ECS84-07285.

Massachusetts Institute of Technology
Research Laboratory of Electronics
Department of Electrical Engineering
and Computer Science
Room 36-615
Cambridge, MA 02139

ERROR PROPERTIES OF HARTLEY TRANSFORM ALGORITHMS

Avideh Zakhor

Technical Report No. 511

October 1985

This work was supported in part by the Advanced Research Projects Agency monitored by ONR under contract N00014-81-K-0742 NR-049-506 and in part by the National Science Foundation under Grant ECS-8407285.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Research Laboratory of Electronics Massachusetts Institute of Technology 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research Mathematical and Information Scien. Div.
6c. ADDRESS (City, State and ZIP Code) 77 Massachusetts Avenue Cambridge, MA 02139	7b. ADDRESS (City, State and ZIP Code) 800 North Quincy Street Arlington, Virginia 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Advanced Research Projects Agency 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-81-K-0742
8c. ADDRESS (City, State and ZIP Code) 1400 Wilson Boulevard Arlington, Virginia 22217	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: Algorithms PROJECT NO.: TASK NO.: NR 049-506 WORK UNIT NO.:
11. TITLE (Include Security Classification) Error Properties of Hartley Transform	
12. PERSONAL AUTHOR(S) Avideh Zakhor	
13a. TYPE OF REPORT Technical 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) October 1985 15. PAGE COUNT 203
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

In this thesis, the error properties of various discrete Hartley transform (DHT) algorithms are investigated theoretically and experimentally. More specifically, we analyze the arithmetic roundoff error characteristics of DHT algorithms proposed by Bracewell and Wang and develop and analyze a new DHT algorithm.

Statistical models for roundoff errors and linear system noise theory are employed to estimate output noise variance for these DHT algorithms. By considering the overflow constraint in conjunction with these noise analyses, output noise to signal ratios are derived for both fixed and floating-point arithmetic. Experiments are used to support the theoretical predictions obtained via the statistical models. The

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Kyra M. Hall RLE Contract Reports	22b. TELEPHONE NUMBER (Include Area Code) (617) 253-2569 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

empirical results are found to be in excellent agreement with the predictions based on the models.

Comparing Bracewell's, Wang's and the new algorithm in terms of their error properties, we find that Bracewell's algorithm exhibits the most desirable error characteristics. These results were found to hold for both decimation-in-time and frequency and for a variety of different radices. For a given radix, the total operation count for all the algorithms investigated in this thesis are found to be identical.

Error Properties of Hartley Transform Algorithms

by

Avideh Zakhor

Submitted to the Department of Electrical Engineering
and Computer Science on June 3, 1985
in partial fulfillment
of the requirements for the degree of
Master of Science

Abstract

In this thesis, the error properties of various discrete Hartley transform (DHT) algorithms are investigated theoretically and experimentally. More specifically, we analyze the arithmetic roundoff error characteristics of DHT algorithms proposed by Bracewell and Wang and develop and analyze a new DHT algorithm.

Statistical models for roundoff errors and linear system noise theory are employed to estimate output noise variance for these DHT algorithms. By considering the overflow constraint in conjunction with these noise analyses, output noise to signal ratios are derived for both fixed and floating-point arithmetic. Experiments are used to support the theoretical predictions obtained via the statistical models. The empirical results are found to be in excellent agreement with the predictions based on the models.

Comparing Bracewell's, Wang's and the new algorithm in terms of their error properties, we find that Bracewell's algorithm exhibits the most desirable error characteristics. These results were found to hold for both decimation-in-time and frequency and for a variety of different radices. For a given radix, the total operation count for all the algorithms investigated in this thesis are found to be identical.

Thesis Supervisor: Prof. Alan V. Oppenheim
Title: Professor of Electrical Engineering

To My Parents

For Their Love, Support and Understanding

ACKNOWLEDGEMENTS

I would like to express my warmest and deepest gratitude to my thesis supervisor Al Oppenheim for the original problem suggestion and his guidance, encouragement and suggestions throughout the course of this thesis. His concern for his students is simply unparalleled. Working with him has been a great privilege and it has made my first research experience a most enjoyable one.

I would like to thank Tom Bordley, Webster Dove and Dennis Martinez and the rest of the DSPG members for introducing me to the computer facilities. Technical discussions with Sue Curtis and Mike Wengrovitz have been particularly helpful. I also wish to acknowledge the financial support given to me by the Hertz Foundation during my graduate studies.

Finally, I would like to thank my parents for their love and for tolerating their daughter's long period of absence from home.

TABLE OF CONTENTS

CHAPTER 1: Introduction

The continuous-time Hartley transform was first proposed by R. V. Hartley [2] in 1942 in the context of transmission problems. Many of the concepts behind Fourier theory such as discrete and continuous-time Fourier series and transforms can be directly applied to the Hartley domain. In particular, Bracewell has recently proposed the discrete Hartley transform (DHT) which is essentially the counterpart of the discrete Fourier transform (DFT).

The real and imaginary parts of the DFT can be obtained from the even and odd parts of the DHT. Therefore discrete Hartley and Fourier transforms can be easily computed from each other. In addition, as we will see in chapter 2, for every property of the DFT, there is a corresponding one for the DHT. However the DHT has two important characteristics that are different from those of the DFT; since it is a real transform it uses only real arithmetic. Second the inverse DHT is identical to the forward DHT (within a scale factor). These characteristics make the DHT an attractive substitute for the DFT in many signal processing applications such as spectral analysis and convolutions. For example, the power spectra can be obtained from the DHT without first calculating the real and imaginary parts of the DFT as in the usual way of calculating the power spectra.

The DHT has fast algorithms similar in style to the FFT, the first of these proposed by Bracewell [4]. Some of these DHT algorithms can be used to compute the DFT more efficiently that the FFT. The fundamental principle that all these algorithms are based upon is that of decomposing the computation of the discrete

Hartley transform of a sequence of length N into successively smaller discrete Hartley transforms as is also the case with the FFT. The manner in which this principle is implemented leads to a variety of algorithms with different computational efficiencies and error properties. In this thesis we will analyze the arithmetic roundoff error characteristics of DHT algorithms proposed by Bracewell and Wang in addition to a new DHT algorithm.

Chapter 3 reviews the basic idea behind Bracewell's original decimation-in-time radix 2 algorithm. Decimation-in-frequency, radix 4 and split radix implementations of Bracewell's algorithm, proposed by Burrus [6], are also described in chapter 3. In chapter 4, we will review Wang's algorithm. Chapter 5 describes the new algorithm we have developed for computing the DHT with decimation-in-time and frequency, radix 2, radix 4 and split radix realizations. In addition, a chirp Hartley transform (CHT) algorithm similar to the chirp z-transform (CZT) algorithm is described in chapter 5.

The effects of quantization on fixed-point and floating-point implementations of the algorithms described in chapters 3 through 5 are studied in some detail in chapters 6 and 7. In general, effects of quantization on implementation of the DHT algorithms are sources of two kinds of error: errors due to coefficient quantization and errors due to rounding in computation. In this thesis we are only concerned with errors due to rounding in computation. In chapter 6, statistical models for roundoff errors and linear system noise theory are employed to estimate output noise variance in various DHT algorithms. By considering the overflow constraint

in conjunction with these noise analyses, output noise to signal ratios are derived. Noise to signal ratio analyses are carried out for both fixed and floating pint arithmetic.

The statistical models used in our noise analysis can not in general be verified theoretically and thus one must resort to experimental noise measurements to support the predictions obtained via the models. The experimental results are presented in chapter 8 and are found to be in excellent agreement with the theoretical predictions based on the statistical models. In chapter 8, all the DHT algorithms of the previous chapters are compared in terms of their error properties and their computational efficiencies. Chapter 9 concludes this thesis with suggestions for future research.

CHAPTER 2: The Discrete Hartley Transform: Definition and Properties

In this chapter, we will begin by defining the continuous and discrete Hartley transforms and exploring their relationships with the Fourier transform. Then the properties of the discrete Hartley transform (DHT) are described in detail. As we will see, since the Hartley and Fourier transforms are related to each other, their properties are somewhat similar. In addition, the DHT has fast algorithms similar in style to the FFT, the first of these proposed by Bracewell. These characteristics make the DHT an attractive substitute for the discrete Fourier transform (DFT) in many signal processing applications such as spectral analysis and linear filtering.

2.1. The Hartley transform

The Hartley transform was first proposed by R. V. Hartley [2] in 1942 in the context of transient and steady state transmission problems. The Hartley transform $H(\omega)$ of a real function $x(t)$ is defined as:

$$H(\omega) = \int_{-\infty}^{+\infty} x(t) \, [\cos(\omega t) + \sin(\omega t)] \, dt \tag{2.1}$$

Comparing the definition of the Hartley transform with that of the Fourier transform given by

$$F(\omega) = \int_{-\infty}^{+\infty} x(t) \, [\cos(\omega t) - j \sin(\omega t)] \, dt \tag{2.2}$$

we see that the two transforms are closely related to each other to a great extent. In fact, since cosine is an even function and sine is an odd function, the even part of the Hartley transform corresponds to the real part of the Fourier transform and

the odd part of the Hartley transform corresponds to the negative of the imaginary part of the Fourier transform. Moreover, the even and odd parts of $H(\omega)$ correspond to the even and odd parts of $x(t)$ respectively; i.e.

$$F_R(\omega) = \frac{H(\omega) + H(-\omega)}{2} = HT\left[\frac{x(t) + x(-t)}{2}\right] = \int_{-\infty}^{+\infty} x(t)\cos(\omega t)dt \tag{2.3a}$$

$$-F_I(\omega) = \frac{H(\omega) - H(-\omega)}{2} = HT\left[\frac{x(t) - x(-t)}{2}\right] = \int_{-\infty}^{+\infty} x(t)\sin(\omega t)dt \tag{2.3b}$$

where $F_R(\omega)$ and $F_I(\omega)$ denote the real and imaginary parts of the Fourier transform respectively and HT stands for the continuous-time Hartley transform. Using equation (2.3), the relationship between the Hartley and Fourier transforms can be summarized as:

$$H(\omega) = F_R(\omega) - F_I(\omega) \tag{2.4a}$$

$$F(\omega) = \frac{H(\omega) + H(-\omega)}{2} - j\,\frac{H(\omega) - H(-\omega)}{2} \tag{2.4b}$$

One of the differences between the Fourier and Hartley transform is the fact that the Hartley transform is its own inverse. That is, the original time function can be obtained by taking the Hartley transform of $H(\omega)$.

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} H(\omega)\, cas(\omega t)dt \tag{2.5}$$

where $cas(\theta)$ as originally defined by Hartley is:

$$cas(\theta) = \cos(\theta) + \sin(\theta)$$

In a completely analogous manner to the Fourier transform, the continuous-time Hartley transform can be used to derive, the continuous-time Hartley series, the discrete-time Hartley transform (DTHT), the discrete Hartley series (DHS) and the discrete Hartley transform (DHT). As shown in [1], there are a number of

points of view that can be taken toward the derivation and interpretation of the DFT presentation of finite duration sequences. As we will see, this is also the case for the DHT. More specifically, one approach in deriving the DHT of an N-point real sequence $x(n)$ is to view the DHT as one period of the discrete Hartley series representation of a periodic sequence for which each period is identical to the finite length sequence $x(n)$. Another approach is to consider the DHT to be equally spaced samples of the discrete-time Hartley transform. That is, if we define the discrete-time Hartley transform of $x(n)$ to be

$$H(\omega) = \sum_{n=0}^{N-1} x(n)\, cas(\,\omega n\,) \tag{2.6}$$

then the DHT of $x(n)$ can be obtained by sampling $H(\omega)$ at $\omega = \frac{2\pi k}{N}$. i.e.

$$H(k) = [\,H(\omega)\,]_{\omega=\frac{2\pi k}{N}} = \begin{cases} \sum_{n=0}^{N-1} x(n)[\,cos(\frac{2\pi nk}{N}) + \sin(\frac{2\pi nk}{N})\,] & 0 \le k \le N-1 \\ 0 & otherwise \end{cases} \tag{2.7}$$

The extension of the continuous-time Hartley transform to the DHT was originally proposed by Bracewell [3]. Comparing equation (2.7) with the definition of the DFT given by

$$F\,(\,k\,) = \begin{cases} \sum_{n=0}^{N-1} x(n)[\,cos(\frac{2\pi nk}{N}) - j\,\sin(\frac{2\pi nk}{N})\,] & 0 \le k \le N-1 \\ 0 & otherwise \end{cases}$$

we see that the DHT and the DFT are closely related to each other. In fact, similar to the continuous-time case, the even and odd parts of the DHT correspond to the real and negative of the imaginary parts of the DFT. Moreover, the even and odd parts of $H(k)$ also correspond to the even and odd parts of $x(n)$. That is:

$$F_R(k) = \frac{H(k) + H(N-k)}{2} = DHT\ [\ \frac{x(n) + x(N-n)}{2}\ ] = \sum_{n=0}^{N-1} x(n) \cos(\frac{2\pi nk}{N}) \qquad (2.8a)$$

$$-F_I(k) = \frac{H(k) - H(N-k)}{2} = DHT\ [\ \frac{x(n) - x(N-n)}{2}\ ] = \sum_{n=0}^{N-1} x(n) \sin(\frac{2\pi nk}{N}) \qquad (2.8b)$$

where

$$H(N) \equiv H(0) \qquad (2.8c)$$

$$x(N) \equiv x(0) \qquad (2.8d)$$

and $F_R(k)$ and $F_I(k)$ denote the real and imaginary parts of the Fourier transform. Using equation (2.8) the relationship between the DFT and the DHT can be summarized as:

$$H(k) = F_R(k) - F_I(k) \qquad (2.9a)$$

$$F(k) = \frac{H(k) + H(N-k)}{2} - j\ \frac{H(k) - H(N-k)}{2} \qquad (2.9b)$$

where $H(N)$ is defined in equation (2.8c). One major difference between the DFT and the DHT is the fact that the inverse DHT is identical to the forward DHT. That is, the original sequence can be obtained by taking the DHT of $H(k)$ and scaling it by a factor of $\frac{1}{N}$:

$$x(n) = \begin{cases} \frac{1}{N} \sum_{n=0}^{N-1} H(k) cas(\frac{2\pi nk}{N}) & 0 \le n < N \\ 0 & otherwise \end{cases} \qquad (2.10)$$

The above result can be obtained by using the orthogonality of the *cas*(.) function.

We shall now state some of the notation to be used in the remainder of this thesis †. The notation $x((n))_N$ is used to denote the periodic replication of $x(n)$ with period N. i.e.,

† The notation used in this thesis is basically that described in section 3.6 of Oppenheim and Schafer [1].

$$x((n))_N = \sum_{r=-\infty}^{+\infty} x(n + rN) \qquad (2.11)$$

The original finite duration sequence $x(n)$ is obtained from $x((N))_N$ by extracting one period; i.e.,

$$x(n) = x((n))_N R_N(n) \qquad (2.12)$$

where $R_N(n)$ is used to denote the rectangular sequence given by:

$$R_N(n) = \begin{cases} 1 & 0 \le n < N \\ 0 & \textit{otherwise} \end{cases} \qquad (2.13)$$

2.2. Properties of the Discrete Hartley Transform

For every property of the DFT, there is a corresponding one for the DHT. There are basically two ways of deriving the properties shown in this chapter. The first approach is to use the properties of the DFT and the relationship between the DFT and the DHT. The second approach is to derive the properties directly. Since most of the derivations are straightforward, we will merely state the theorems and the properties. Let $H_1(k), H_2(k), H(k), F_1(k), F_2(k)$ and $F(k)$ denote the discrete Hartley and Fourier transforms of the sequences $x_1(n), x_2(n)$ and $x(n)$ respectively. The following observations can be made:

2.2.1. Linearity Property

This property is shared by both the Hartley and Fourier transforms; if two real sequences $x_1(n)$ and $x_2(n)$ are linearly combined as in:

$$x(n) = ax_1(n) + bx_2(n) \qquad (2.14)$$

Then the DHT of $x(n)$ is

$$H(k) = aH_1(k) + bH_2(k) \tag{2.15}$$

Clearly if $x_1(n)$ has duration N_1 and $x_2(n)$ has duration N_2, then the maximum duration of $x(n)$ will be $N = \max[N_1, N_2]$. Thus in general N-point DHTs must be computed for equation (2.15) to hold.

2.2.2. Circular Shift Property

Using the notation introduced earlier, if we have:

$$x_1(n) = x((n+m))_N R_N(n) \tag{2.16}$$

then

$$H_1(k) = H(k)\cos\frac{2\pi mk}{N} - H((N-k))_N R_N(k)\sin\frac{2\pi mk}{N} \tag{2.17}$$

Because of duality between the time and frequency domains, a similar result holds when a circular shift is applied to the DHT coefficients.

2.2.3. Symmetry Properties

If we define the even part of the real sequence $x(n)$ by:

$$x_{ep}(n) = \frac{[x(n) + x((-n))_N]R_N(n)}{2} \tag{2.18}$$

and its odd part by

$$x_{op}(n) = \frac{[x(n) - x((-n))_N]R_N(n)}{2} \tag{2.19}$$

then DFT of $x_{ep}(n)$ is real and its DHT denoted by $H_{ep}(k)$ is even, i.e,

$$H_{ep}(k) = H_{ep}((N-k))_N R_N(k) \tag{2.20}$$

Also DFT of $x_{op}(n)$ is purely imaginary and its DHT denoted by $H_{op}(k)$ is odd:

$$H_{op}(k) = -H_{op}((N-k))_N R_N(k) \tag{2.21}$$

This property was also mentioned in the first part of this chapter. It states that the DHT of an even sequence is even and the DHT of an odd sequence is odd.

2.2.4. Convolution Property

Let $x_1(n)$ and $x_2(n)$ be N_1 and N_2-point sequences respectively and their convolution be an $(N_1 + N_2 - 1)$-point sequence given by

$$x(n) = x_1(n) * x_2(n) \tag{2.22}$$

The DHT can be used to perform linear convolution. More specifically, the $(N_1 + N_2 - 1)$-point discrete Hartley transform of the sequences $x(n), x_1(n), x_2(n)$ are related as follows:

$$H(k) = H_1(k)H_{2e}(k) + H_1((N-k))_N R_N(k)\, H_{2o}(k) \tag{2.23}$$

where

$$H_{2e}(k) \doteq \frac{H_2(k) + H_2((N-k))_N R_N(k)}{2} \tag{2.24}$$

$$H_{2o}(k) = \frac{H_2(k) - H_2((N-k))_N R_N(k)}{2} \tag{2.25}$$

Note that the reason behind choosing the size of the DHTs to be $(N_1 + N_2 - 1)$ is completely analogous to the DFT case described in [1]. The convolution property is by far one of the most important properties of the DHT. In many applications such as linear filtering, one can bypass the Fourier domain altogether and perform the convolution in the Hartley domain. This is particularly attractive in applications such as image processing where the impulse response of the filters used are usually symmetric. In this case H_{2o} becomes zero and equation (2.23) becomes

$$H(k) = H_1(k)\, H_2(k) \tag{2.26}$$

which is similar to what we would have obtained had we used the DFT to perform convolution.

2.2.5. Reciprocal Property

To perform inverse DFT using a forward DFT algorithm, we would have to rearrange the sequence. This is not necessary for the discrete Hartley transform since it is its own inverse. This is shown in equations (2.7) and (2.10).

2.2.6. Reversal Property

Let

$$x_1(n) = x((N-n))_N R_N(n) \tag{2.27}$$

Then

$$H_1(k) = H((N-k))_N R_N(k) \tag{2.28}$$

Note that the symmetry properties can be derived using the reversal property in a straightforward manner.

2.2.7. Product theorem

If

$$x(n) = x_1(n)\, x_2(n) \tag{2.29}$$

Then

$$H(k) = \frac{1}{N}\sum_{l=0}^{N-1} H_1(l)\, H_{2e}((N-k+l))_N R_N(l) + H_1((N-l))_N\, H_{2o}((k-l))_N R_N(l) \tag{2.30}$$

where $H_{2e}(k)$ and $H_{2o}(k)$ are defined in equations (2.24) and (2.25) respectively. The product theorem stated above is the dual of the convolution theorem described

earlier.

2.2.8. Parseval's Theorem

For an N-point real sequence x(n) we have

$$\sum_{n=0}^{N-1} x^2(n) = N \sum_{k=0}^{N-1} H^2(k) \tag{2.31}$$

The cross correlation, autocorrelation, initial value, sum of sequence, similarity and packing theorems are identical for the DFT and DHT [3],[17].

For comparison purposes the DHT and DFT properties are shown in table 2.1.

Sequence	DHT	DFT
$x(n)$	$H(k)$	$F(k)$
$x_1(n)$	$H_1(k)$	$F_1(k)$
$x_2(n)$	$H_2(k)$	$F_2(k)$
$ax_1(n)+bx_2(n)$	$aH_1(k)+bH_2(k)$	$aF_1(k)+bF_2(k)$
$x((n+m))_N R_N(n)$	$H(k)\cos(\frac{2\pi mk}{N})$ $-H((N-k))_N R_N(k)\sin(\frac{2\pi mk}{N})$	$e^{j\frac{2\pi km}{N}}F(k)$
$x_1(n)*x_2(n)$	$H_1(k)H_{2e}(k)+H_1((N-k))R_N(k)_N H_{2o}(k)$	$F_1(k)F_2(k)$
$x_1(n)x_2(n)$	$\frac{1}{N}\sum_{l=0}^{N-1}[H_1(l)H_{2e}((N-k+l))R_N(k)_N +$ $H_1((N-l))_N R_N(l)H_{2o}((k-l))_N R_N(l)]$	$\frac{1}{N}\sum_{l=0}^{N-1}F_1(l)F_2((k-l))_N R_N(l)$
$x_{ep}(n)$	$H_{ep}(k)=H_{ep}((N-k))_N R_N(k)$	$Re[F(k)]$
$x_{op}(n)$	$H_{op}(k)=-H_{op}((N-k))_N R_N(k)$	$jIm[F(k)]$
$x((N-n))_N R_N(n)$	$H((N-k))_N R_N(k)$	$F^*((N-k))_N R_N(k)$
$\sum_{n=0}^{N-1}x^2(n)$	$N\sum_{k=0}^{N-1}H^2(k)$	$N\sum_{k=0}^{N-1}\lvert F(k)\rvert^2$

Table 2.1 Properties of the DHT and the DFT

CHAPTER 3: Bracewell's Discrete Hartley Transform Algorithm

As explained in chapter 2, the discrete Hartley transform can potentially be used in the implementation of many digital signal processing algorithms and systems. The DHT has fast algorithms similar in style to the FFT, the first of these proposed by Bracewell [4]. The fundamental principle that all these algorithms are based upon is that of decomposing the computation of the discrete Hartley transform of a sequence of length N into successively smaller discrete Hartley transforms. The manner in which this principle is implemented leads to a variety of algorithms with different computational efficiencies and error properties.

In this chapter, we shall begin by describing the original decimation-in-time radix 2 algorithm proposed by Bracewell [4]. As is the case with the FFT, the idea in Bracewell algorithm can be extended to radix 4, split radix 4 and decimation-in-frequency algorithms [6]. Sections 3.1.2, 3.1.3 and 3.2 will review these algorithms. Wang's algorithm [5], and a new algorithm for computing the DHT will be covered in chapters 4 and 5 respectively. Other algorithms such as prime factor algorithm and Winograd-type Hartley transform algorithm are discussed at length in [6].

3.1. Decimation-in-Time Algorithms

3.1.1. Bracewell's Original Algorithm

Bracewell has developed a decimation-in-time radix 2 algorithm for performing the discrete Hartley transform of a data sequence of N real elements in a time proportional to $N\log_2 N$ [4]. In the remainder of this thesis we shall refer to this

algorithm as DT1 where DT stands for decimation-in-time. In this section, we will derive Bracewell's decomposition in two different ways and propose a minor modification to the DT1 algorithm.

The simplest way of deriving the DT1 algorithm which is very similar to the FFT computes $H(k)$ by separating $x(n)$ into two $\frac{N}{2}$-point sequences consisting of even and odd points in $x(n)$. Thus we obtain,

$$H(k) = H_1(k) + H_2(k) \tag{3.1}$$

where

$$H_1(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n)cas(\frac{2\pi nk}{N/2}) \tag{3.2a}$$

$$H_2(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)cas(\frac{2\pi(2n+1)k}{N}) \tag{3.2b}$$

$H_1(k)$ can be identified as an $\frac{N}{2}$-point DHT. Using the identity

$$cas(\alpha + \beta) = \cos(\beta)\, cas(\alpha) + \sin(\beta)\, cas(-\alpha) \tag{3.3}$$

and letting $\alpha = \frac{2\pi nk}{N/2}$ and $\beta = \frac{2\pi k}{N}$, $H_2(k)$ of equation (3.2b) can be written as

$$H_2(k) = cos(\frac{2\pi k}{N})H_3(k) + \sin(\frac{2\pi k}{N})H_3((\frac{N}{2}-k))_{\frac{N}{2}}R_{N/2}(k) \tag{3.4}$$

where

$$H_3(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)cas(\frac{2\pi nk}{N/2}) \tag{3.5}$$

is an $\frac{N}{2}$-point DHT. Thus we have managed to show that an N-point DHT can be

obtained by computing two $\frac{N}{2}$-point DHTs. By repeating the above procedure the DHT can be decomposed further.

Another way of looking at Bracewell algorithm is through the concept of index mapping [6], [9]. Index mapping has been used to derive various versions of the FFT in a systematic fashion [9]. It involves mapping a one-dimensional array of size $N = N_1N_2$ onto a two dimensional array of size N_1 by N_2. The mapping is done through the substitution

$$n = K_1n_1 + K_2n_2 \quad (mod\ N) \tag{3.6a}$$

$$k = K_3k_1 + K_4k_2 \quad (mod\ N) \tag{3.6b}$$

in equation (2.7). This will result in a complicated expression which will not be reproduced here. When index mapping is used with DFTs, suitable choices of the constants K_1 through K_4 make it possible to save operations by breaking the DFT into smaller DFTs. In the case of the DHT, the particular map

$$n = n_1 + Ln_2 \tag{3.7a}$$

$$k = k_1 + Kk_2 \tag{3.7b}$$

$$K\ L = N \tag{3.7c}$$

proposed by Burrus [6], will be examined. Substituting the above equation in (2.7) and using the two identities

$$cas(\alpha + \beta) = cas(\alpha)\cos(\beta) + cas(-\alpha)\sin(\beta) \tag{3.8a}$$

$$cas(-\alpha) = cas(\alpha) - 2\sin(\alpha) \tag{3.8b}$$

we obtain the following equation:

$$H(k_1 + Kk_2) = \sum_{n_1=0}^{L-1}\sum_{n_2=0}^{K-1} x(n_1 + Ln_2)\,[cas(\frac{2\pi n_1 k_1}{N})cas(\frac{2\pi n_2 k_1}{K})cas(\frac{2\pi n_1 k_2}{L}) \qquad (3.9)$$

$$-2\sin(\frac{2\pi n_2 k_1}{K})\cos(\frac{2\pi n_1 k_2}{L})\sin(\frac{2\pi n_1 k_1}{N})$$

$$-2\sin(\frac{2\pi n_2 k_1}{K})\sin(\frac{2\pi n_1 k_2}{L})\cos(\frac{2\pi n_1 k_1}{N})$$

$$-2\sin(\frac{2\pi n_2 k_1}{K})\sin(\frac{2\pi n_1 k_2}{L})\sin(\frac{2\pi n_1 k_1}{N})$$

$$-2\cos(\frac{2\pi n_2 k_1}{K})\sin(\frac{2\pi n_1 k_2}{L})\sin(\frac{2\pi n_1 k_1}{N})\,]$$

Choosing $L=2$ and $K=\frac{N}{2}$, the last three terms of the above expression become zero. Using the identity

$$cas(\alpha)\,cas(\beta) - 2\sin(\alpha)\sin(\beta) = cas(\alpha + \beta) \qquad (3.10)$$

we get

$$H(k_1 + \frac{N}{2}k_2) = \sum_{n_1=0}^{1}\sum_{n_2=0}^{\frac{N}{2}-1} (-1)^{n_1 k_2}\, x(n_1 + 2n_2)\, cas[\,\frac{2\pi k_1 (n_1+2n_2)}{N}\,] \qquad (3.11)$$

Since we have chosen $K = \frac{N}{2}$ in equation (3.7b), to generate N frequency points k, we are only concerned about $k_2 = 0, 1$ and $k_1 = 0, \ldots, \frac{N}{2}-1$. For $k_2 = 0$, equation (3.11) becomes

$$H(k) = H(k_1) = \sum_{n_2=0}^{\frac{N}{2}-1} x(2n_2)cas(\frac{2\pi k_1(2n_2)}{N}) \qquad (3.12a)$$

$$+ \sum_{n_2=0}^{\frac{N}{2}-1} x(2n_2+1)cas(\frac{2\pi k_1(2n_2+1)}{N}) \qquad 0 \le k_1 < \frac{N}{2}$$

On the other hand, when $k_2 = 1$ we get

$$H(k) = H(k_1+\frac{N}{2}) = \sum_{n_2=0}^{\frac{N}{2}-1} x(2n_2)cas(\frac{2\pi k_1(2n_2)}{N}) \qquad (3.12b)$$

$$- \sum_{n_2=0}^{\frac{N}{2}-1} x(2n_2+1)cas(\frac{2\pi k_1(2n_2+1)}{N}) \qquad 0 \le k_1 < \frac{N}{2}$$

Thus equation (3.12) is essentially the same decomposition shown in equations (3.1) and (3.2) and (3.9) reduces to (3.1) and (3.2) for $L = 2$ and $k = \frac{N}{2}$.

Conceptually speaking, Bracewell algorithm consists of two parts: In the first part the input sequence is rearranged in a bit reversed manner and in the second part the subsequences are combined in a butterfly-type structure. The butterfly for the last stage of an N-point DHT is shown in figure 3.1 Although the algorithm can be implemented in place, unlike the butterflies in the FFT , the radix two version of Bracewell algorithm requires butterflies with four inputs and outputs. In other words, four elements should be included in each butterfly in order to assure that no element which will be needed later is overwritten. The flow graph of the decimation-in-time version of Bracewell algorithm for the case $N = 16$ is shown in figure 3.2. Note that C_M^r and S_M^r in figure 3.2 and all the flow graphs in this thesis denote the quantities $\cos(\frac{2\pi r}{M})$ and $\sin(\frac{2\pi r}{M})$ respectively. The number of real multiplies for an N-point sequence using this algorithm is on the order of $N\log_2 N$ and the number of real adds is proportional to $\frac{3N}{2}\log_2 N$. This is the same as a real valued FFT.

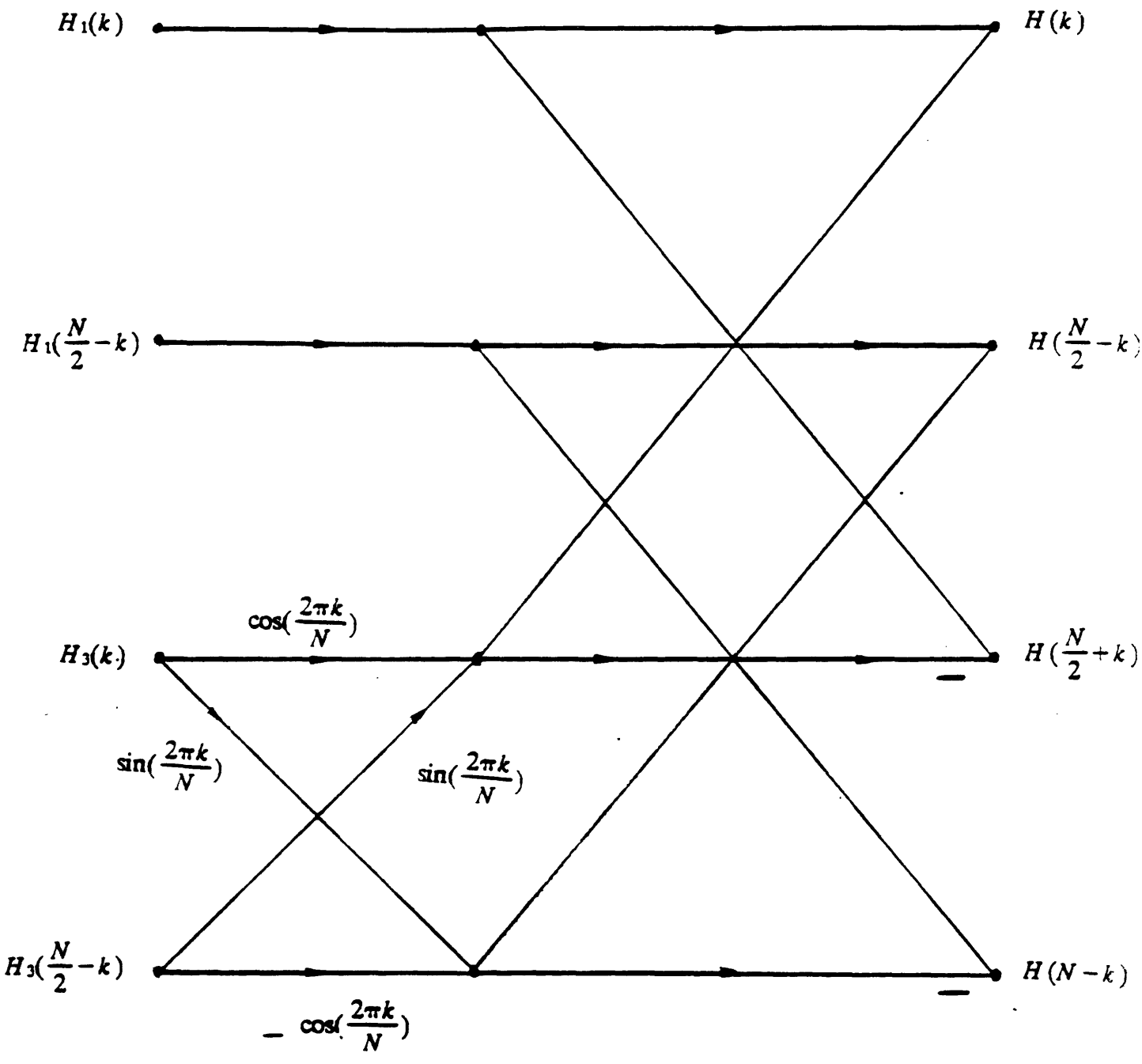


Fig. 3.1 Flow graph of the *kth* butterfly of the last stage of an N-point DHT computation using the DTI algorithm

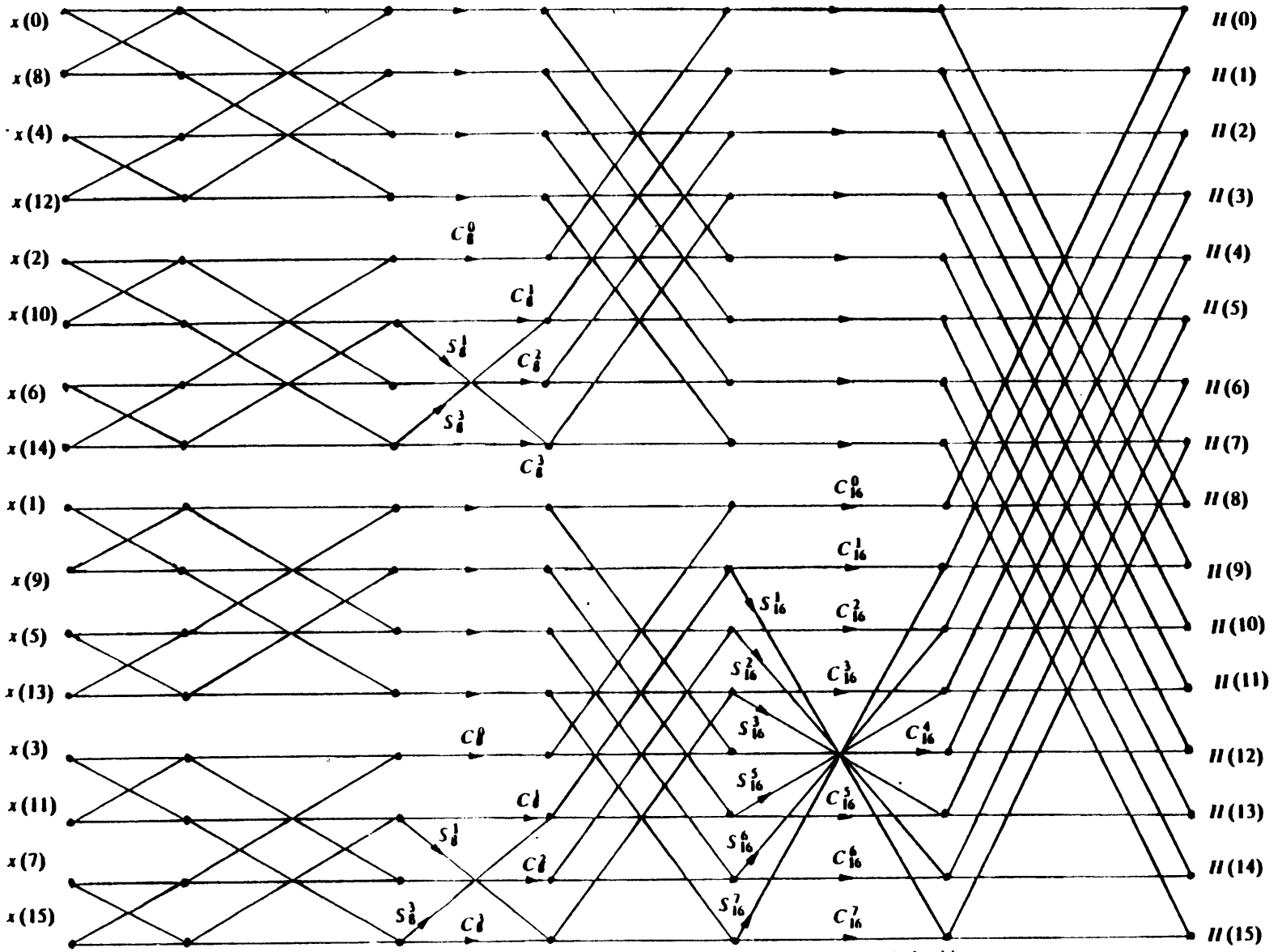


Fig. 3.2 **Flow graph of the decimation-in-time decomposition of an 16-point DHT computation using the DIT algorithm**

The operation count can be further reduced if the following observation is made while computing the butterfly shown in figure 3.1: It is obvious that computing any four points $H(k)$, $H(\frac{N}{2}-k)$, $H(\frac{N}{2}+k)$ and $H(N-k)$ in figure 3.1 requires the computation of the two intermediate quantities:

$$Y_1(k) = H_3(k)cos(\frac{2\pi k}{N}) + H_3((\frac{N}{2}-k))_{\frac{N}{2}} R_{\frac{N}{2}}(k)\sin(\frac{2\pi k}{N}) \qquad (3.13a)$$

$$Y_2(k) = H_3(k)sin(\frac{2\pi k}{N}) - H_3((\frac{N}{2}-k))_{\frac{N}{2}} R_{\frac{N}{2}}(k)\cos(\frac{2\pi k}{N}) \qquad (3.13b)$$

More specifically, using figure 3.1 we have

$$H(k) = H_1(k) + Y_1(k) \qquad (3.14a)$$

$$H(\frac{N}{2}+k) = H_1(k) - Y_1(k) \qquad (3.14b)$$

$$H(\frac{N}{2}-k) = H_1(k) + Y_2(k) \qquad (3.14c)$$

$$H(N-k) = H_1(k) - Y_2(k) \qquad (3.14d)$$

Instead of using 4 multiplies and 2 adds, $Y_1(k)$ and $Y_2(k)$ can be computed with three of each in the following manner:

$$Y_1(k) = [sin(\frac{2\pi k}{N}) + cos(\frac{2\pi k}{N})]H_3(k) + sin(\frac{2\pi k}{N})[\, H_3((\frac{N}{2}-k))_{\frac{N}{2}} R_{\frac{N}{2}}(k) - H_3(k)\,] \qquad (3.15a)$$

$$Y_2(k) = [sin(\frac{2\pi k}{N}) - cos(\frac{2\pi k}{N})]H_3(k) - \sin(\frac{2\pi k}{N})[\, H_3((\frac{N}{2}-k))_{\frac{N}{2}} R_{\frac{N}{2}}(k) - H_3(k)] \qquad (3.15b)$$

The above implementation will be referred to as the MDT1 algorithm where MDT stands for modified decimation-in-time. The MDT1 algorithm requires on the order of $\frac{3N}{4}\log_2 N$ multiplies and $\frac{3N}{2}\log_2 N$ adds. Note that the total operation count for the original and the modified version of Bracewell algorithm are the same. However, the error properties of the MDT1 are different from the original one

proposed by Bracewell. This will be discussed in more detail in future chapters.

3.1.2. Radix 4 Decimation-in-Time (R4DT1) Algorithm

As is the case with the FFT, the idea behind the original radix 2 Bracewell algorithm can be extended to other radices such as radix 4 or the recently proposed split radix algorithms [6]-[8]. In this section we shall describe the radix 4 decimation-in-time algorithm which will be referred to as the R4DT1 algorithm. The next section will deal with the split radix algorithm.

The R4DT1 algorithm is obtained by decomposing an N-point DHT into four $\frac{N}{4}$-point DHTs. Thus equation (2.7) can be written as:

$$H(k) = H_0(k) + H_1(k) + H_2(k) + H_3(k) \tag{3.16}$$

where

$$H_i(k) = \sum_{n=0}^{\frac{N}{4}-1} x(4n+i)cas(\frac{2\pi(4n+i)k}{N}) \tag{3.17}$$

Letting $\alpha = \frac{2\pi nk}{N/4}$ and $\beta = \frac{2\pi ik}{N}$ in the identity (3.3) equation (3.17) can be written as

$$H_i(k) = \cos(\frac{2\pi ik}{N})H'_i((k))_{N/4}R_{N/4}(k) + \sin(\frac{2\pi ik}{N})H'_i((\frac{N}{4}-k))_{N/4}R_{N/4}(k) \tag{3.18}$$

where

$$H'_i(k) = \sum_{n=0}^{\frac{N}{4}-1} x(4n+i)cas(\frac{2\pi nk}{N/4}) \tag{3.19}$$

Another way of arriving at this result would be to choose $L=4$ and $K=\frac{N}{4}$ in the index mapping equation (3.7) [6]. The *kth* butterfly of the last stage of an N-point

DHT using the R4DT1 algorithm is shown in figure 3.3. Figure 3.4 depicts the signal flow graph of the R4DT1 algorithm for an 16-point transform.

The total multiplication count for this algorithm is on the order of $\frac{3N}{2}\log_4 N$ and the total number of additions are on the order of $\frac{11N}{4}\log_4 N$. The operation count therefore is the same as a real valued radix 4 FFT. Again, the observation made in equation (3.15) can be used to replace 4 multiplies and 2 adds with 3 of each.

Although the operation count for the R4DT1 algorithm is less than that of the DT1 algorithm, the former is more complex to implement. Comparison of figures 3.2 and 3.4 are indicative of this fact.

3.1.3. Split Radix Decimation-in-Time (SRDT1) Algorithm

Recently, Duhamel and Hollmann derived an algorithm called the split radix FFT (SRFFT) [7]. Burrus developed an indexing scheme which efficiently implements the Duhamel-Hollmann SRFFT [8]. A similar approach can be taken to derive split radix DHT algorithms. In particular, the idea behind Bracewell's original radix 2 algorithm (DT1) can be extended to a split radix algorithm [6]. We will refer to this algorithm as the SRDT1 algorithm. The SRDT1 algorithm applies a radix 2 decomposition to the even indexed samples and a radix 4 decomposition to the odd indexed samples. Thus using the notation of equation (3.17) we get:

$$H(k) = [\, H_0(k) + H_2(k) \,] + H_1(k) + H_3(k) \qquad (3.20)$$

The first term in the above equation corresponds to the $\frac{N}{2}$-point DHT of the even

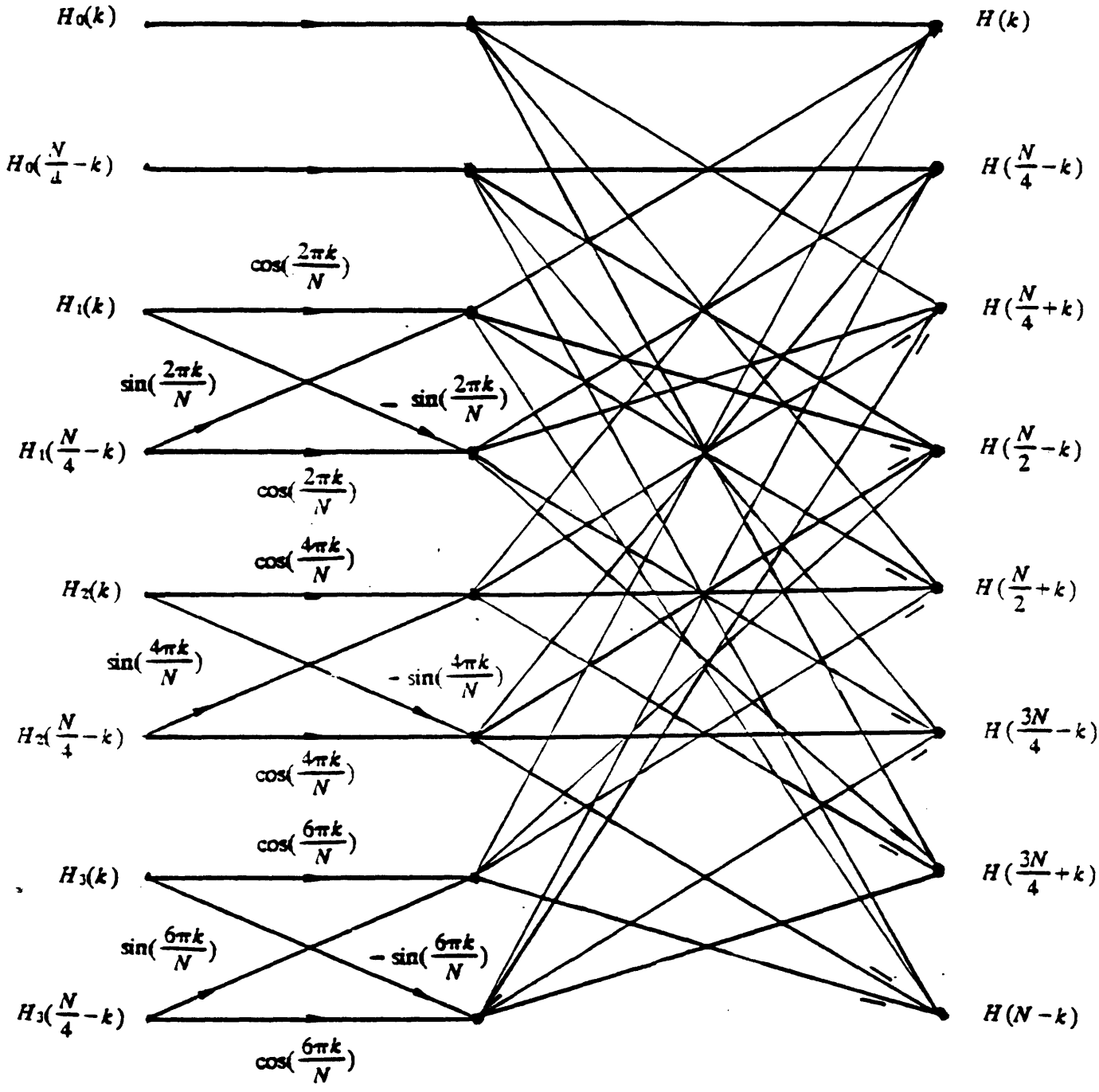


Fig. 3.3 Flow graph of the *kth* butterfly of the last stage of an N-point DHT computation using the R4DT1 algorithm

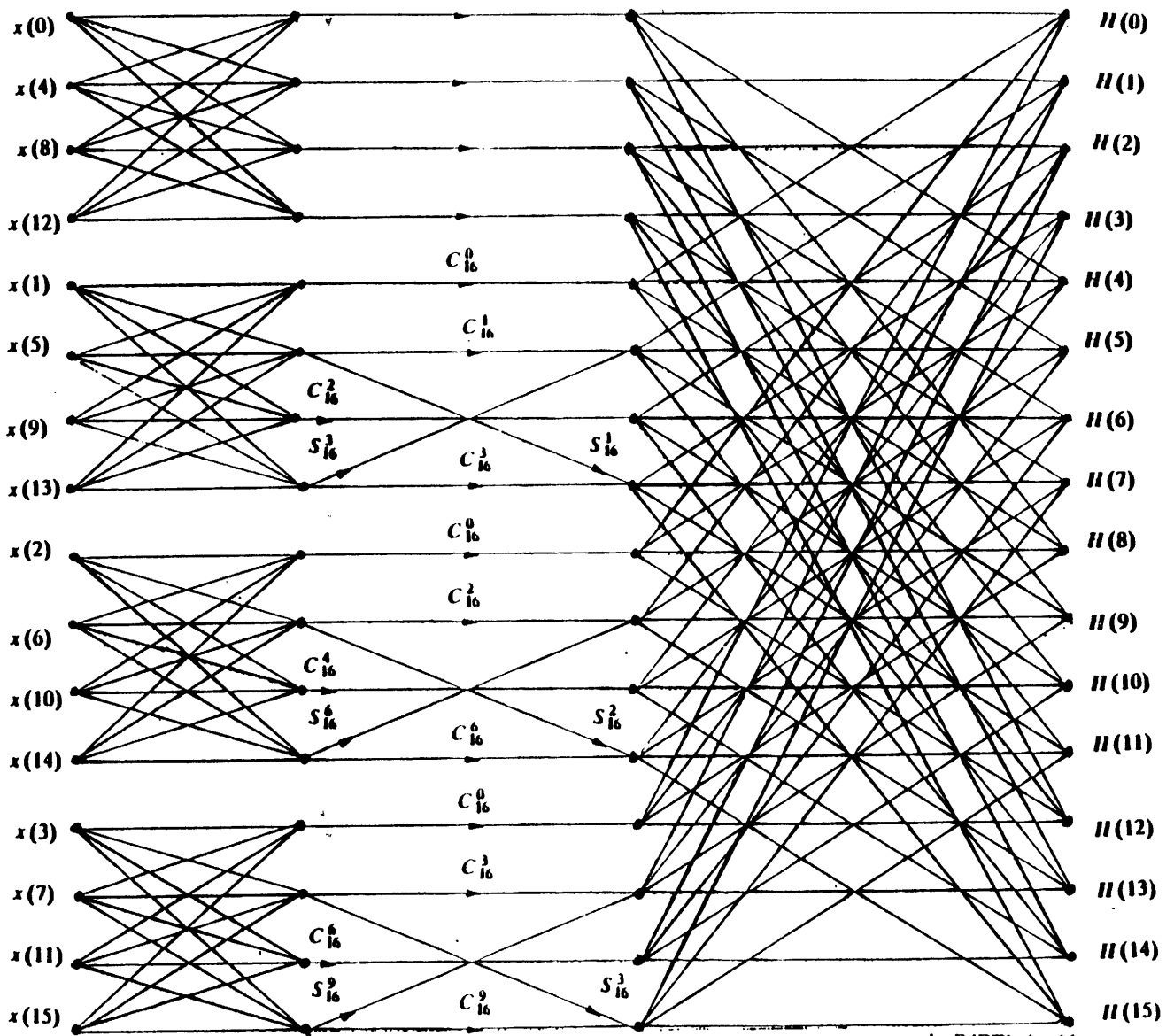


Fig. 3.4 Flow graph of the decimation-in-time decomposition of an 16-point DHT computation using the R4DT1 algorithm

points of the original sequence $x(n)$. That is:

$$H_0(k) + H_2(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n)cas(\frac{2\pi nk}{N/2}) \tag{3.21}$$

The second and third term correspond to two $\frac{N}{4}$-point DHTs. They can be written as

$$H_i(k) = \cos(\frac{2\pi ik}{N})H'_i(k) + \sin(\frac{2\pi ik}{N})H'_i((\frac{N}{4}-k))_{\frac{N}{4}}R_{\frac{N}{4}}(k) \qquad i = 1, 3 \tag{3.22}$$

where

$$H'_i(k) = \sum_{n=0}^{\frac{N}{4}-1} x(4n+i)cas(\frac{2\pi nk}{N/4}) \tag{3.23}$$

Thus we have shown that an N-point DHT can be computed via an $\frac{N}{2}$-point DHT and two $\frac{N}{4}$-point DHTs. By repeating the above procedure the DHT can be decomposed further. This result can also be obtained using the index mapping technique described earlier [6].

The multiplication count for this algorithm is on the order of $\frac{2N}{3}\log_2 N$ and the number of additions is on the order of $\frac{4N}{3}\log_2 N$. These are the same as the operation count for a similar real-valued split radix FFT.

Although the operation count for the split radix algorithm is less than that of radix 2 or radix 4 algorithms, the SRDT1 algorithm does not progress stage by stage or in terms of indices does not complete each nested sum in order. This makes the indexing scheme more complex than that of the fixed radix algorithms.

3.2. Radix 2 Decimation-in-Frequency (DF1) Algorithm

As is the case with the FFT, the idea behind the DT1 algorithm can be extended to the decimation-in-frequency algorithm which we will refer to as the DF1 algorithm.

The DF1 algorithm can be derived using two different approaches. In the first approach, which is very similar to the FFT, we divide the input sequence into the first half and the last half of the points so that the transform $H(k)$ can be written as:

$$H(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n)cas(\frac{2\pi nk}{N}) + \sum_{n=0}^{\frac{N}{2}-1} x(n+\frac{N}{2})(-1)^k cas(\frac{2\pi nk}{N}) \quad (3.24)$$

Consider k even and k odd separately, with $H(2r)$ and $H(2r+1)$ representing the even numbered points and the odd numbered points respectively, so that

$$H(2r) = \sum_{n=0}^{\frac{N}{2}-1} [\, x(n) + x(n+\frac{N}{2}) \,] \, cas(\frac{2\pi n\ r}{N/2}) \quad (3.25a)$$

$$H(2r+1) = \sum_{n=0}^{\frac{N}{2}-1} [\, x(n) - x(n+\frac{N}{2}) \,] \, cas(\frac{2\pi n(\,2r+1\,)}{N}) \quad (3.25b)$$

Equation (3.25a) is an $\frac{N}{2}$-point DHT. Letting $\alpha = \frac{2\pi rn}{N/2}$ and $\beta = \frac{2\pi n}{N}$ in identity (3.3) equation (3.25b) can be written as

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

$$+ [\, x(\frac{N}{2}-n) - x((N-n))_N R_N(n)]\sin(\frac{2\pi n}{N}) \Big\} cas(\frac{2\pi nr}{N/2}) \quad (3.26)$$

which is another $\frac{N}{2}$-point DHT. Thus once again an N-point DHT has been

decomposed into two $\frac{N}{2}$-point DHTs. By repeating the above procedure the DHT can be decomposed further.

Another way of deriving the DF1 algorithm is to substitute the index map

$$n = n_1 + \frac{N}{2} n_2 \tag{3.27a}$$

$$k = k_1 + 2 k_2 \tag{3.27b}$$

in equation (2.7) [6].

The *kth* butterfly of the first stage of the DF1 algorithm is shown in figure 3.5 and the signal flow graph of an 16-point DHT using the DF1 algorithm is shown in figure 3.6. Figure 3.2 is the transpose of the flow graph in figure 3.6 and can be obtained by reversing the direction of the signal flow and interchanging the input and the output in figure 3.6. By transposition theorem, the input output characteristics of the two flow graphs are the same.

The operation count for the DF1 algorithm is identical to that of the DT1 algorithm and can be modified by applying equation (3.15) in computing the butterflies.

Radix 4 and split radix decimation-in-frequency versions of Bracewell's original algorithm are very similar to the corresponding decimation-in-time algorithms and can be derived in a similar manner.

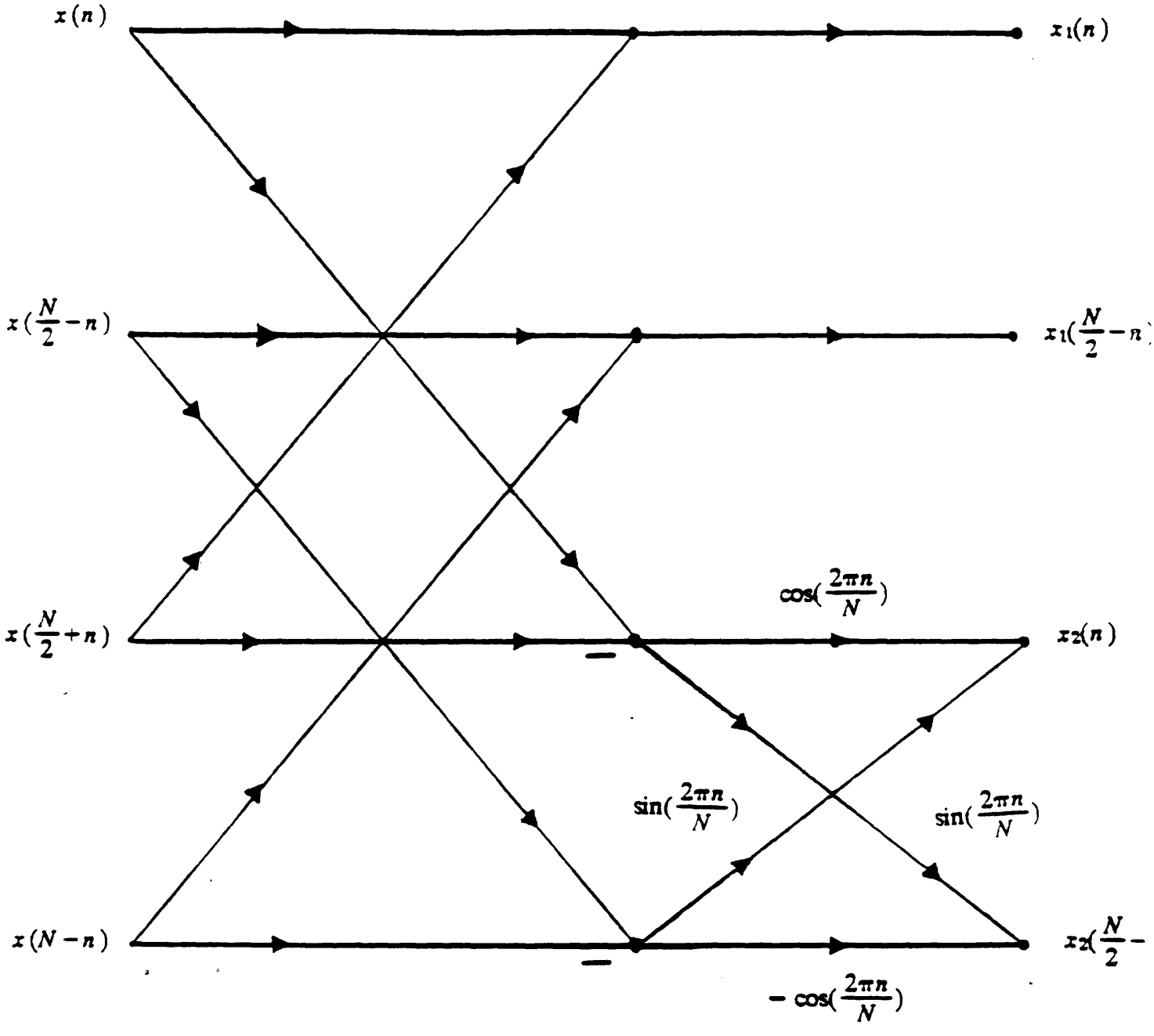


Fig. 3.5 Flow graph of the *kth* butterfly of the first stage of an N-point DHT computation

using the DF1 algorithm

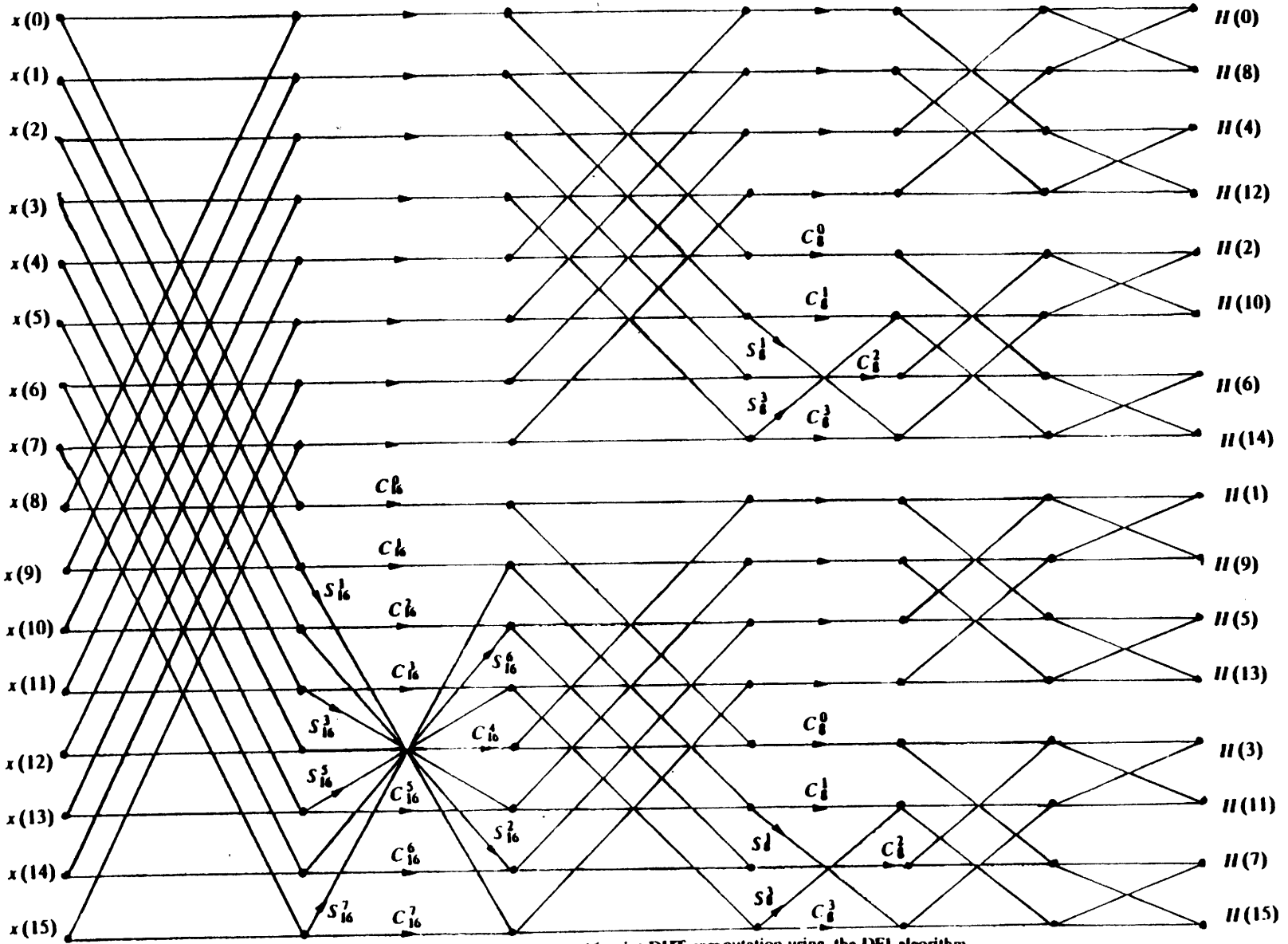


Fig 3.6 Flow graph of the decimation-in-frequency decomposition of an 16-point DHT computation using the DF1 algorithm

CHAPTER 4: Wang's Algorithm for the Discrete Hartley Transform

4.1. The Algorithm

Wang has recently proposed a new algorithm for computing the DHT [5]. This algorithm is based on a systematic factorization of the DHT matrix. An attempt is made here to explain the intuitive reasoning behind different stages of the matrix decompositions used to derive the algorithm.

Throughout this section we assume N is a power of 2 unless otherwise stated. Let $\left[\ .\ \right]$ denote a square matrix for various discrete transforms with its dimension represented by a subscript inside the pair of square brackets and its version number (as defined in [5] for the discrete cosine or sine transform) represented by a superscript. For instance $\left[\ C_{N+1}^{1}\ \right]$ and $\left[\ S_{N-1}^{1}\ \right]$ stand for $(N+1)$-point discrete cosine transform matrix and $(N-1)$-point discrete sine transform matrix of the first kind respectively. These transforms will be referred to DCT1 and DST1 and are defined as:

$$C^{1}(k) = \sum_{n=0}^{N} x(n)\cos\left(\frac{\pi nk}{N}\right) \tag{4.1a}$$

$$S^{1}(k) = \sum_{n=1}^{N-1} x(n)\sin\left(\frac{\pi nk}{N}\right) \tag{4.1b}$$

The first step in Wang's algorithm is to divide the problem of finding an N-point DHT into an $(\frac{N}{2}+1)$-point DCT1 and an $(\frac{N}{2}-1)$-point DST1. This can be done by separating the sine and cosine terms in the DHT expression of equation

(2.7). Thus we get:

$$H(k) = \sum_{n=0}^{N-1} x(n)\cos(\frac{2\pi nk}{N}) + \sum_{n=0}^{N-1} x(n)\sin(\frac{2\pi nk}{N}) \tag{4.2}$$

Exploiting the fact that cosine is an even function and sine is an odd function, equation (4.2) can be written as:

$$H(k) = C^{\frac{1}{1}}_{\frac{N}{2}+1}(k) + S^{\frac{1}{1}}_{\frac{N}{2}-1}(k) \tag{4.3}$$

where

$$C^{\frac{1}{1}}_{\frac{N}{2}+1}(k) = \sum_{n=0}^{\frac{N}{2}} [x(n) + x((N-n))_N R_N(n)]\cos(\frac{\pi nk}{N/2}) \tag{4.4a}$$

$$S^{\frac{1}{1}}_{\frac{N}{2}-1}(k) = \sum_{n=1}^{\frac{N}{2}-1} [x(n) - x((N-n))_N R_N(n)]\sin(\frac{\pi nk}{N/2}) \tag{4.4b}$$

Equations (4.4a) and (4.4b) can be identified as $(\frac{N}{2}+1)$-point DCT1 and $(\frac{N}{2}-1)$-point DST1 of the sequences

$$x_1(n) = x(n) + x((N-n))_N R_N(n) \tag{4.5a}$$

$$x_2(n) = x(n) - x((N-n))_N R_N(n) \tag{4.5b}$$

respectively.

In terms of matrices, the decomposition shown in equations (4.2) through (4.4) can be expressed as [5]:

$$\frac{1}{\sqrt{2}}\left[H_N\right] = \left[A_N^{\frac{1}{1}}\right]\begin{bmatrix} C^{\frac{1}{1}}_{\frac{N}{2}+1} & 0 \\ 0 & \bar{I}\, S^{\frac{1}{1}}_{\frac{N}{2}-1}\, \bar{I} \end{bmatrix}\left[A_N^{\frac{1}{1}}\right] \tag{4.6}$$

where

$$\left[A_N^1\right] = \frac{1}{\sqrt{2}} \begin{bmatrix} \sqrt{2} & 0 & 0 & 0 \\ 0 & I_{\frac{N}{2}-1} & 0 & \bar{I}_{\frac{N}{2}-1} \\ 0 & 0 & \sqrt{2} & 0 \\ 0 & \bar{I}_{\frac{N}{2}-1} & 0 & -I_{\frac{N}{2}-1} \end{bmatrix} \tag{4.7}$$

$$\bar{I} = \begin{bmatrix} & & & 1 \\ & & \cdot & \\ & \cdot & & \\ 1 & & & \end{bmatrix} \tag{4.8}$$

and $\left[H_N\right]$, $\left[C_{\frac{N}{2}+1}^{\frac{1}{2}}\right]$ and $\left[S_{\frac{N}{2}-1}^{\frac{1}{2}}\right]$ denote the DHT, DCT1 and DST1 matrices respectively. Thus equation (4.6) is another way of looking at the decomposition of an N-point DHT into an ($\frac{N}{2}$ + 1)-point DCT1 and ($\frac{N}{2}$ − 1)-point DST1 g.

The second step in the algorithm is to consider the even and odd frequency points of equation (4.4a) separately with $C_{\frac{N}{2}+1}^{\frac{1}{2}}(2r)$ and $C_{\frac{N}{2}+1}^{\frac{1}{2}}(2r+1)$ representing the even numbered and the odd numbered points respectively so that

$$C_{\frac{N}{2}+1}^{\frac{1}{2}}(2r) = \sum_{n=0}^{\frac{N}{4}} [x_1(n) + x_1((\frac{N}{2}-n))_N R_N(n)] cos(\frac{\pi nr}{N/4}) \tag{4.9a}$$

$$C_{\frac{N}{2}+1}^{\frac{1}{2}}(2r+1) = \sum_{n=0}^{\frac{N}{4}-1} [x_1(n) - x_1((\frac{N}{2}-n))_N R_N(n)] cos(\frac{\pi n(r+\frac{1}{2})}{N/4}) \tag{4.9b}$$

Equations (4.9a) and (4.9b) denote ($\frac{N}{4}$+1)-point DCT1 and $\frac{N}{4}$-point DCT3 of the sequences

† Note that our definition of sine and cosine transforms are within $\sqrt{\frac{2}{N}}$ of the ones used in [5]. Similarly, our definition of the DHT is within $\frac{1}{\sqrt{N}}$ of the definition of the discrete Wang transform described in [5].

$$x_3(n) = x_1(n) + x_1((\frac{N}{2}-n))_N R_N(n) \tag{4.10a}$$

$$x_4(n) = x_1(n) - x_1((\frac{N}{2}-n))_N R_N(n) \tag{4.10b}$$

respectively where DCT3 and DST3 of an N-point sequence $x(n)$ are defined as:

$$C_N^3(k) = \sum_{n=0}^{N-1} x(n)\cos(\frac{\pi n(k+\frac{1}{2})}{N}) \tag{4.11a}$$

$$S_N^3(k) = \sum_{n=1}^{N} x(n)\sin(\frac{\pi n(k-\frac{1}{2})}{N}) \tag{4.11b}$$

Therefore equation (4.9) can be written as :

$$C_{\frac{N}{2}+1}^{\frac{1}{N}}(2r) = C_{\frac{N}{4}+1}^{\frac{1}{N}}(r) = \sum_{n=0}^{\frac{N}{4}} x_3(n)\cos(\frac{\pi nr}{N/4}) \tag{4.12a}$$

$$C_{\frac{N}{2}+1}^{\frac{1}{N}}(2r+1) = C_{\frac{N}{4}}^{\frac{3}{N}}(r) = \sum_{n=0}^{\frac{N}{4}-1} x_4(n)\cos(\frac{\pi n(r+\frac{1}{2})}{N/4}) \tag{4.12b}$$

Thus we have shown that the $(\frac{N}{2}+1)$-point DCT1 of equation (4.4a) can be decomposed into an $(\frac{N}{4}+1)$-point DCT1 and an $\frac{N}{4}$-point DCT3. A similar approach can be taken to show that the $(\frac{N}{2}-1)$ DST1 of equation (4.4b) can be decomposed into an $(\frac{N}{4}-1)$-point DST1 and an $\frac{N}{4}$-point DST3.

In terms of matrices, the second step of the algorithm involves decomposing $[C_{\frac{N}{2}+1}^{\frac{1}{N}}]$ and $[S_{\frac{N}{2}-1}^{\frac{1}{N}}]$ of equation (4.6) of the first step into the smaller matrices $[C_{\frac{N}{4}+1}^{\frac{1}{N}}]$, $[C_{\frac{N}{4}}^{\frac{3}{N}}]$ $[S_{\frac{N}{4}}^{\frac{3}{N}}]$ and $[S_{\frac{N}{4}-1}^{\frac{1}{N}}]$ in the following manner:

$$\left[C^1_{\frac{N}{2}+1}\right] = \left[P_{\frac{N}{2}+1}\right] \begin{bmatrix} C^1_{\frac{N}{4}+1} & 0 \\ 0 & \bar{I}\, C^3_{\frac{N}{4}}\, \bar{I} \end{bmatrix} \left[A^2_{\frac{N}{2}+1}\right] \tag{4.13a}$$

$$\left[S^1_{\frac{N}{2}-1}\right] = \left[P_{\frac{N}{2}-1}\right] \begin{bmatrix} S^3_{\frac{N}{4}} & 0 \\ 0 & \bar{I}\, S^1_{\frac{N}{4}-1}\, \bar{I} \end{bmatrix} \left[A^2_{\frac{N}{2}-1}\right] \tag{4.13b}$$

where for J odd we have

$$[P_J] = \begin{bmatrix} 1 & 0 & . & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & . & 1 \\ 0 & 1 & 0 & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & 1 & 0 \\ . & . & . & . & . & . & . & . & . \\ 0 & . & . & . & 1 & 0 & . & . & . \\ 0 & . & . & . & . & 0 & 1 & . & 0 \\ 0 & . & . & . & . & 1 & 0 & . & . \end{bmatrix} \tag{4.14}$$

$$[A^2_J] = \frac{1}{\sqrt{2}} \begin{bmatrix} I_{\frac{J-1}{2}} & 0 & \bar{I}_{\frac{J-1}{2}} \\ 0 & \sqrt{2} & 0 \\ \bar{I}_{\frac{J-1}{2}} & 0 & -I_{\frac{J-1}{2}} \end{bmatrix} \tag{4.15}$$

and $\left[C^3_N\right]$ and $\left[S^3_N\right]$ stand for N-point discrete cosine and sine transforms of the third kind (DCT3 and DST3) defined in equation (4.11).

In the second step of the algorithm we showed that an $(\frac{N}{2}+1)$-point DCT1 can be decomposed into an $(\frac{N}{4}+1)$-point DCT1 and an $\frac{N}{4}$-point DCT3. In order to carry on the recursion further, we have to find a way of computing the DCT3 of the second step. Therefore the third step of the algorithm is to consider the even and odd points of the sequence $x_4(n)$ in equation (4.12b) of the second step separately. Thus we get:

$$C^3_{\frac{N}{4}}(r) = \sum_{n=0}^{\frac{N}{8}} x_4(2n)\cos\left(\frac{\pi n(r+\frac{1}{2})}{N/8}\right) + \sum_{n=0}^{\frac{N}{8}} x_4(2n+1)\cos\left(\frac{\pi(n+\frac{1}{2})(r+\frac{1}{2})}{N/8}\right) \tag{4.16}$$

Again the first and second sum can be identified as $\frac{N}{8}$-point DCT3 and DCT4 of the even and odd points of the sequence $x_4(n)$ respectively where DCT4 and DST4 of an N-point sequence $x(n)$ are defined as:

$$C^4_N(k) = \sum_{n=0}^{N-1} x(n)\cos\left(\frac{\pi(n+\frac{1}{2})(k+\frac{1}{2})}{N}\right) \tag{4.17a}$$

$$S^4_N(k) = \sum_{n=0}^{N-1} x(n)\sin\left(\frac{\pi(n+\frac{1}{2})(k+\frac{1}{2})}{N}\right) \tag{4.17b}$$

The same approach can be used to decompose the $\frac{N}{4}$-point DST3 of the second part of the algorithm into an $\frac{N}{8}$-point DST3 and an $\frac{N}{8}$-point DST4.

In terms of matrices, the third step in Wang's algorithm involves decomposing $\left[C^3_{\frac{N}{4}}\right]$ and $\left[S^3_{\frac{N}{4}}\right]$ obtained in equation 4.13 of the second step into $\left[C^3_{\frac{N}{8}}\right]$, $\left[C^4_{\frac{N}{8}}\right]$, $\left[S^3_{\frac{N}{8}}\right]$ and $\left[S^4_{\frac{N}{8}}\right]$. In matrix notation this can be written as

$$\left[C^3_{\frac{N}{4}}\right] = \left[A^2_{\frac{N}{4}}\right] \begin{bmatrix} C^3_{\frac{N}{8}} & 0 \\ 0 & \bar{I}\, C^4_{\frac{N}{8}}\, \bar{I} \end{bmatrix} \left[P_{\frac{N}{4}}\right]^T \tag{4.18a}$$

$$\left[S^3_{\frac{N}{4}}\right] = \left[A^2_{\frac{N}{4}}\right] \begin{bmatrix} S^4_{\frac{N}{8}} & 0 \\ 0 & \bar{I}\, S^3_{\frac{N}{8}}\, \bar{I} \end{bmatrix} \left[P_{\frac{N}{4}}\right]^T \tag{4.18b}$$

where for J even $\left[P_J\right]$ and $\left[A^2_J\right]$ are defined as:

$$[P_J] = \begin{bmatrix} 1 & 0 & . & . & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & . & . & 1 \\ 0 & 1 & 0 & . & . & . & . & . & . & . \\ 0 & . & . & . & . & . & . & . & 1 & 0 \\ . & . & . & . & . & . & . & . & . & . \\ 0 & . & . & . & . & 1 & 0 & . & . & . \\ 0 & . & . & 1 & 0 & . & . & . & . & . \\ 0 & . & . & . & 1 & 0 & . & . & . & . \end{bmatrix} \tag{4.19a}$$

$$\left[A_J^2\right] = \frac{1}{\sqrt{2}} \begin{bmatrix} I_{\frac{J}{2}} & \bar{I}_{\frac{J}{2}} \\ \bar{I}_{\frac{J}{2}} & -I_{\frac{J}{2}} \end{bmatrix} \tag{4.19b}$$

and $\left[C_N^4\right]$ and $\left[S_N^4\right]$ are N-point discrete cosine and sine transform matrices of the forth kind (DCT4 and DST4) defined in (4.17).

To summarize, in the first step of the algorithm we decomposed an N-point Hartley transform into an $(\frac{N}{2}+1)$ and $(\frac{N}{2}-1)$-point DCT1 and DST1 respectively. In the second step the $(\frac{N}{2}+1)$-point DCT1 of the first step is decomposed into an $(\frac{N}{4}+1)$-point DCT1 and an $\frac{N}{4}$-point DCT3. A similar procedure was applied to the DST1 of the first step. In the third step, $\frac{N}{4}$-point DCT3 of the second step is decomposed into an $\frac{N}{8}$-point DCT3 and an $\frac{N}{8}$-point DCT4 and the $\frac{N}{4}$-point DST3 of the second step is decomposed into an $\frac{N}{8}$-point DST3 and an $\frac{N}{8}$-point DST4. Thus the remaining part of Wang's algorithm involves finding an efficient way of computing the DST4 and DCT4.

The DST4 of an N-point real sequence $x(n)$ denoted by $S_N^4(k)$ can be computed via an DCT4 algorithm in the following manner:

$$S_N^4(N-k-1) = \sum_{n=0}^{N-1} (-1)^n\, x(n) \cos\left(\frac{\pi(n+\frac{1}{2})(k+\frac{1}{2})}{N}\right)$$

Therefore it is sufficient to find an efficient algorithm for one of them only. The rest of this chapter will describe a way of computing the DCT4 efficiently. The detailed description of factorizing $\left[C_N^4\right]$ is included in Chen [14] and Wang [5],[15]. Here we will go over the basic idea behind the matrix factorization described in the mentioned papers.

The DCT4 of a sequence $x(n)$ described in equation (4.17a) can be written as :

$$C_N^4(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n)\cos\left(\frac{\pi(2n+\frac{1}{2})(k+\frac{1}{2})}{N}\right) + (-1)^k x(N-2n-1)\sin\left(\frac{\pi(2n+\frac{1}{2})(k+\frac{1}{2})}{N}\right) \quad (4.20)$$

Let us now consider the even and odd frequency points of $C_N^4(k)$ denoted by $C(2m)$ and $C(2m+1)$ separately. Then we get

$$C(2m) = \sum_{n=0}^{\frac{N}{2}-1} y_1(n)\cos\left(\frac{\pi(4n+1)m}{N}\right) - y_2(n)\sin\left(\frac{\pi(4n+1)m}{N}\right) \quad (4.21a)$$

$$C(2m+1) = \sum_{n=0}^{\frac{N}{2}-1} y_1(n)\cos\left(\frac{\pi(4n+1)(m+1)}{N}\right) + y_2(n)\sin\left(\frac{\pi(4n+1)(m+1)}{N}\right) \quad (4.21b)$$

where

$$y_1(n) = x(2n)\cos\left(\frac{\pi(4n+1)}{4N}\right) + x(N-2n-1)\sin\left(\frac{\pi(4n+1)}{4N}\right) \quad (4.22a)$$

$$y_2(n) = x(2n)\sin\left(\frac{\pi(4n+1)}{4N}\right) - x(N-2n-1)\cos\left(\frac{\pi(4n+1)}{4N}\right) \quad (4.22b)$$

After simple algebraic manipulations equations (4.21a) and (4.21b) can be written as:

$$C(2m) = \sum_{n=0}^{\frac{N}{4}-1} [y_1(n) + y_1(\frac{N}{4}+n)(-1)^m]cos(\frac{\pi(4n+1)m}{N}) \tag{4.23a}$$

$$- [y_2(n) + y_2(\frac{N}{4}+n)(-1)^m]sin(\frac{\pi(4n+1)m}{N})$$

$$C(2m+1) = \sum_{n=0}^{\frac{N}{4}-1} [y_1(n) + y_1(\frac{N}{4}+n)(-1)^{m+1}]cos(\frac{\pi(4n+1)(m+1)}{N}) \tag{4.23b}$$

$$+[y_2(n) + y_2(\frac{N}{4}+n)(-1)^{m+1}]sin(\frac{\pi(4n+1)(m+1)}{N})$$

The above equations can in turn be decomposed for odd and even values of m. Let odd and even values of m be denoted by $2r+1$ and $2r$ respectively. Then after substituting odd and even values of m in equation (4.23) we get a set of four equations:

$$C(4r) = \sum_{n=0}^{\frac{N}{4}-1} y_3(n)\cos(\frac{\pi(4n+1)r}{N/2}) - y_4(n)\sin(\frac{\pi(4n+1)r}{N/2}) \tag{4.24a}$$

$$C(4r+1) = \sum_{n=0}^{\frac{N}{4}-1} y_5(n)\cos(\frac{\pi(4n+1)r}{N/2}) - y_6(n)\sin(\frac{\pi(4n+1)r}{N/2}) \tag{4.24b}$$

$$C(4r+2) = \sum_{n=0}^{\frac{N}{4}-1} y_5(n)\cos(\frac{\pi(4n+1)(r+1)}{N/2}) + y_6(n)\sin(\frac{\pi(4n+1)(r+1)}{N/2}) \tag{4.24c}$$

$$C(4r+3) = \sum_{n=0}^{\frac{N}{4}-1} y_3(n)\cos(\frac{\pi(4n+1)(r+1)}{N/2}) + y_4(n)\sin(\frac{\pi(4n+1)(r+1)}{N/2}) \tag{4.24d}$$

where

$$y_3(n) = y_1(n) + y_1(n+\frac{N}{4}) \tag{4.25a}$$

$$y_4(n) = y_2(n) + y_2(n+\frac{N}{4}) \tag{4.25b}$$

$$y_5(n) = [y_1(n)-y_1(n+\frac{N}{4})]\cos(\frac{\pi(4n+1)}{N}) + [y_2(n)-y_2(n+\frac{N}{4})]\sin(\frac{\pi(4n+1)}{N}) \qquad (4.25c)$$

$$y_6(n) = [y_1(n)-y_1(n+\frac{N}{4})]\sin(\frac{\pi(4n+1)}{N}) - [y_2(n)-y_2(n+\frac{N}{4})]\cos(\frac{\pi(4n+1)}{N}) \qquad (4.25d)$$

Comparing equations (4.21) and (4.24) we can see that a problem of size $\frac{N}{2}$ has been converted into two problems of size $\frac{N}{4}$. More specifically equation (4.21a) has been decomposed into (4.24a) and (4.24b) and equation (4.21b) has been decomposed into (4.24c) and (4.24d). By repeating the above process we can carry on the decomposition further. This completes the outline of DCT4 algorithm and the last step of Wang's algorithm.

We will be referring to Wang's algorithm as the DT3 algorithm for the remainder of this thesis. The DT3 algorithm requires on the order of $\frac{3N}{4}\log_2 N$ real multiplications and $\frac{7N}{4}\log_2 N$ real additions. Hence its total operation count is the same as a real-valued radix 2 FFT; however, the indexing scheme is substantially more complex for the DT3 algorithm than it is for the FFT.

CHAPTER 5: New Discrete Hartley Transform Algorithms

In chapters 3 and 4, we reviewed Bracewell and Wang DHT algorithms. This chapter is concerned with several new methods for computing the discrete Hartley transform. We shall begin by describing a new radix 2 decimation-in-time algorithm which will be referred to as the DT2 algorithm. The idea behind the DT2 algorithm will then be extended to decimation-in-frequency, radix 4 and split radix algorithms. The second part of this chapter will introduce the chirp Hartley transform algorithm which is similar to the chirp z-transform algorithm for computing the DFT [1]. The error properties of these algorithms are explored in future chapters.

5.1. A New Algorithm for Computing the DHT

5.1.1. Radix 2 Decimation-in-Time (DT2) Algorithm

This section derives a new radix 2 decimation-in-time DHT algorithm which we will refer to as the DT2 algorithm. To achieve substantial efficiency in computing the DHT, it is necessary to decompose it into successively smaller DHT computations. The principle of the decimation-in-time algorithm is most conveniently illustrated by considering the special case of N an integer power of 2; i.e.,

$$N = 2^{\nu}$$

Since ν is even, we can consider computing $H(k)$ by separating $x(n)$ into two $\frac{N}{2}$-point sequences consisting of the even and odd numbered points in $x(n)$. Therefore equation (2.7) can be written as

$$H(k) = H_1(k) + H_2(k) \tag{5.1}$$

where

$$H_1(k) = \sum_{n=0}^{N-1} x(2n)\, cas\left(\frac{2\pi nk}{N/2}\right) \tag{5.2a}$$

$$H_2(k) = \sum_{n=0}^{N-1} x(2n+1) cas\left(\frac{2\pi(2n+1)k}{N}\right) \tag{5.2b}$$

$H_1(k)$ is an $\frac{N}{2}$-point Hartley transform. Using the identity

$$2\cos(\beta)\, cas(\alpha) = cas(\alpha + \beta) + cas(\alpha - \beta) \tag{5.3}$$

and letting $\alpha = \frac{2\pi(2n+1)k}{N}$ and $\beta = \frac{2\pi k}{N}$ and multiplying both sides of equation (5.2b) by $\cos(\frac{2\pi k}{N})$, $H_2(k)$ of equation (5.2b) can be written as:

$$H_2(k) = \begin{cases} \frac{1}{2\cos(\frac{2\pi k}{N})} \sum_{n=0}^{\frac{N}{2}-1} [\, x(2n+1) + x(2n-1)\,] cas(\frac{2\pi nk}{N/2}) & 0 \le k < \frac{N}{2},\ k \ne \frac{N}{4} \\ \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)\,(-1)^n & k = \frac{N}{4} \end{cases} \tag{5.4a}$$

$$H_2(k) = -H_2(k - \frac{N}{2}) \qquad \frac{N}{2} \le k < N \tag{5.4b}$$

where

$$x(-1) \equiv x(N-1)$$

Equation (5.4a) shows that $H_2(k)$ can also be computed via an $\frac{N}{2}$-point DHT. Therefore we have demonstrated that an N-point DHT can be obtained by computing two $\frac{N}{2}$-point DHTs. By repeating the above process we can decompose the DHT further. Computing the DHT of an N-point real sequence can thus be accomplished with $\frac{N}{2}\log_2 N$ real multiplies and $2N\log_2 N$ real additions.

Note that equation (5.2) is identical to equation (3.2) which was used to derive Bracewell's algorithm. The difference between the two algorithms however, is the identity used in computing $H_2(k)$ of equation (5.2b).

The flow graph corresponding to the DT2 algorithm for $N = 16$ is shown in figure 5.1. Note that the special case of $k = \frac{N}{4}$ in equation (5.4a) is not treated separately in the diagram for clarity. Conceptually, one can think of the algorithm as having two major parts: In the rearrangement section the even points of the subsequences are grouped together and the odd points are added and grouped together; In the recombination stage the multiplication by $\frac{1}{2\cos(\frac{2\pi k}{N})}$ is performed and the subsequences are combined in a butterfly type manner. The *kth* butterfly of the last stage of the recombination part of the algorithm is shown in figure 5.2.

It is interesting to note that although the algorithm can be implemented in place, in the *ith* rearrangement stage we need to store 2^{i-1} values to accommodate for the special case of $k = \frac{N}{4}$ in equation (5.5). This will require

$$\sum_{i=1}^{v-2} 2^{i-1} = \frac{N}{4} - 1 \qquad (5.5)$$

additional storage space beyond the N points of the array which is being processed. In computing the DFT of an N-point real sequence via FFT, no additional storage beyond N points is required.

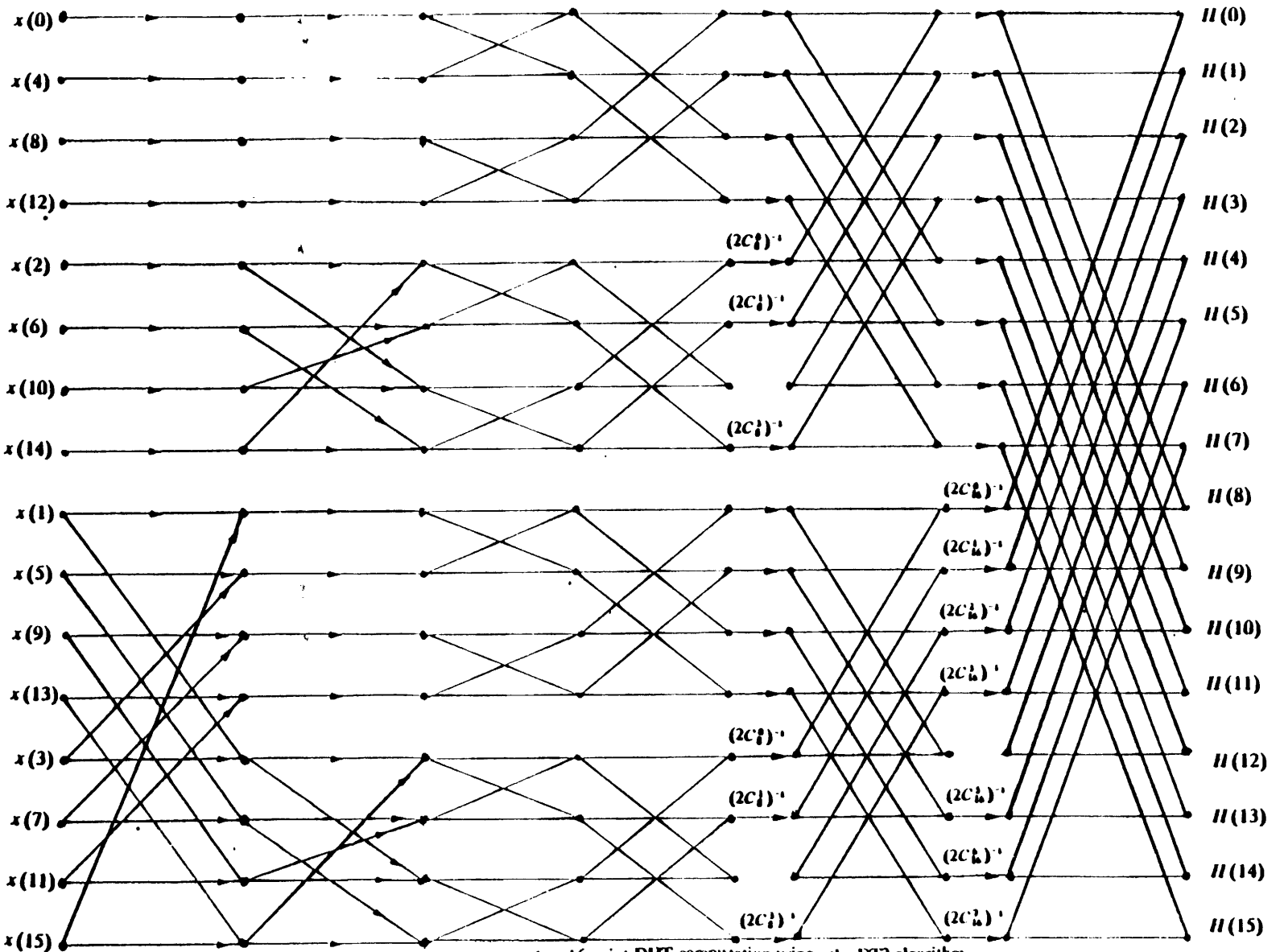


Fig. 5.1 Flow graph of the decimation-in-time decomposition of an 16-point DHT computation using the DIT2 algorithm

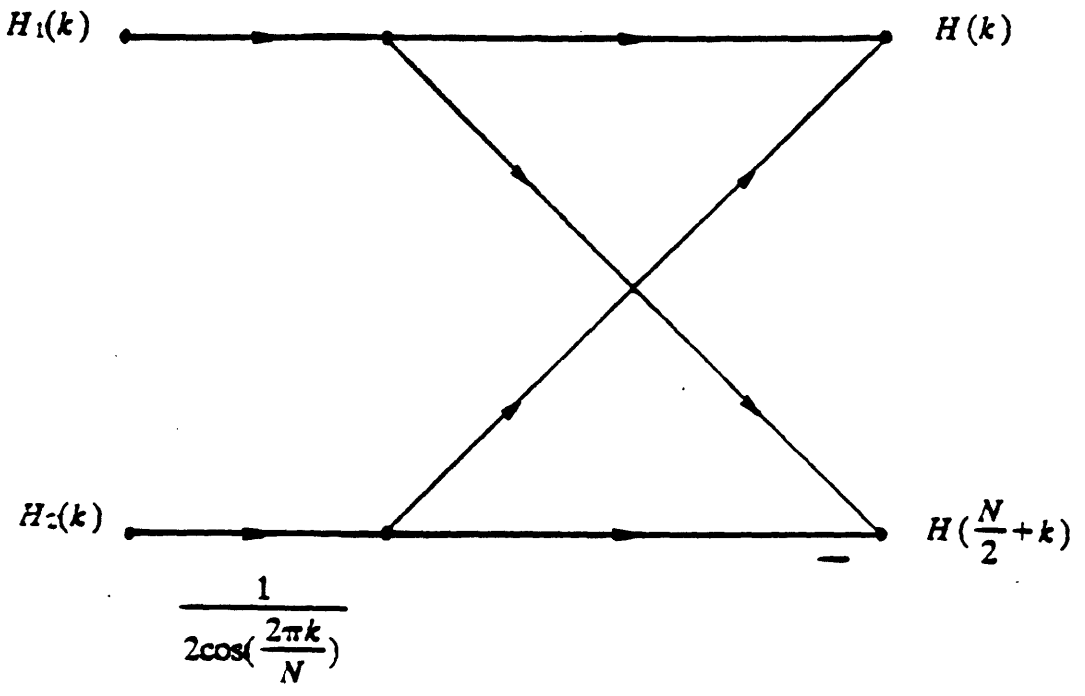


Fig. 5.2 Flow graph of the *kth* butterfly of the last stage of an N-point DHT computation using the DT2 algorithm

The rearrangement of the odd points of the sequence as shown in equation (5.4a) can be generalized by multiplying both sides of equation (5.2b) by $\cos(\frac{2\pi k(2r+1)}{N})$ and using the identity (5.3) with $\alpha = \frac{2\pi k(2n+1)}{N}$ and $\beta = \frac{2\pi k(2r+1)}{N}$ in the following manner:

$$H_2(k) = \begin{cases} \frac{1}{\cos(\frac{2\pi k(2r+1)}{N})} \sum_{n=0}^{\frac{N}{2}-1} [\, x((2n-2r-1))_N R_N(n) + x((2n+2r+1))_N R_N(n) \,]\, cas(\frac{2\pi kn}{N/2}) & k \neq \frac{N}{4} \\ \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)(-1)^n & k = \frac{N}{4} \end{cases} \quad (5.6)$$

where r is an arbitrary integer. We will show in future chapters that by picking various values of r we can change the distribution of the variance of the error at the output of the transform.

5.1.2. Radix 4 Decimation-in-time (R4DT2) Algorithm

A radix 4 version of the DT2 algorithm, which we will refer to as the R4DT2 algorithm, can be easily derived by dividing the original sequence $x(n)$ into 4 subsequences of length $\frac{N}{4}$. Thus $H(k)$ can be written as:

$$H(k) = H_0(k) + H_1(k) + H_2(k) + H_3(k) \quad (5.7)$$

where

$$H_i(k) = \sum_{n=0}^{\frac{N}{4}-1} x(4n+i)cas(\frac{2\pi(4n+i)k}{N}) \quad (5.8)$$

First we will consider sequences $H_0(k)$ and $H_2(k)$. It can be shown that

$$H_0(k) = \sum_{n=0}^{\frac{N}{4}-1} x(4n)cas(\frac{2\pi nk}{N/4}) \tag{5.9}$$

Choosing $i = 2$ in equation (5.8) and multiplying both sides of it by $\cos(\frac{4\pi k}{N})$ and using identity (5.3) with $\alpha = \frac{2\pi(4n+2)k}{N}$ and $\beta = \frac{4\pi k}{N}$, $H_2(k)$ can be written as:

$$H_2(k) = \begin{cases} \frac{1}{2\cos(\frac{4\pi k}{N})} \sum_{n=0}^{n=\frac{N}{4}-1} [x((4n-2))_N R_N(n) + x(4n+2)]cas(\frac{2\pi nk}{N/4}) & 0 \le k < \frac{N}{4}, \quad k \neq \frac{N}{8} \\ \sum_{n=0}^{\frac{N}{4}-1} x(4n+2)(-1)^n & k = \frac{N}{8} \end{cases} \tag{5.10a}$$

$$H(k) = -H(k+\frac{N}{4}) = H(k+\frac{N}{2}) = -H(k+\frac{3N}{4}) \qquad 0 \le k < \frac{N}{4} \tag{5.10b}$$

Thus $H_2(k)$ can be recognized as an $\frac{N}{4}$-point DHT. The sum of $H_1(k)$ and $H_3(k)$ which corresponds to the DHT of the odd points of the original sequence can be written as:

$$H_1(k) + H_3(k) = \begin{cases} \frac{1}{2\cos(\frac{2\pi k}{N})}[G_1(k) + G_2(k)] & 0 \le k < \frac{N}{2}, k \neq \frac{N}{4} \\ \sum_{n=0}^{\frac{N}{2}-1} x(2n+1)(-1)^n & k = \frac{N}{4} \end{cases} \tag{5.11a}$$

$$H_1(k) + H_3(k) = -[H_1(k+\frac{N}{2}) + H_3(k+\frac{N}{2})] \qquad 0 \le k < \frac{N}{2} \tag{5.11b}$$

where

$$G_1(k) = \sum_{n=0}^{\frac{N}{4}-1} [x(4n+1) + x((4n-1))_N R_N(n)]cas(\frac{2\pi nk}{N/4}) \tag{5.12a}$$

and

$$G_2(k) = \begin{cases} \dfrac{1}{2\cos(\frac{4\pi k}{N})} \sum_{n=0}^{\frac{N}{4}-1} g_2(n)\,cas(\frac{2\pi nk}{N/4}) & 0 \le k < \frac{N}{4},\ k \neq \frac{N}{8} \\ \sum_{n=0}^{\frac{N}{4}-1} [x(4n+1) + x(4n+3)](-1)^n & k = \frac{N}{8} \end{cases} \tag{5.12b}$$

$$g_2(n) = x(4n+1) + x(4n+3) + x((4n-1))_N R_N(n) + x((4n-3))_N R_N(n) \tag{5.12c}$$

$$G_2(k) = -G_2(k-\frac{N}{4}) \qquad \frac{N}{4} \le k < \frac{N}{2} \tag{5.12d}$$

Equation (5.12) shows that the sum $H_1(k) + H_3(k)$ can be computed via two $\frac{N}{4}$-point DHTs. Thus the problem of computing an N-point DHT has been reduced to that of finding four $\frac{N}{4}$-point DHTs. By repeating the above procedure, DHT can be decomposed further. Clearly, at every stage we need to do N multiplies, $\frac{7N}{4}$ real adds for forming new sequences and taking care of the special cases, and $2N$ adds for the butterflies in the recombination stage of the algorithm. Thus the number of multiplies for a sequence of N numbers is on the order of $N\log_4 N$ and the number of adds is on the order of $\frac{15N}{4}\log_4 N$.

The *kth* butterfly of the last stage of the DF1 algorithm for an N point sequence and the signal flow graph of the algorithm for an 16-point sequence are shown in figures 5.3 and 5.4 respectively. It is important to bear in mind that although the operation count is lower for the radix four algorithm than it is for the radix two algorithm, the former is more complex to implement; This has to do with the relative complexity of the basic unit of computation for the two algorithms shown in figures 5.1 and 5.3.

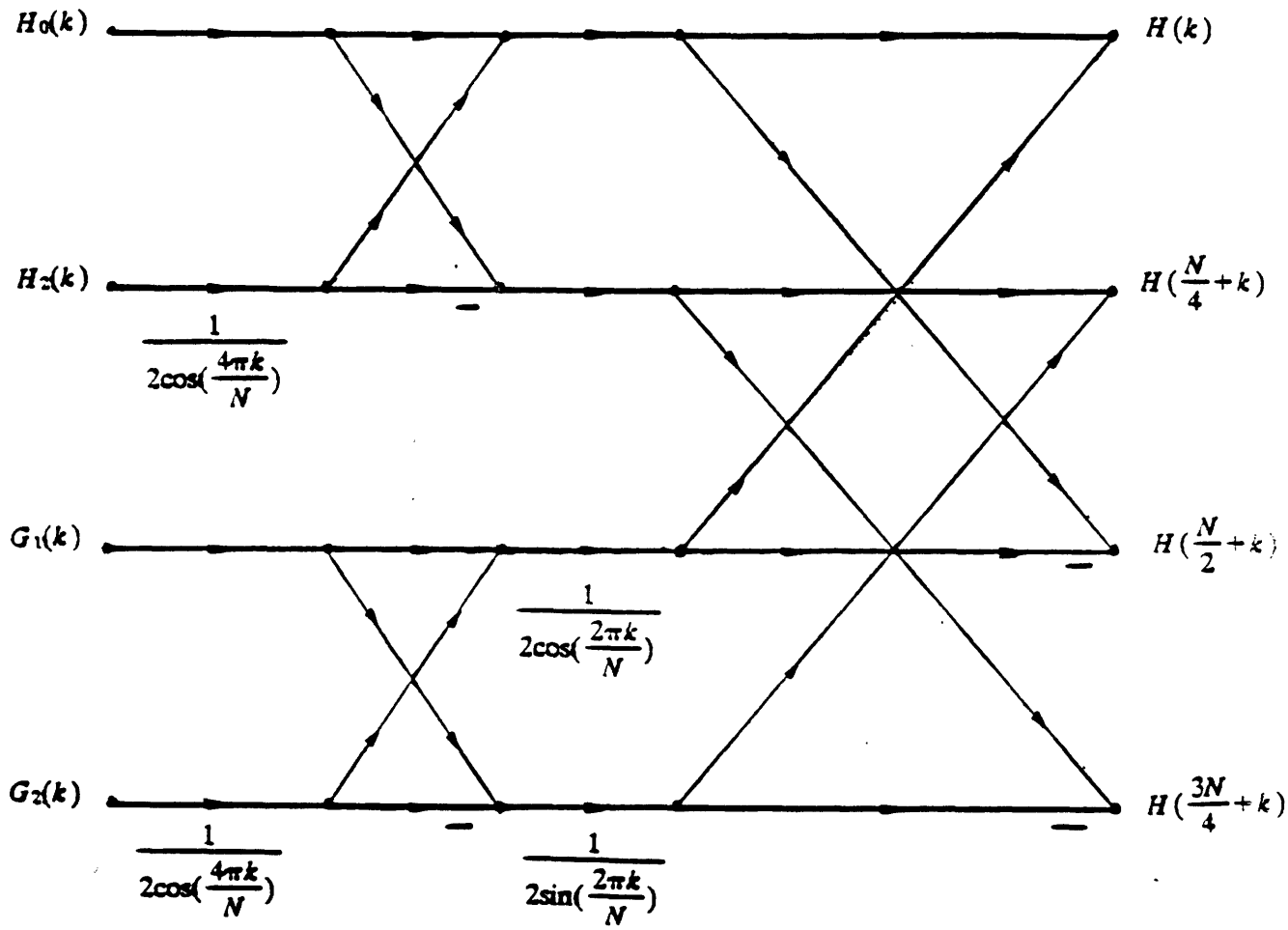


Fig. 5.3 Flow graph of the *kth* butterfly of the last stage of an N-point DHT computation using the R4DT2 algorithm

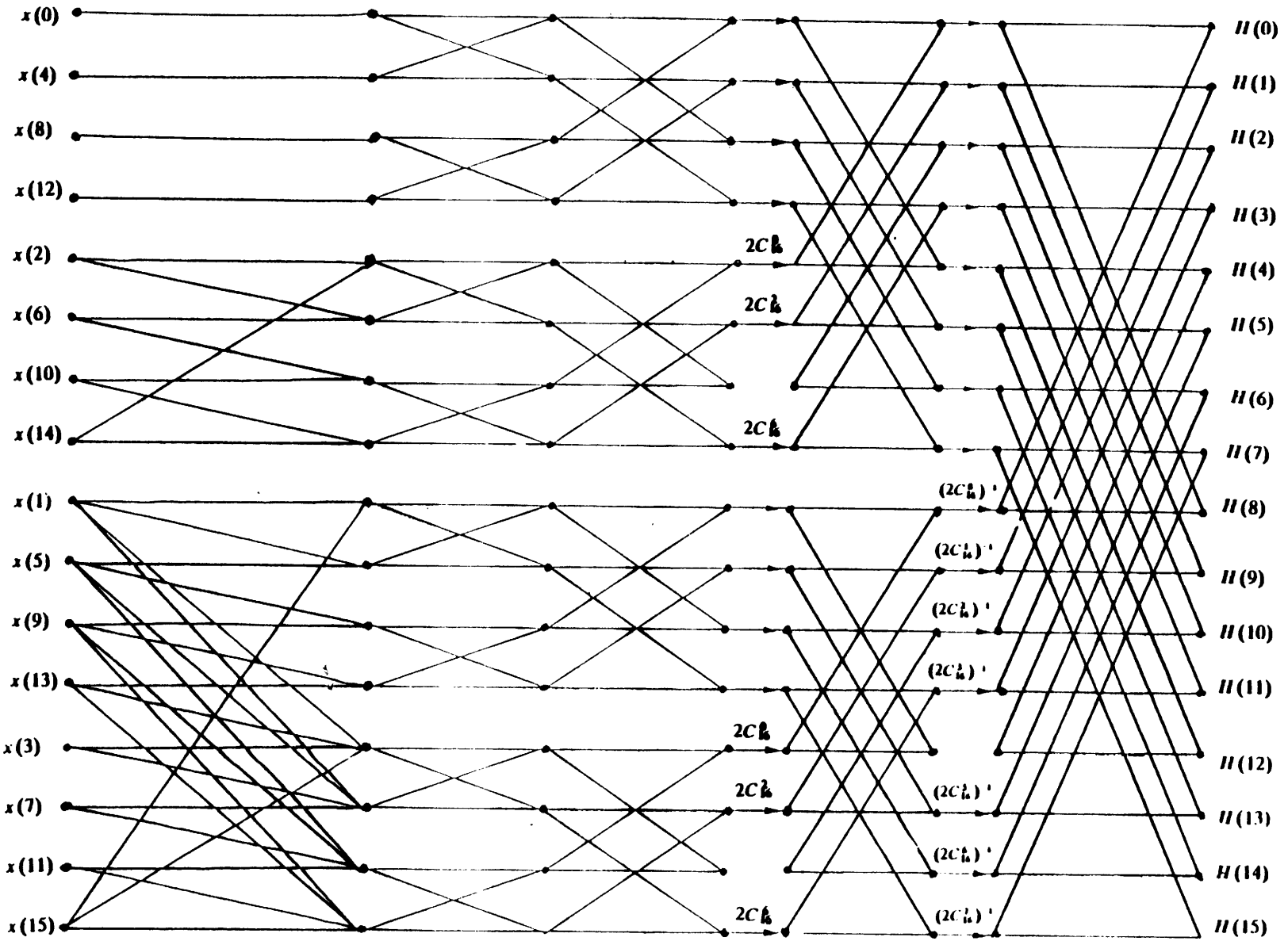


Fig. 5.4 Flow graph of the decimation-in-time decomposition of an 16-point DHT computation using the R4DT2 algorithm

5.1.3. Split Radix Decimation-in-Time (SRDT2) Algorithm

A Split radix version of the DT2 algorithm which we will refer to as the SRDT2 algorithm, is obtained by applying a radix two decomposition to the even indexed samples and a radix four decomposition to the odd indexed samples of the input sequence. Following the notation introduced in equation (5.8) we get :

$$H(k) = [H_0(k) + H_2(k)] + [H_1(k) + H_3(k)] \tag{5.13}$$

The second sum which corresponds to two $\frac{N}{4}$-point DHTs can be evaluated using equations (5.11) and (5.12) of the previous section. The first sum can be written as

$$H_0(k) + H_2(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n)\, cas(\frac{2\pi nk}{N/2}) \tag{5.14}$$

which can be identified as another $\frac{N}{2}$-point DHT. Thus we have reduced the problem of finding one N-point DHT to one $\frac{N}{2}$ and two $\frac{N}{4}$-point DHTs. This process can be repeated in order to decompose the DHT further.

The split radix algorithm presented here does not progress stage by stage or in terms of indices, does not complete each nested sum in order. This makes the indexing structure much more complex than the fixed radix algorithms described earlier.

5.1.4. Radix 2 Decimation-in-Frequency (DF2) Algorithm

The decimation-in-time algorithm was based upon the DHT computation by forming smaller and smaller subsequences of the input sequence, $x(n)$. Alternatively, we can consider dividing the output sequence, $H(k)$ into smaller and smaller

subsequences in the same manner. In the decimation-in-frequency version of the DT2 algorithm which we will refer to as the DF2 algorithm, we can first divide the input sequence into the first half and the last half of the points so that

$$H(k) = H_1(k) + H_2(k) \tag{5.15}$$

where

$$H_1(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n) cas(\frac{2\pi nk}{N}) \tag{5.16a}$$

$$H_2(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n+\frac{N}{2}) (-1)^k cas(\frac{2\pi nk}{N}) \tag{5.16b}$$

Let us now consider k even and k odd separately, with $H(2r)$ and $H(2r+1)$ representing the even and odd numbered points of $H(k)$ respectively, so that

$$H(2r) = \sum_{n=0}^{\frac{N}{2}-1} [x(n) + x(\frac{N}{2}+n)] cas(\frac{2\pi nr}{N/2}) \tag{5.17a}$$

$$H(2r+1) = \sum_{n=0}^{\frac{N}{2}-1} [x(n) - x(\frac{N}{2}+n)] cas(\frac{2\pi n(2r+1)}{N}) \tag{5.17b}$$

Equation (5.17a) can be recognized as an $\frac{N}{2}$-point DHT; Multiplying and dividing the right hand side of equation (5.17b) by $\cos(\frac{2\pi n}{N})$ and using the identity (5.3) with $\alpha = \frac{2\pi n(2r+1)}{N}$ and $\beta = \frac{2\pi n}{N}$ we get:

$$H(2r+1) = (-1)^r [x(\frac{N}{4}) - x(\frac{3N}{4})] + G(r) + G((r+1))_{N/2} R_{N/2}(n) \tag{5.18}$$

where

$$G(r) = \sum_{\substack{n=0 \\ n \neq \frac{N}{4}}}^{\frac{N}{2}-1} \frac{x(n) - x(n+\frac{N}{2})}{2\cos(\frac{2\pi n}{N})} \, cas(\frac{2\pi nr}{N/2}) \qquad (5.19)$$

$G(r)$ is also an $\frac{N}{2}$-point DHT . Therefore once again an N-point DHT has been decomposed into two $\frac{N}{2}$-point DHTs. Repeating the above process decomposes the DHT further. The arithmetic count as well as the storage requirements for decimation-in-frequency algorithm are identical to that of the decimation-in-time version. The *kth* butterfly of the first stage of an N-point transform using the DF2 algorithm is shown in figure 5.5. The flow graph corresponding to this algorithm for an 16-point sequence is shown in figure 5.6. Comparing figures 5.1 and 5.6 we see that the rearrangement and butterfly computations are distinctly different for the two classes of the DHT algorithms. However, we also notice some similarity between their basic structures. Indeed, figure 5.1 can be obtained from figure 5.6 by reversing the direction of signal flow and interchanging the input and output. (Note that in figures 5.1 and 5.6 the special cases $k = \frac{N}{4}$ for decimation-in-time and $n = \frac{N}{4}$ for decimation-in-frequency which require separate computations are not treated separately in the diagram for clarity). Consequently, by the transposition theorem the input-output characteristics of the two flow graphs must be the same.

As mentioned in earlier chapters, in order to perform an inverse FFT, the transmittance of all the branches in the flow graph of the forward FFT has to be

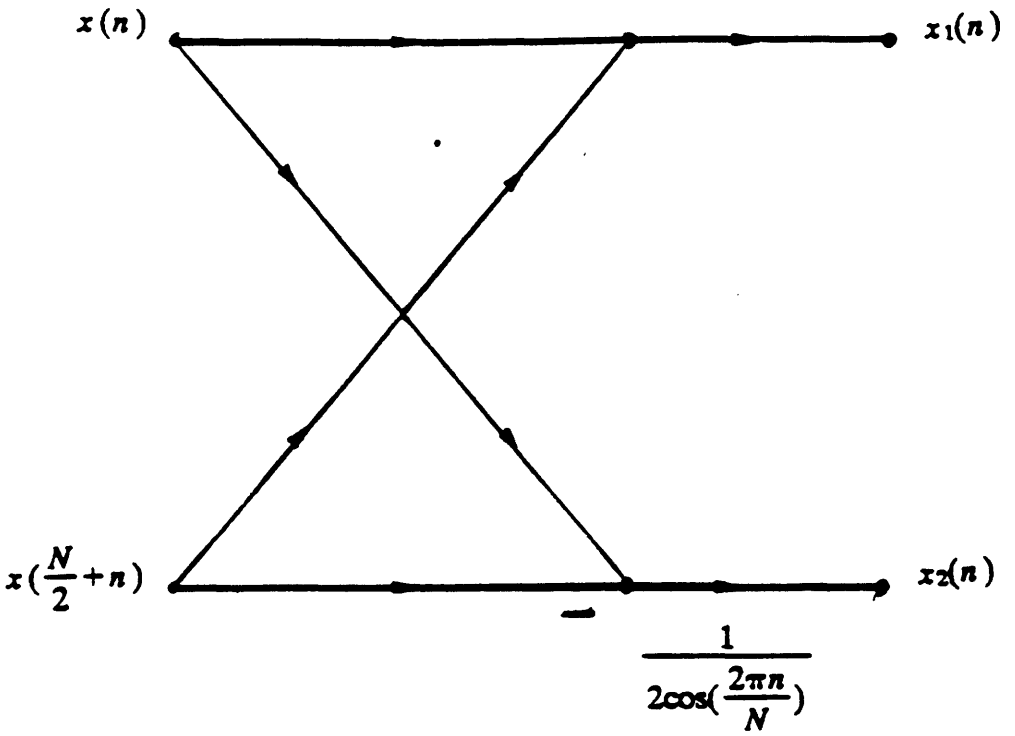


Fig. 5.5 Flow graph of the *nth* butterfly of the first stage of an N-point DHT computation using the DF2 algorithm

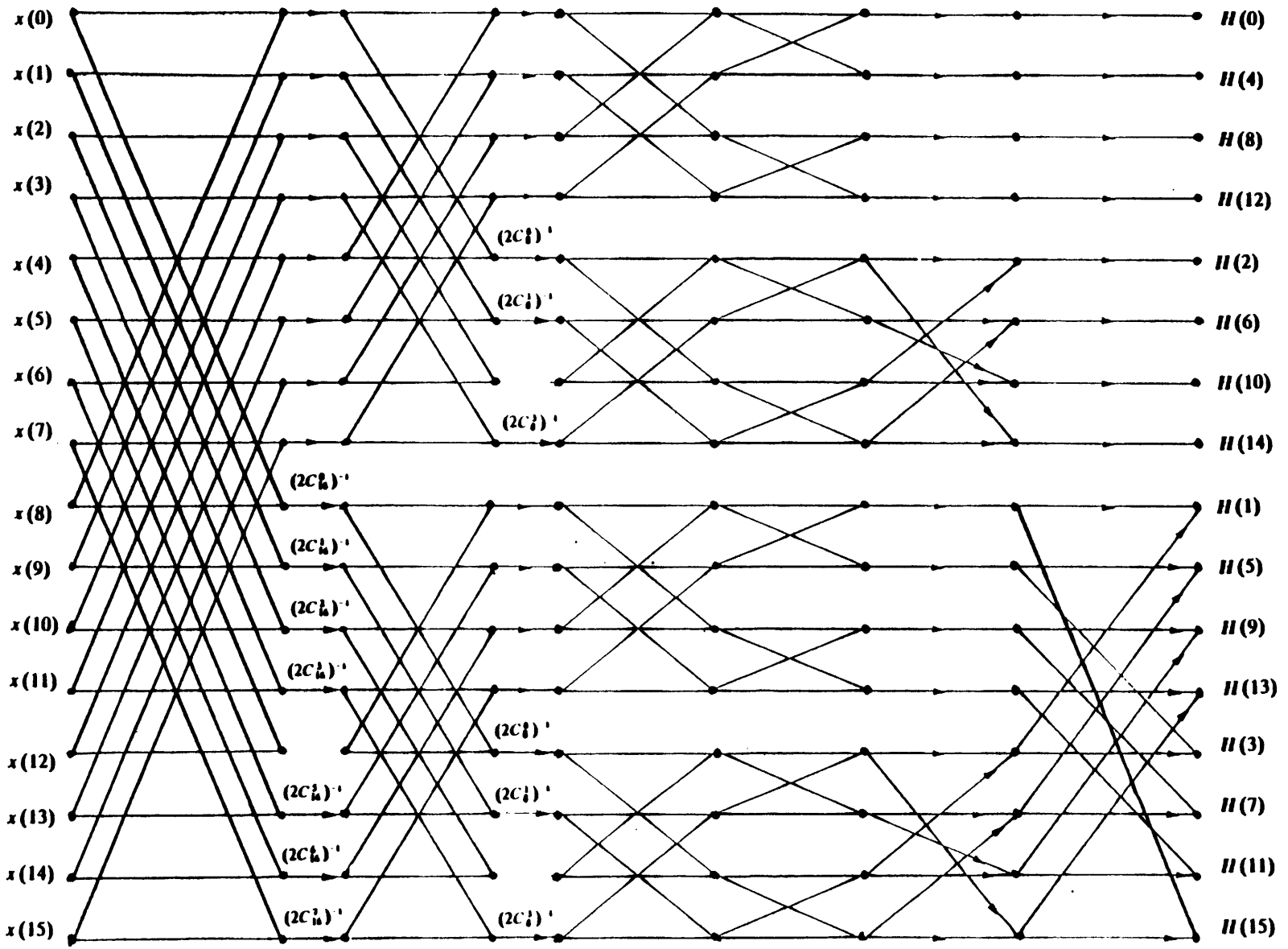


Fig. 5.6 Flow graph of the decimation-in-frequency decomposition of an 16-point DHT computation using the DF2 algorithm

conjugated. This corresponds to using powers of $e^{j\frac{2\pi k}{N}}$ instead of powers of $e^{-j\frac{2\pi k}{N}}$ or rearranging the sequence in order to be able to use the same flow graph for forward and inverse FFT. For the DHT on the other hand the flow graph is the same for forward and inverse transforms and no rearrangement is necessary.

The radix 4 and split radix version of the DF2 algorithm discussed here, can be derived in a similar manner to the corresponding decimation-in-time algorithms.

5.2. Chirp Hartley Transform (CHT) Algorithm

The DHT may be computed using an algorithm similar to the chirp z-transform (CZT), a method which is suitable for implementation via acoustic surface wave devices or charge coupled devices (CCD) as well as other forms of transversal filters.

The CZT algorithm was first proposed in 1968 [16]. It was directed toward computation of samples of the z-transform on a spiral contour equally spaced in angle over some portion of the unit circle. More specifically, let $x(n)$ denote an N-point sequence with $X(z)$ representing its z-transform. Using the CZT algorithm, $X(z)$ can be computed at the points z_k given by

$$z_k = A_0 e^{j\theta_0}(W_0 e^{j\phi_0})^k \qquad k=0,1,\ldots,M-1 \qquad (5.20)$$

The parameter W_0 controls the rate at which the contour spirals; These samples are located along the spiral contour with an angular spacing of ϕ_0. Since the DFT of an N-point sequence is equally spaced samples of its z-transform on the unit circle, by choosing $\theta_0=0$, $A_0=1$, $W_0=1$, $M=N$ and $\phi_0=\frac{2\pi}{N}$, in equation (5.20) and com-

puting $X(z_k)$, we have effectively calculated the DFT of the sequence. It can be shown [1] that using the CZT algorithm the DFT of an N-point sequence $F(k)$ can be written as:

$$F(k) = W^{\frac{k^2}{2}} \sum_{n=0}^{N-1} [\, x(n)\, W^{\frac{n^2}{2}}]\, W^{-\frac{(k-n)^2}{2}} \tag{5.21}$$

where

$$W = e^{-j\frac{2\pi}{N}} \tag{5.22}$$

The summation in equation (5.21) can be recognized as the convolution of the sequence $x(n)W^{\frac{n^2}{2}}$ with the sequence $W^{-\frac{n^2}{2}}$. The computation of the equation (5.21) is depicted in figure 5.7. Implementing the complex arithmetic convolution of equation (5.21) with real hardware requires the use of four convolvers. This is shown more clearly in figure 5.7. The incoming signal is multiplied by the real and imaginary part of $W^{\frac{n^2}{2}}$ and combined in pairs to drive the inputs of four chirp convolvers. The convolver outputs are combined in pairs and multiplied by the real and imaginary components of the postmultiplier-chirp and combined again to provide the real and imaginary components of the output.

The discrete Hartley transform can be computed in a similar fashion to the chirp transform for the DFT. Let $W_r(m)$ and $W_i(m)$ denote the real and imaginary part of $W^{\frac{m^2}{2}}$. Thus we have:

$$W_r(m) = \cos(\frac{\pi m^2}{N}) \tag{5.23a}$$

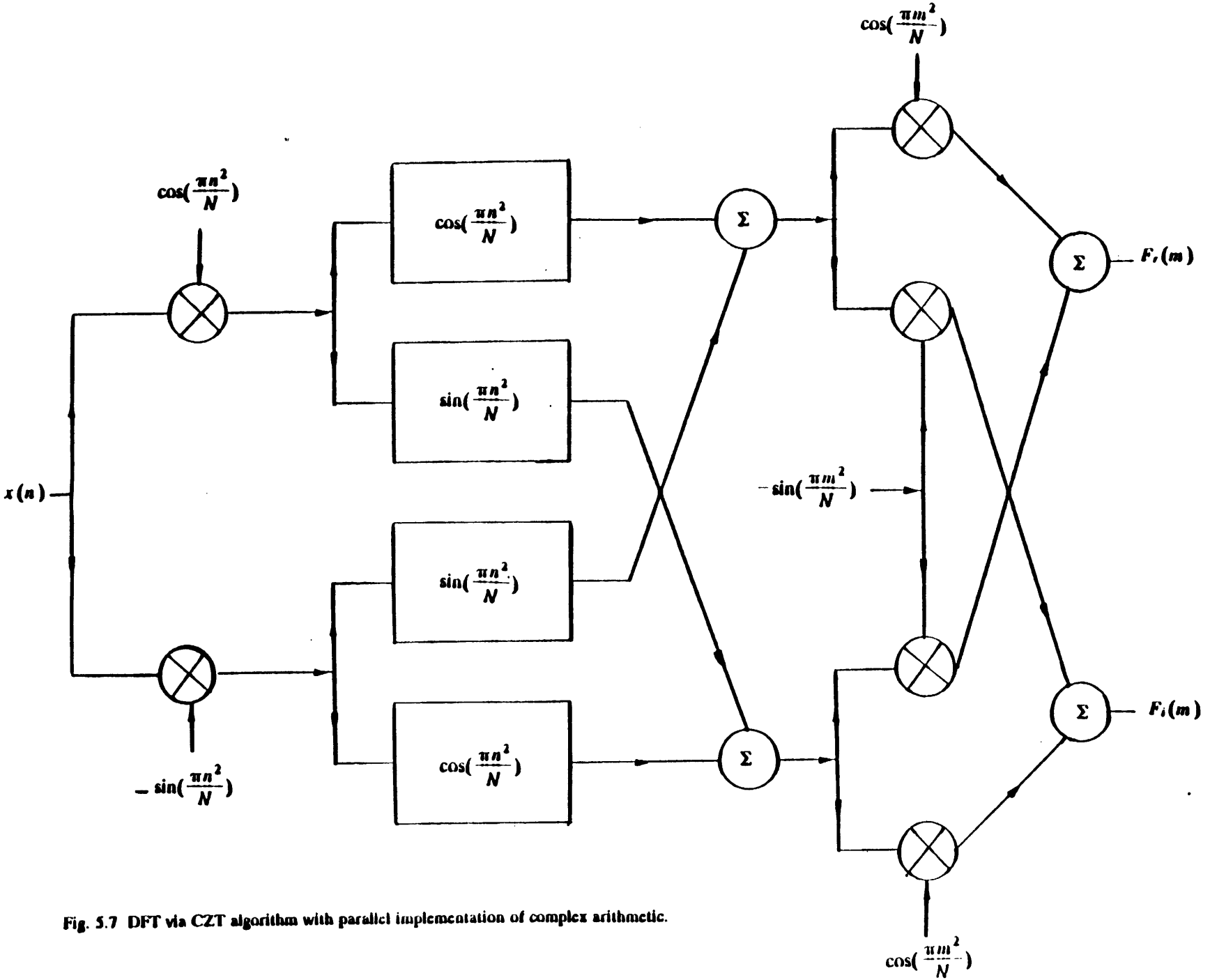


Fig. 5.7 DFT via CZT algorithm with parallel implementation of complex arithmetic.

$$W_i(m) = -\sin(\frac{\pi m^2}{N}) \tag{5.23b}$$

Then equation (5.21) becomes

$$F(k) = [W_r(k)+jW_i(k)]\sum_{n=0}^{N-1} x(n)[W_r(n)+jW_i(n)][W_r(n-k)-jW_i(n-k)] \tag{5.24}$$

Since from chapter 2 we know that the DHT of a sequence is the difference between the real and imaginary part of its DFT, using equation (5.24) the DHT of $x(n)$ denoted by $H(k)$ can be written as:

$$H(k) = H_1(k) + H_2(k) + H_3(k) \tag{5.25}$$

where

$$H_1(k) = cas(-\frac{\pi k^2}{N})[\, x(n)cas(\frac{\pi n^2}{N}) * cas(-\frac{\pi n^2}{N})\,] \tag{5.26a}$$

$$H_2(k) = \cos(\frac{\pi k^2}{N})[\, 2\,x(n)\sin(\frac{\pi n^2}{N}) * \sin(\frac{\pi n^2}{N})] \tag{5.26b}$$

$$H_3(k) = \sin(\frac{\pi k^2}{N})[\, 2\,x(n)\cos(\frac{\pi n^2}{2}) * \cos(\frac{\pi n^2}{N})] \tag{5.26c}$$

where * stands for the convolution. The computation of the above equation is depicted in figure 5.8. DHT can thus be implemented using three convolvers as opposed to four which is needed for the DFT.

This chapter concludes our discussion of the existing and the new DHT algorithms. A list of all the algorithms discussed in chapters 3, 4 and 5 and the key used for each algorithm is shown in table 5.1. In the remainder of this thesis we will investigate the statistical error properties of some of these algorithms theoretically and experimentally.

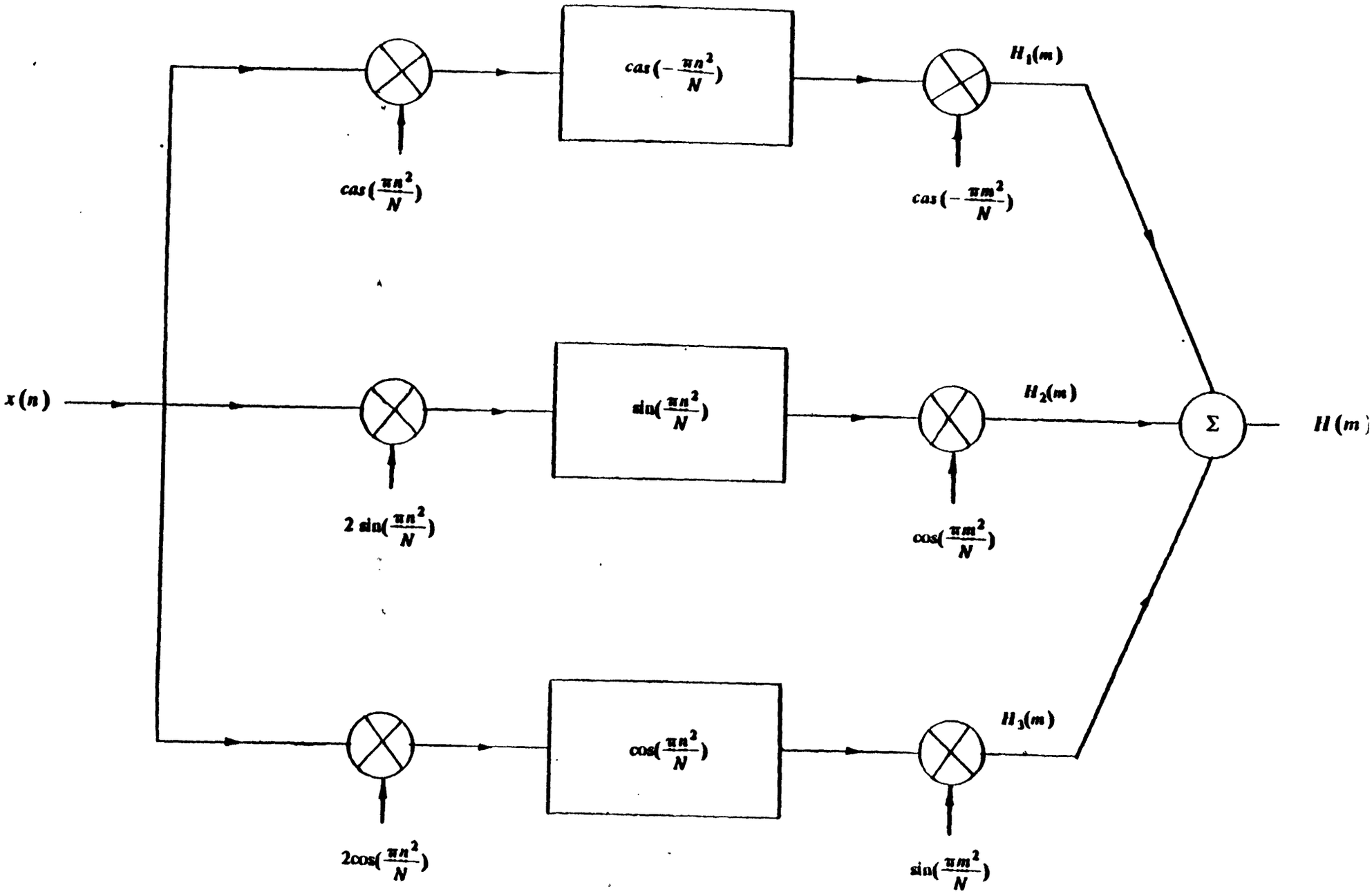


Fig. 5.8 DHT via Chirp Hartley Transform Algorithm

Key	Full name of the algorithm	Section
DT1	Bracewell's original radix 2 decimation in time algorithm	3.1.1
MDT1	Modified version of Bracewell's original radix 2 decimation in time algorithm	3.1.1
R4DT1	Radix 4 decimation in time version of Bracewell's algorithm	3.1.2
SRDT1	Split radix decimation in time version of Bracewell's algorithm	3.1.3
DF1	Radix 2 decimation in frequency of Bracewell's algorithm	3.2
DT3	Wang's algorithm	4
DT2	Radix 2 decimation in time version of the new DHT algorithm	5.1.1
R4DT2	Radix 4 deciamtion in time version of the new DHT algorithm	5.1.2
SRDT2	Split radix decimation in time version of the new DHT algorithm	5.1.3
DF2	Radix 2 decimation in frequency version of the new DHT algorithm	5.1.4
CHT	Chirp Hartley transform algorithm	5.2

Table 5.1 List of the various DHT algorithms

CHAPTER 6: Theoretical Noise Analysis For the DT1, MDT1, DF1 Algorithms

In this chapter, the effects of floating-point and fixed-point roundoff errors in computing the DHT algorithms of chapter 3 will be explored. The error properties of the algorithms presented in chapter 5 will be described in chapter 7. Error is caused when the result of a multiplication or an addition must be rounded to a word length smaller than that needed to represent the exact result. One approach to quantifying the amount of the error would be to derive deterministic bounds on the noise of the transformed sequence. The major drawback of these bounds is that they are very pessimistic in comparison with the results of experiments. A second approach is to model the error sources statistically, with experiments used to test their validity. In this thesis the second approach is taken.

In general effects of quantization on implementation of the DHT algorithms are sources of two kinds of error; errors due to coefficient quantization and errors due to rounding in computation. In this thesis we are only concerned with errors due to rounding in computation. Section 6.1 will discuss the roundoff error models. In section 6.2 we will derive the statistical error properties of the DHT algorithms of chapter 3. In chapter 7, the error characteristics of the algorithms described in chapter 5 will be derived and chapter 8 includes the experimental verification of the error properties derived in chapters 6 and 7.

6.1. Roundoff Error Models

6.1.1. Fixed-Point Error Models

In fixed-point arithmetic, rounding errors occur only when multiplications are performed. Fixed-point additions are basically free of errors provided no overflows occur.

In fixed-point arithmetic the manner in which additions are done is independent of the location of the binary point. For multiplication purposes, with no loss of generality, we can assume that all the numbers are fractions. Thus we will consider fixed-point numbers to be represented as $(b+1)$-bit binary fractions, with the binary point just to the right of the highest order bit. We will also assume that two's complement is used as a way of representing the negative numbers. Thus one bit of the $b+1$ bits used in representing a fixed-point numbers is used to indicate its sign.

When two fixed-point b-bit numbers are multiplied, it is necessary to approximate the $2b$-bit product by a b-bit result. With fractional arithmetic this can be accomplished by truncating or rounding the most significant b bits. The range of values which the resulting error can take on, depends on exactly how the product is reduced from double precision to single precision. If the product is rounded to the nearest single precision fraction then the error denoted by E_R will be in the range [1]:

$$-\frac{1}{2}2^{-b} < E_R \leq \frac{1}{2}2^{-b} \tag{6.1}$$

If the product is truncated, assuming two's complement is used, then the error E_T is always negative and is in the range [1]:

$$-2^{-b} < E_T \leq 0 \tag{6.2}$$

This implies that truncation introduces some bias in the error and therefore it results in larger mean square error than rounding does. Although truncation can usually be implemented more simply and in less time, our analysis is primarily concerned with the roundoff noise.

Let us now define the statistical model we will use for fixed-point rounding errors. Since the quantization width is 2^{-b}, it is plausible to assume the rounding errors to have a probability density function which is uniform in the interval $(-\frac{1}{2}2^{-b}, \frac{1}{2}2^{-b})$ with variance of $\sigma_\epsilon^2 = \frac{1}{12}2^{-2b}$. Furthermore, we will assume that the roundoff error due to multiplications are uncorrelated with each other and with the input. With these assumptions in mind we can associate noise source generators for every multiplier that appears in the flow graph of a specific algorithm and then analyze the effects of the noise sources on the output.

6.1.2. Floating-Point Error Models

In the most common floating-point representation, a positive number F is presented as $F = 2^e M$ where M, the mantissa, is a fraction, such that:

$$\frac{1}{2} \leq M < 1$$

and e, the exponent can be either positive or negative. When M is in the above

range, the floating-point number is said to be normalized. When two floating-point numbers are multiplied, the mantissas are multiplied as fixed-point fractions and the exponents are added together. The product of mantissas is a $2b$-bit number and has to be rounded to b bits. Since the product of the mantissas is between $\frac{1}{4}$ and 1, it might also be necessary to renormalize the product. When adding two floating-point numbers, the mantissa of the smaller number is shifted to the right until their exponents become equal. Then the mantissas are added together. Again the result has to be normalized and rounded. Thus in floating-point arithmetic, unlike fixed-point arithmetic, the results of additions as well as multiplications must be rounded and normalized. Furthermore, the expected magnitude of a floating-point roundoff error depends on the magnitude of the signal. Therefore when dealing with floating-point numbers, we are only concerned about relative errors as opposed to absolute errors. Thus, in order to perform a statistical analysis of noise in DHT algorithms, we must assume a statistical model for the signal, as well as for the roundoff variables. In this thesis, we will assume our signals to be white. It turns out that this assumption not only simplifies the analysis to a great extent, but also gives us good insight about other types of signals such as sinusoids.

We shall consider floating-point numbers with mantissas represented as $(b+1)$-bit binary fractions. Let x be the exact result of an addition or multiplication and $Q(x)$ represent the truncated or rounded value of x. Then

$$Q(x) = x\,(1+\epsilon) \tag{6.3}$$

where ϵ is the relative error. For the case of two's complement rounding we get

[1]:

$$-2^{-b} < \epsilon \leq 2^{-b} \tag{6.4}$$

and for two's complement truncation we get [1]:

$$-2 \cdot 2^{-b} < \epsilon \leq 0 \qquad x > 0 \tag{6.5a}$$

$$0 \leq \epsilon \leq 2 \cdot 2^{-b} \qquad x < 0 \tag{6.5b}$$

Thus for two's complement truncation, the sign of the error is correlated with the sign of the result. Clearly, this is in contradiction with the assumption we made earlier about the independence of signal and the error. In this thesis we will primarily be concerned with rounding two's complement numbers.

Our model for rounding is as follows: By analogy with our fixed-point model, we will assume that ϵ is uniformly distributed in the interval $(-2^{-b}, 2^{-b})$ with variance $\frac{1}{3}2^{-2b}$. Experiments have shown that the variance of error due to multiplications and additions are slightly different from each other and that the distribution for ϵ is not quite uniform [13]. However, the variance of the error has been verified [13] to be proportional to 2^{-2b}. Since in most cases we are interested in finding the variance of error as a function of the transform size, our general approach will be to express variance as a variable in the form represented by

$$\sigma_\epsilon^2 = \alpha 2^{-2b}$$

where α is a constant for a given algorithm which depends on the number of multiplies and additions and the order in which they are performed in that algorithm. The factor α can be determined by matching the theoretical and experimental noise to signal ratio curves. In effect, the value of α represents an empirical average of

σ_ϵ^2 for all the multiplications and additions used in computing the DHT using a specific algorithm. Our general approach in this chapter is to derive the theoretical output noise variance in terms of the parameter σ_ϵ^2. We then use the empirical value for α (from chapter 8) in order to obtain a numerical value for the output noise variance.

We will also assume that the noise sources due to multiplications and additions are uncorrelated with each other. Furthermore, we will assume that when the result of additions or multiplications lies equally between two quantization levels (that is the first extra bit of the mantissa is 1, and all remaining extra bits are zero), a random choice is made as to whether to round up or down. This situation occurs frequently when we add floating numbers of the same order of magnitude. Always rounding up (or down) rather than randomly up or down in this situation would introduce a correlation between roundoff error and signal sign.

6.2. Error Analysis of the DT1, MDT1 and DF1 Algorithms

6.2.1. Roundoff Noise in the DT1 Algorithm

6.2.1.1. Roundoff Noise Analysis of the DT1 Algorithm Using Fixed-Point Arithmetic

In this section we analyze the effects of roundoff errors for output noise-to-signal ratio of the DT1 algorithm described in section 3.1.1. At first we ignore the overflow constraint and derive the output noise variance analytically. Then, dynamic range issues will be considered.

Recall our approach to the analysis of fixed-point roundoff noise introduced in the previous section. We insert additive, signal independent, white noise sources after each multiplier in the signal flow graph of the algorithm. There are two ways of finding the output noise variance for Bracewell's DHT algorithm. The first one results in a closed form expression and provides us with more insight into the algorithm. The second method on the other hand, predicts the output noise variance more accurately.

Our approach in estimating output noise variance using the first method is to identify all the noise sources and their associated variances in the algorithm and investigate the way in which these errors propagate to the output. As mentioned in earlier chapters, Bracewell's algorithm consists of two major parts. In the first part the input signal is bit reversed and in the second part, the algorithm passes through arrays of N real numbers, generating a new array of N real numbers while performing butterfly-type computations. The basic numerical computation at the *nth* stage involves operating on the $(n-1)st$ array in order to generate the *nth* array. The *vth* array is the desired DHT output where $v = \log_2 N$ and N is the transform size. Clearly no error is introduced in the bit reversing section of the algorithm. To quantify the amount of error in the butterfly section of the algorithm, consider a typical butterfly shown in figure 6.1 defined by:

$$X_{n+1}(i) = X_n(i) + \cos(\frac{2\pi p}{N})X_n(j) + \sin(\frac{2\pi p}{N})X_n(l) \qquad (6.6a)$$

$$X_{n+1}(j) = X_n(i) - \cos(\frac{2\pi p}{N})X_n(j) - \sin(\frac{2\pi p}{N})X_n(l) \qquad (6.6b)$$

where p is an integer related to n, i, j and l. Fortunately our analysis is not tied

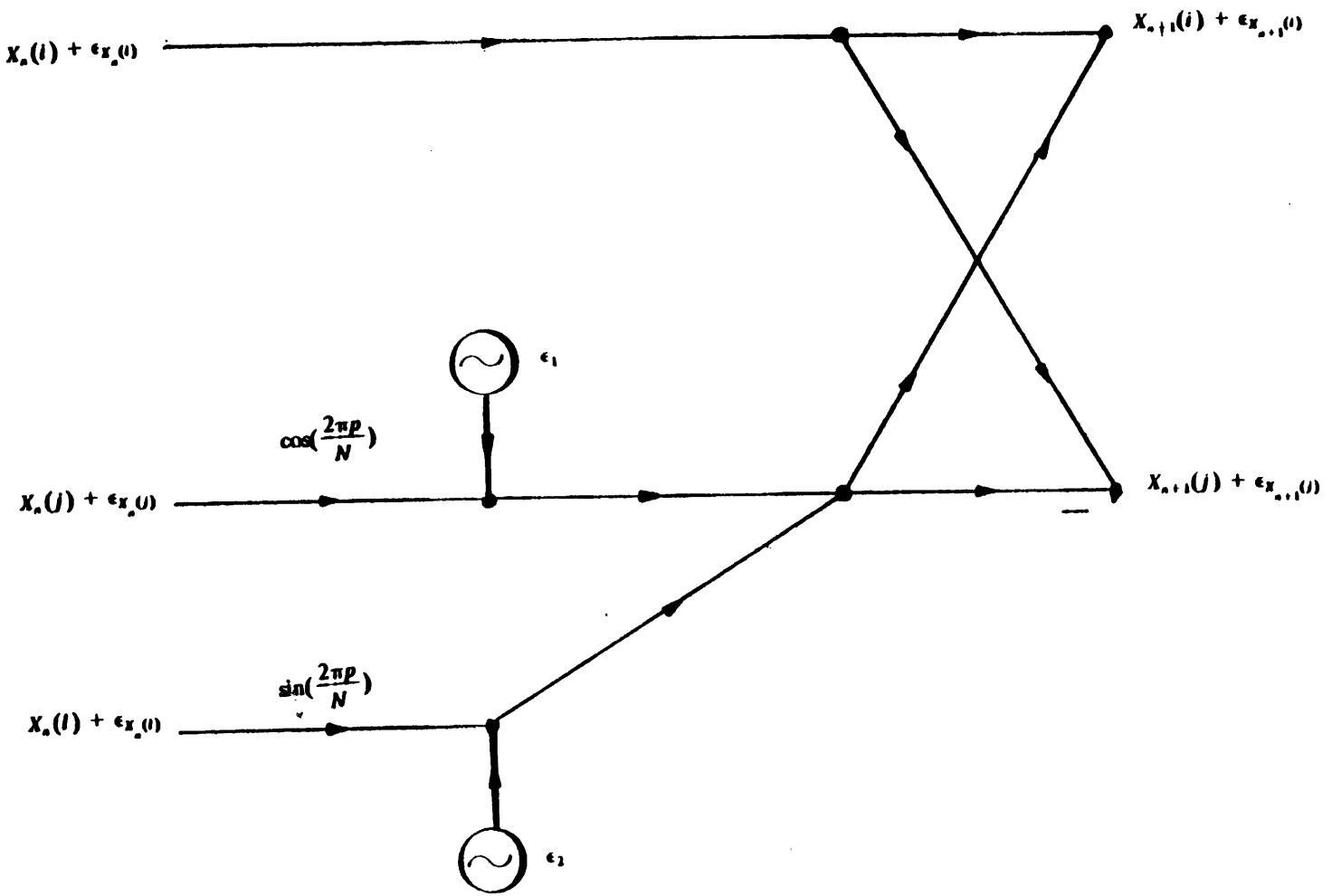


Figure 6.1 Statistical model for fixed-point roundoff noise in a butterfly computation using the DIT algorithm

to the specific way in which p varies. Also the relationships between n, i, j and l are not important for the analysis. At each stage, $\frac{N}{2}$ separate butterfly computations like equation (6.6) are carried out. The error at the output of the butterflies of a given stage, is due to the error inherent in the input of the butterflies and the error due to the computations of that stage. In other words

$$\epsilon_{X_{n+1}(i)} = \epsilon_{X_n(i)} + \epsilon_{X_n(j)}\cos(\frac{2\pi p}{N}) + \epsilon_{X_n(l)}\sin(\frac{2\pi p}{N}) + \epsilon_1 + \epsilon_2 \qquad (6.7a)$$

$$\epsilon_{X_{n+1}(j)} = \epsilon_{X_n(i)} - \epsilon_{X_n(j)}\cos(\frac{2\pi p}{N}) - \epsilon_{X_n(l)}\sin(\frac{2\pi p}{N}) - \epsilon_1 - \epsilon_2 \qquad (6.7b)$$

where $\epsilon_{X_k(m)}$ denotes the error at the *mth* point of the *kth* array of the algorithm. ϵ_i denotes the roundoff error due to fixed-point multiplications. Its variance denoted by σ_ϵ^2 was defined in section 6.1.1. Using induction we can show that the variance of error is the same for all the inputs to a given stage of butterflies. This implies that the variance of $\epsilon_{X_n(i)}$, $\epsilon_{X_n(j)}$ and $\epsilon_{X_n(l)}$ are equal to each other. Using induction, we can also show that the errors in the input of a given butterfly are uncorrelated with each other. In other words we have

$$E[\epsilon_{X_n(i)}\epsilon_{X_n(j)}] = E[\epsilon_{X_n(i)}\epsilon_{X_n(l)}] = E[\epsilon_{X_n(j)}\epsilon_{X_n(l)}] = 0 \qquad (6.8)$$

From equations (6.7) and (6.8) we can conclude that the variance of error at the output of a given butterfly due to the error in its input is simply twice the variance of the error at the input. Since there are two multiplications involved in computing a given butterfly, the variance of the error at the output of a given butterfly due to computations at that butterfly is $2\sigma_\epsilon^2$. (we are taking advantage of the assumption that the noise sources ϵ_1 and ϵ_2 are uncorrelated with each other and with the sig-

nal). Thus the total variance of error at the output of the butterfly is given by

$$Var(\epsilon_{X_{n+1}}) = 2\,Var(\epsilon_{X_n}) + 2\sigma_\epsilon^2 \tag{6.9}$$

where $Var(\epsilon_{X_n})$ denotes the variance of error at the *nth* stage of the algorithm. $Var(\epsilon_{X_\nu})$ is the output noise variance for a sequence of $N = 2^\nu$ points. Since the butterflies at the first and second stage use only additions, they are essentially error free. That is:

$$Var(\epsilon_{X_1}) = Var(\epsilon_{X_2}) = 0 \tag{6.10}$$

For $n > 2$, we apply equation (6.9) in order to find a general expression for the output noise variance of an $N = 2^\nu$ point sequence:

$$Var(\epsilon_{X_\nu}) = (\,2^{\nu-1} - 2\,)\,\sigma_\epsilon^2 \tag{6.11}$$

Using $\sigma^2(N,k)$ to denote the variance of error at the *kth* point of an N-point transform, equation (6.11)can be written as

$$\sigma^2(N,k) = (\frac{N}{2} - 2)\,\sigma_\epsilon^2 \tag{6.12}$$

Equation (6.12) is the basic result we have been seeking for output noise variance of Bracewell algorithm. It says that the variance of the output noise is uniform across the output array (i.e. it is independent of k), and for large values of N the effect of doubling N or adding another stage is to double the output noise variance.

Equation (6.12) can be derived in a more intuitive way; The variance of error in the $(m+1)st$ array due to the roundoff noise of the butterflies in the *mth* stage of the algorithm is $2\sigma_\epsilon^2$ and doubles as we move from one stage to the next. Thus the variance of error at the output of the DHT (*vth* array) due to the errors introduced at the *mth* stage is $2^{\nu-m+1}\sigma_\epsilon^2$. Since the noise source generators of

different stages are assumed to be independent of each other, summing up the effect of the noise due to various stages at the output of the DHT, and taking into account the fact that the first two stages of the algorithm are error free, we get:

$$\sigma^2(N,k) = \sum_{m=3}^{\nu} 2^{\nu-m+1}\sigma_\epsilon^2 = (2^{\nu-1} - 2)\sigma_\epsilon^2 \qquad (6.13)$$

which is the same result as equation (6.12).

In order to simplify the analysis leading to equation (6.12), we have neglected the fact that multiplications by unity can be performed noiselessly. If we assume, in the analysis that these multiplications are noiseless, the output noise variance will no longer be uniform over the entire array (For example the zeroth output point would be noiseless.). The average variance over the output array will be somewhat lower than the result in equation (6.12). In order to take into account the special cases mentioned above, an alternative way of finding the output noise variance is suggested. Recall from chapter 3 that the idea behind Bracewell algorithm is to decompose the problem of finding an N-point DHT into computing two $\frac{N}{2}$-point DHTs. This is demonstrated more clearly in equations (3.1) through (3.5). Pictorially, this can be shown in figure 6.2; $x(n)$ denotes an N-point input sequence which we wish to transform. The DHT of its even and odd points, $X_1(k)$ and $X_2(k)$, are combined in a manner shown in figure 6.2 to form $X(k)$. Let $\epsilon_{X_i(k)}$ denote the the error in $X_i(k)$. Then by inspection of figure 6.2 we have

$$\epsilon_{X(k)} = \epsilon_{X_1(k)} + \epsilon_{X_2(k)}\cos(\frac{2\pi k}{N}) + \epsilon_{X_2(\frac{N}{2}-k)}\sin(\frac{2\pi k}{N}) + \epsilon_1 + \epsilon_2 \qquad (6.14a)$$

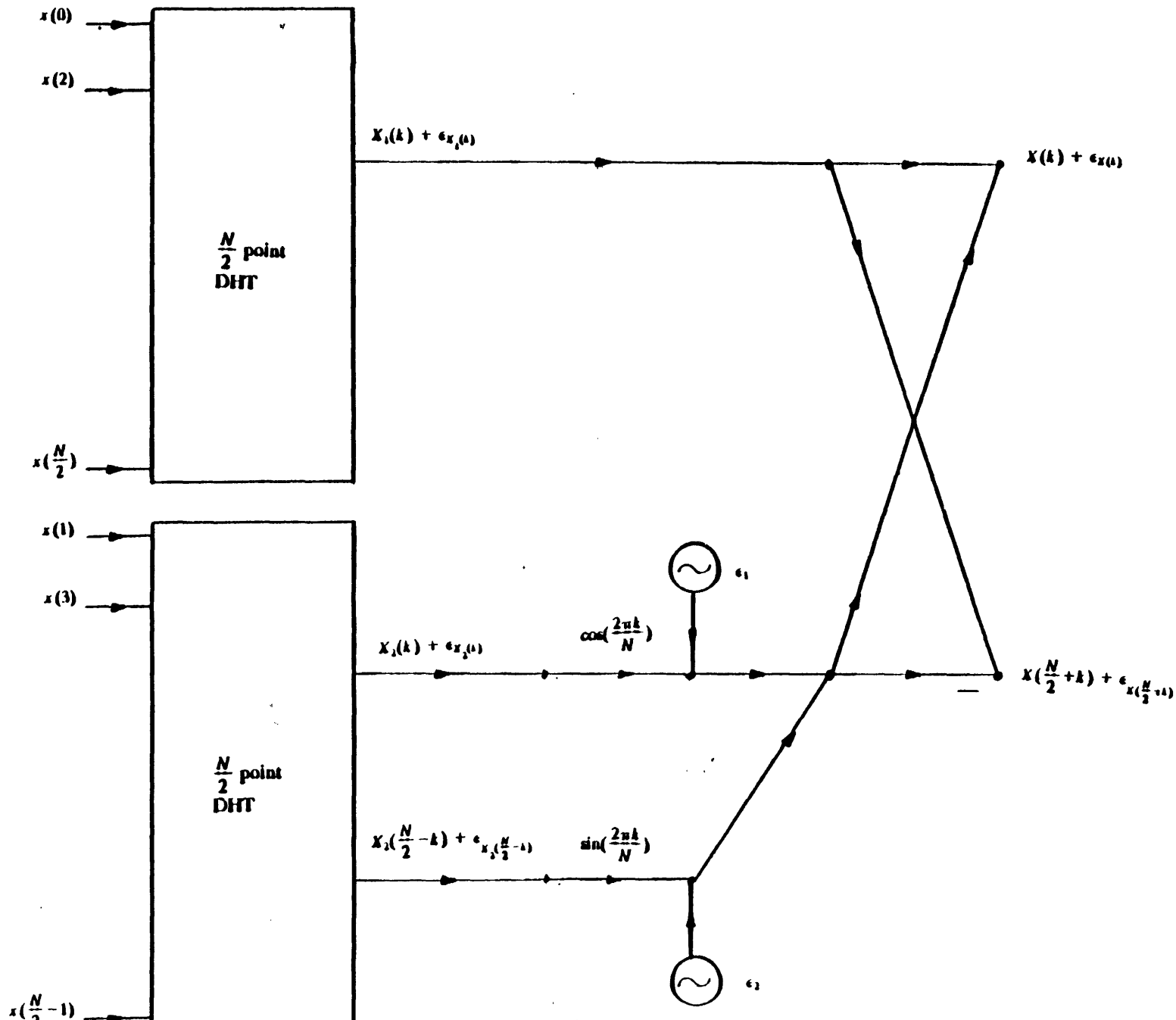


Figure 6.2 Statistical model for fixed-point roundoff noise in a flow graph of the decimation-in-time decomposition of an N-point DHT computation into two N/2-point DHT computations using the

$$\epsilon_{X(k+\frac{N}{2})} = \epsilon_{X_1(k)} - \epsilon_{X_2(k)}\cos(\frac{2\pi k}{N}) - \epsilon_{X_2(\frac{N}{2}-k)}\sin(\frac{2\pi k}{N}) - \epsilon_1 - \epsilon_2 \qquad (6.14b)$$

where ϵ_i denotes the roundoff error due to multiplications by sine and cosine. If we know the output noise variance distribution for $\frac{N}{2}$-point sequences, using equation (6.14) we can find it for N-point sequences. This way we can find the distribution of variance of error for arbitrarily long sequences in a recursive manner. More specifically, if we denote the variance of error at the *kth* point of an N-point sequence by $\sigma^2(N,k)$ by inspection of figure 6.2 we get:

$$\sigma^2(N,k) = \begin{cases} \sigma^2(\frac{N}{2},k) + \cos^2(\frac{2\pi k}{N})\sigma^2(\frac{N}{2},k) + & 0 < k < \frac{N}{2}, k \neq \frac{N}{4} \\ \quad \sin^2(\frac{2\pi k}{N})\sigma^2(\frac{N}{2},\frac{N}{2}-k) + 2\sigma_\epsilon^2 & \\ 2\sigma^2(\frac{N}{2},k) & k = 0, \frac{N}{4} \end{cases} \qquad (6.15a)$$

$$\sigma^2(N,k) = \sigma^2(N,k-\frac{N}{2}) \qquad \frac{N}{2} \leq k < N \qquad (6.15b)$$

Note that for $k = 0, \frac{N}{4}, \frac{N}{2}, \frac{3N}{4}$ the coefficients of the butterflies of figure 6.2 become 0 or 1. By taking care of these special cases, we have incorporated the noiselessness of these multipliers in our theoretical predictions.

Having discussed the output noise variance in detail, we now address the dynamic range considerations. We would like to obtain a formula for output noise to signal ratio, by considering the overflow constraint in conjunction with our noise analysis. We can ensure against overflow in Bracewell algorithm by keeping the input $x(n)$ sufficiently small so that no element of the intermediate arrays or the output array exceeds unity. Our approach here is to find a relationship between the

maximum magnitude of elements of any two succeeding arrays in the algorithm. Let us denote the maximum magnitude of the input sequence by $|X_{in}|_{max}$ and the maximum magnitude of the elements of the *ith* array by $|X_i|_{max}$. Then using equation (6.6) we get:

$$|X_{n+1}|_{max} \leq (1 + \sqrt{2})\,|X_n|_{max}$$

Since there are ν stages of butterfly computations, the maximum magnitude of the output array is given by

$$|X_{out}|_{max} < (1 + \sqrt{2})^{\nu}\,|X_{in}|_{max}$$

Therefore, in order to guarantee that there will be no overflows at any stage of the algorithm (i.e. no intermediate quantity exceeds one) we should choose the input as follows:

$$|X_{in}|_{max} \leq \frac{1}{(1+\sqrt{2})^{\nu}} \tag{6.16}$$

To obtain an explicit expression for output signal variance, we assume $x(n)$ to be white and uniformly distributed in $(-\frac{1}{(1+\sqrt{2})^{\nu}}, +\frac{1}{(1+\sqrt{2})^{\nu}})$. Thus the variance of the input will be $\frac{1}{3\,(1 + \sqrt{2})^{2\nu}}$ and therefore the variance of the output signal denoted by σ^2_{out} can be written as:

$$\sigma^2_{out} = \frac{2^{\nu}}{3\,(1 + \sqrt{2})^{2\nu}} \tag{6.17}$$

The upper bound on the maximum magnitude of the input shown in equation (6.16) can be tightened using a numerical technique which is described in appendix A. The second column of table 6.1 shows this tighter upper bound as a function of the transform size. For large values of N, doubling the transform size scales down

Transform size N	Maximum input $\lvert X_{in} \rvert_{max}$	Variance of output $\frac{N \lvert X_{in} \rvert^2_{max}}{3}$
8	1.0×10^{-1}	2.9×10^{-2}
16	4.5×10^{-2}	1.1×10^{-2}
32	1.9×10^{-2}	3.8×10^{-3}
64	8.0×10^{-3}	1.4×10^{-3}
128	3.4×10^{-3}	4.9×10^{-4}
256	1.4×10^{-3}	1.7×10^{-4}
512	6.0×10^{-4}	6.2×10^{-5}
1024	2.5×10^{-4}	2.2×10^{-5}

Table 6.1 The upper bound on maximum magnitude of the input to ensure against overflow in fixed point implementation of the DT1 algorithm.

the maximum magnitude of the input by a factor of 2.365. As it is mentioned in appendix A, the numerical technique used to generate table 6.1 results in the sufficient but not the necessary condition for the input to prevent overflows. The third column of table 6.1 shows the output signal variance provided the input is white and uniformly distributed in the range specified by the upper bound in the second column.

In figure 6.3 the average output noise to signal ratio using equations (6.12) and (6.15) and table 6.1 as a function of transform size is plotted. Equation (6.15) results in lower average output noise variance as we predicted earlier. The distribution of output noise variance using equation (6.15) among the frequency points of 256-point sequences is shown in figure 6.4. In chapter 8 the experimental results confirming the theoretical predictions of figures 6.3 and 6.4 will be presented.

6.2.1.2. Roundoff Noise Analysis of the DT1 Algorithm Using Floating-Point Arithmetic

The roundoff noise analysis of the DT1 algorithm using floating-point arithmetic is similar to that of fixed-point arithmetic. We shall insert multiplicative, signal independent white noise sources after each multiplier or adder in the signal flow graph. Postulating that all roundoffs are independent of each other, we assume that all the noise sources injected at the various nodes are mutually uncorrelated with the signal. In floating-point arithmetic we deal with relative errors as opposed to absolute errors. We will assume that the input signal is white with variance of

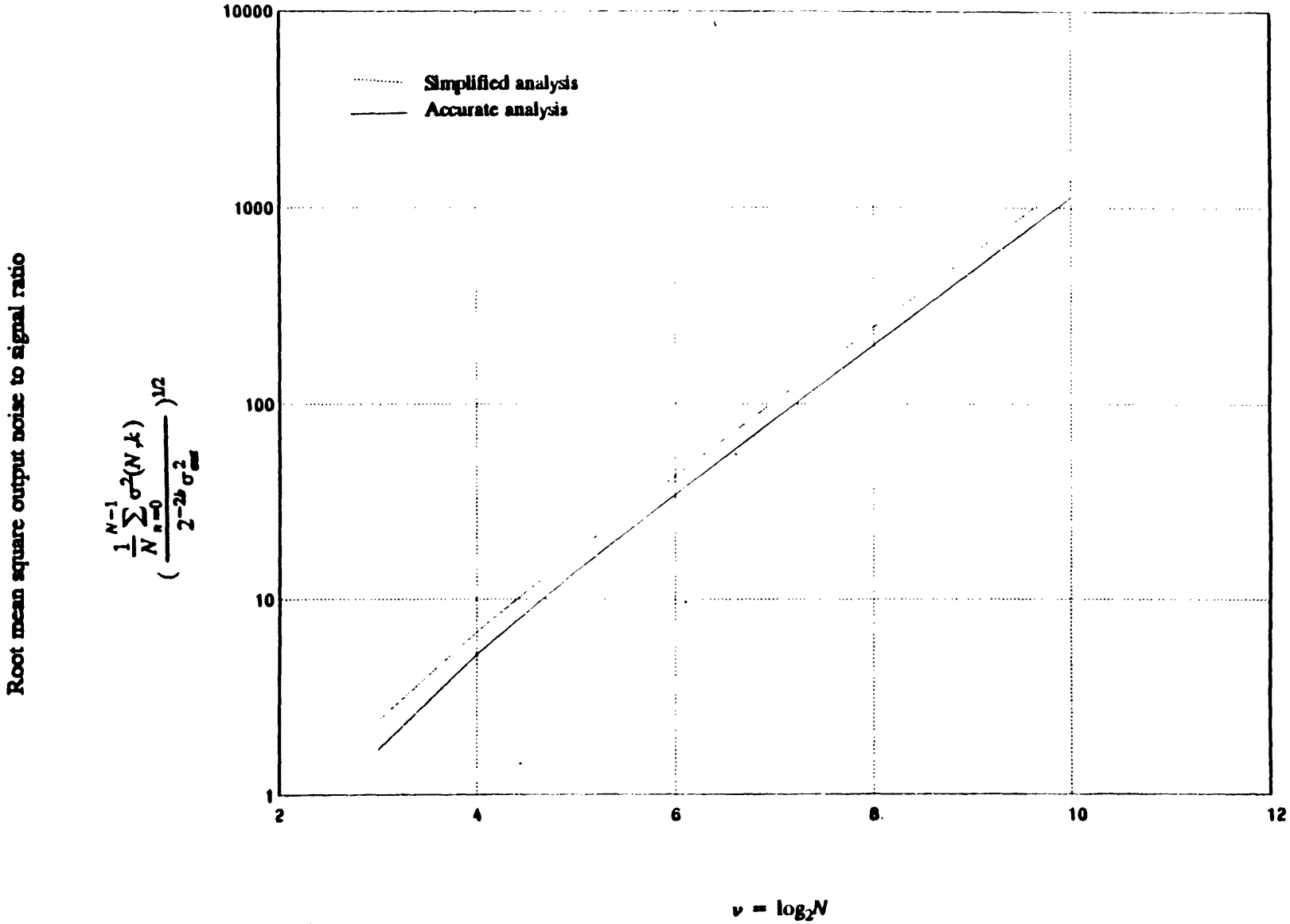


Figure 6.3 Theoretical noise to signal ratio for fixed-point realization of DT1 algorithm

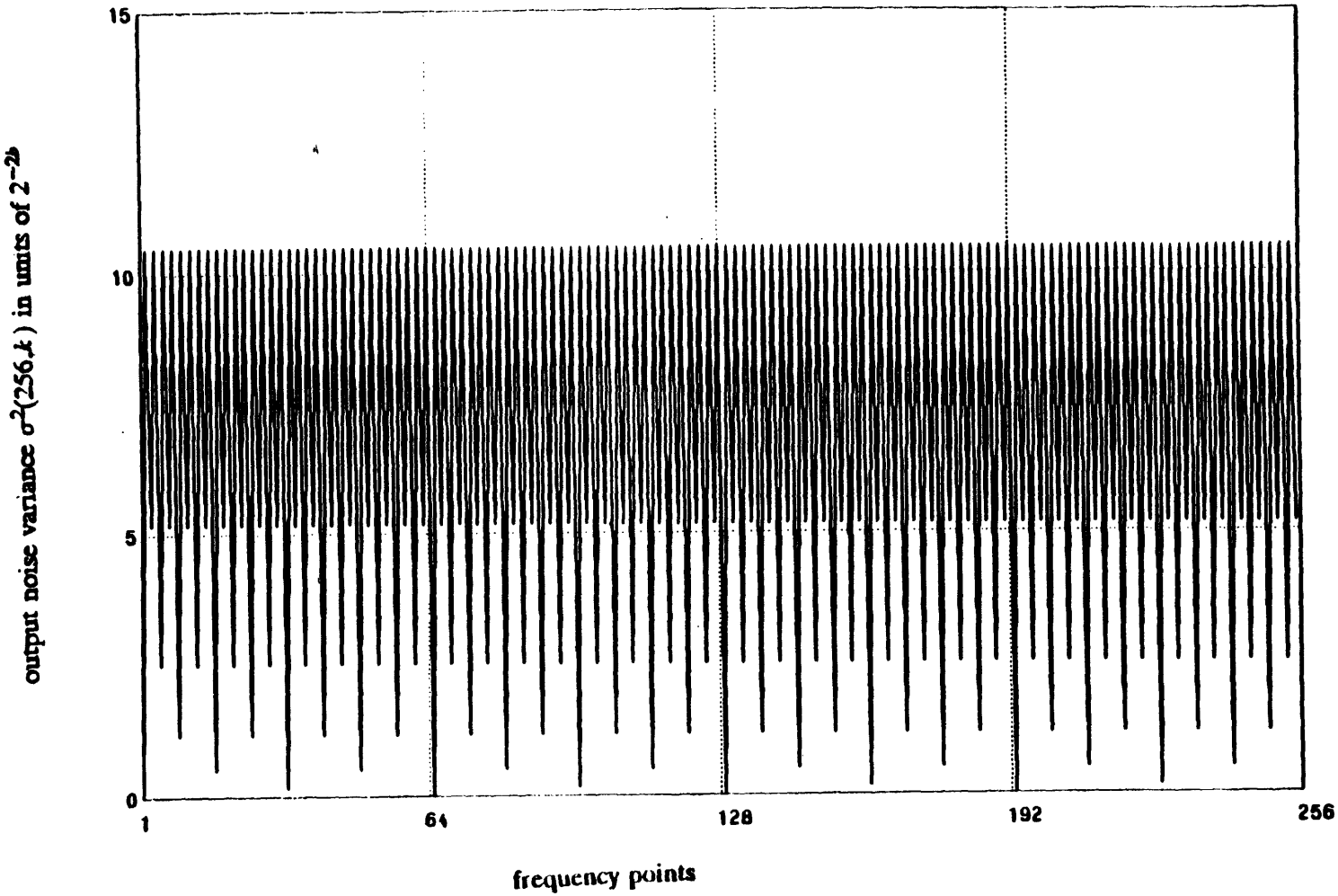


Figure 6.4 Distribution of the output noise variance of 256-point sequences for fixed-point implementation of the DT1 algorithm.

σ_{in}^2. There are again two different methods of carrying out the statistical error analysis. The first one results in a closed form expression for output noise variance and provides us with more insight into the algorithm. The second method on the other hand is more accurate in predicting the error and therefore is in better agreement with the experimental results.

Our approach in estimating output noise variance in the first method is to identify all the noise sources and their associated variances in the algorithm and investigate the way in which these errors propagate to the output. The basic numerical computation at the *mth* stage operates on the $(m-1)st$ array to generate the *mth* array. The *vth* array is the desired DHT output where $v = \log_2 N$. Since our input is assumed to be white with variance σ_{in}^2, the bit reversing operation does not change its whiteness or its variance. Passing through the butterfly section of the algorithm however, changes the variance of the signal. To express this more quantitatively, consider figure 6.5 where a typical butterfly is shown. The outputs of the butterfly are related to its input in the following manner:

$$X_{m+1}(i) = X_m(i) + [X_m(j)\cos(\frac{2\pi p}{N}) + X_m(l)\sin(\frac{2\pi p}{N})] \qquad (6.18a)$$

$$X_{m+1}(j) = X_m(i) - [X_m(j)\cos(\frac{2\pi p}{N}) + X_m(l)\sin(\frac{2\pi p}{N})] \qquad (6.18b)$$

It can be shown (using induction) that the signal at the inputs to the butterflies of all stages of the algorithm retains its whiteness and its variance doubles as we move from one stage to the next. Thus the variance of the signal at the input to the butterflies of the *mth* stage is simply $2^{m-1}\sigma_{in}^2$. That is

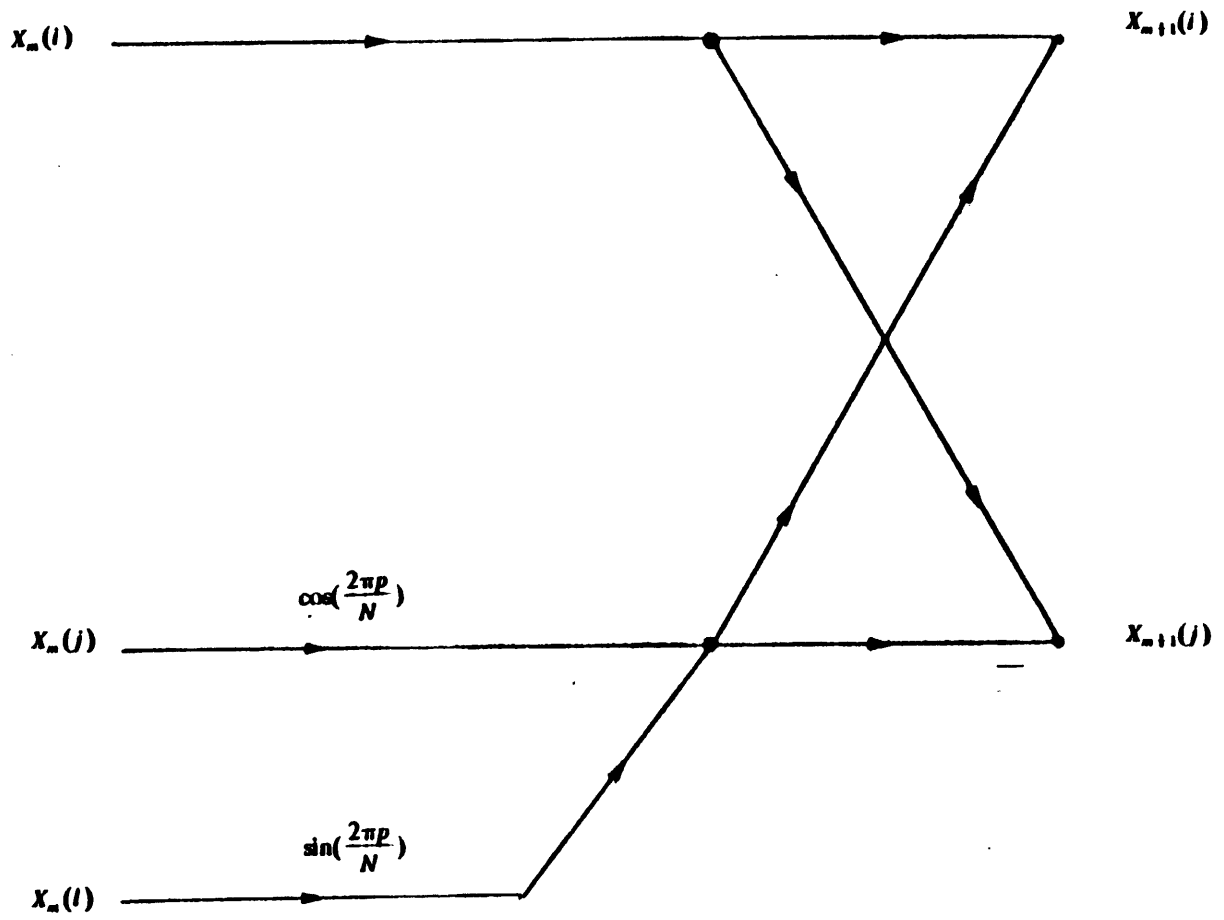


Figure 6.5 Butterfly computation for DT1

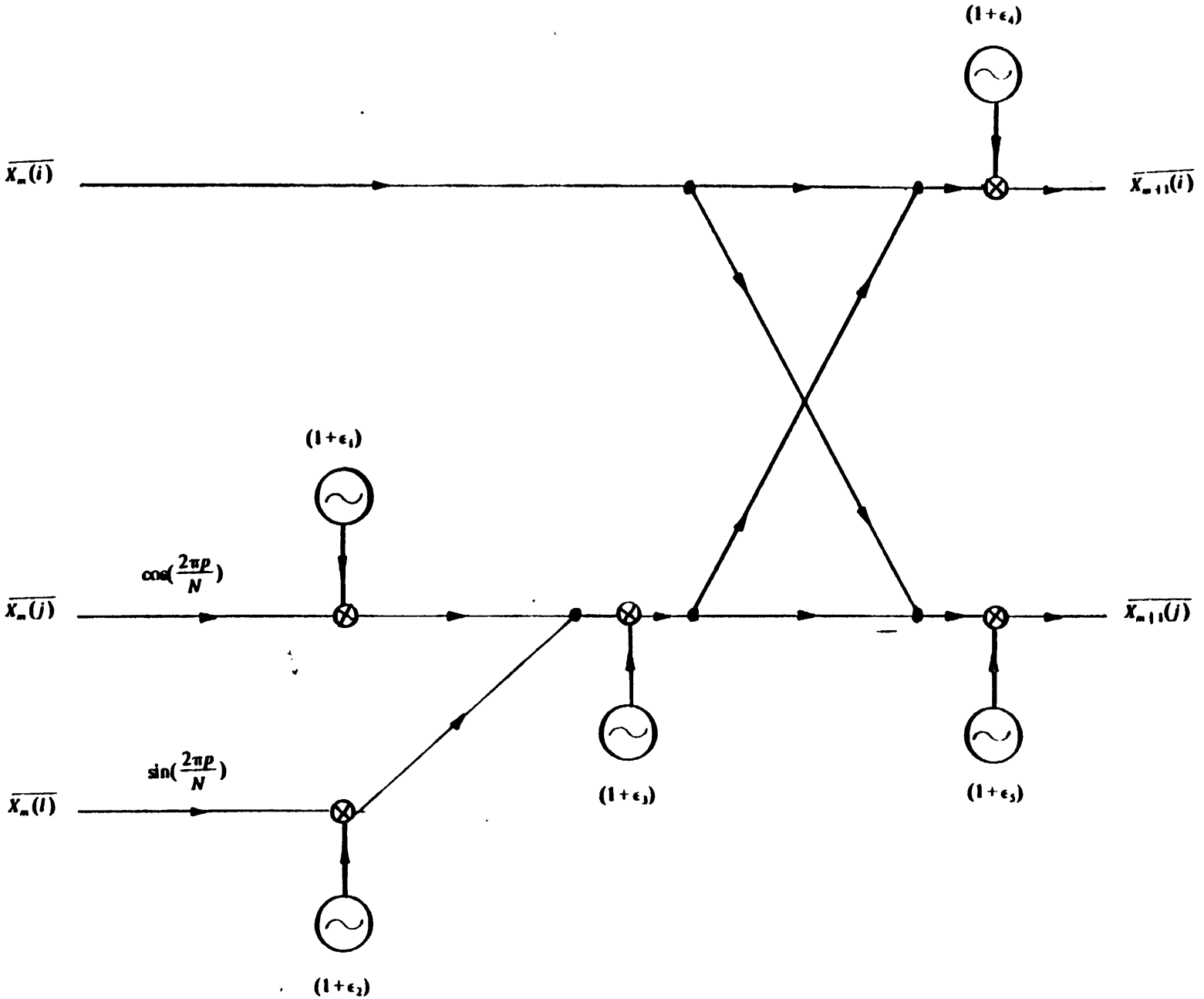


Figure 6.6 Statistical model for floating-point roundoff noise in a butterfly computation using the DT1 algorithm

$$Var[X_m(i)] = Var[X_m(j)] = Var[X_m(l)] = 2^{m-1}\sigma_{in}^2 \qquad (6.19)$$

Figure 6.6 shows the butterfly of figure 6.5 with the white noise source generators indicating the roundoff error due to each addition and multiplication. By inspection of figure 6.6 we have:

$$\overline{X_{m+1}(i)} = \left\{\overline{X_m(i)} + [\,\overline{X_m(j)}\cos(\frac{2\pi p}{N})\,(1+\epsilon_1) + \right. \qquad (6.20a)$$

$$\left. \overline{X_m(l)}\sin(\frac{2\pi p}{N})(1+\epsilon_2)\,]\,(1+\epsilon_3)\right\}(1+\epsilon_4)$$

$$\overline{X_{m+1}(j)} = \left\{\overline{X_m(i)} - [\,\overline{X_m(j)}\cos(\frac{2\pi p}{N})\,(1+\epsilon_1) + \right. \qquad (6.20b)$$

$$\left. \overline{X_m(l)}\sin(\frac{2\pi p}{N})(1+\epsilon_2)\,]\,(1+\epsilon_3)\right\}(1+\epsilon_5)$$

where

$$\overline{X_n(k)} \equiv X_n(k) + \epsilon_{X_n(k)} \qquad (6.21)$$

$\epsilon_{X_n(k)}$ denotes the error accumulated in computing $X_n(k)$ and ϵ_i in equation (6.20) represents the error associated with the addition or multiplication of two floating-point numbers. Variance of ϵ_i is denoted by σ_ϵ^2 and was defined in section 6.1.2. As mentioned earlier we are assuming the injected noise sources ϵ_i to be independent of each other and of the signal. Using induction we can show that the variance of error for all the inputs to the butterflies of any given stage is the same. In terms of the quantities in equation (6.20) this implies that

$$Var[\epsilon_{X_m(i)}] = Var[\epsilon_{X_m(j)}] = Var[\epsilon_{X_m(l)}] \qquad (6.22)$$

Also the error at the three inputs of a given butterfly are uncorrelated with each other. That is

$$E[\epsilon_{X_m(i)}\,\epsilon_{X_m(j)}] = E[\epsilon_{X_m(i)}\,\epsilon_{X_m(l)}] = E[\epsilon_{X_m(j)}\,\epsilon_{X_m(l)}] = 0 \qquad (6.23)$$

Using equations (6.23), (6.22) and (6.20) and the fact that the variance of the signal doubles at every stage of the butterflies we can conclude that

$$Var[\epsilon_{X_{m+1}}] = 2\,Var[\epsilon_{X_m}] + 2\,.\,2^{m+1}\,\sigma_{\epsilon}^2\sigma_{in}^2$$

Since the input to the fist stage of butterflies is error free, we get:

$$Var[\epsilon_{X_0}]=0$$

Solving the above difference equation the output noise variance for a white sequence of length $N = 2^{\nu}$ can be written as:

$$Var[\epsilon_{X_\nu}] = Var[\epsilon_{X_{out}}] = 2\,.\,\nu\,2^{\nu}\sigma_{in}^2\,\sigma_{\epsilon}^2 \qquad (6.24)$$

Using $\sigma^2(N,k)$ to denote the output noise variance at the *kth* point of an N-point white sequence, and σ_{out}^2 to denote the output signal variance, the noise to signal ratio can be written as

$$\frac{\sigma^2(N,k)}{\sigma_{out}^2} = 2\,\nu\,\sigma_{\epsilon}^2 \qquad (6.25)$$

Equation (6.25) is the basic result we have been seeking . It states that the output noise to signal ratio for a white sequence is uniformly distributed among the frequency points and is independent of the variance of the input signal. Furthermore, as we double the transform size from 2^{ν} to $2^{\nu+1}$, the output noise to signal variance is multiplied by $\frac{\nu+1}{\nu}$.

An alternative way of deriving equation (6.24) is as follows: The variance of error in the $(m+1)st$ array due to the roundoff noise of the butterflies in the *mth* stage of the algorithm is $4\sigma_{\epsilon}^2$ times the variance of input to that butterfly (this can easily be verified by considering equation (6.20)). Since the input signal is white

with variance σ_{in}^2, and the signal variance doubles as we move from stage to stage, the variance of the signal at the *mth* array is $2^m \sigma_{in}^2$. Therefore, the variance of the error in the $(m+1)st$ array due to the roundoff noise of the butterflies in the *mth* stage is $4\sigma_\epsilon^2 2^m \sigma_{in}^2$ and doubles as we move from stage to stage. Thus the variance of error at the output (*vth* array) due to the errors introduced at the *mth* stage is $4 \sigma_\epsilon^2 2^m \sigma_{in}^2 2^{\nu-m-1} = 2 N \sigma_\epsilon^2 \sigma_{in}^2$. Since this expression is independent of m, we can conclude that all stages of the algorithm contribute equally to the output noise variance. Since there are ν stages, the total variance of error at the output is simply $\nu N \sigma_\epsilon^2 \sigma_{in}^2$ which is the same result as equation (6.24).

In order to simplify the analysis leading to equation (6.24), we have neglected some details. First, we have associated equal variance noise sources with all multipliers, including when the coefficients are zero or one. If we assume, in the analysis that these multiplications are noiseless, the output noise variance will no longer be uniform over the entire array . The average variance over the output array will be somewhat lower than the result in equation (6.24). In order to take care of the special cases mentioned above an alternative way of finding the output noise variance is suggested. Recall that an N-point DHT can be decomposed into two $\frac{N}{2}$-point DHTs as shown in figure 6.7. Let $\sigma^2(N,k)$ denote the variance of error at the *kth* point of an N-point white sequence. By inspection of figure 6.7, the output noise for the *kth* point of an N-point transform denoted by $\epsilon_{X(k)}$ can be written in terms of $\epsilon_{X_1(k)}$, $\epsilon_{X_2(k)}$ and $\epsilon_{X_2(\frac{N}{2}-k)}$. Since $X_1(k)$ and $X_2(k)$ are $\frac{N}{2}$-point

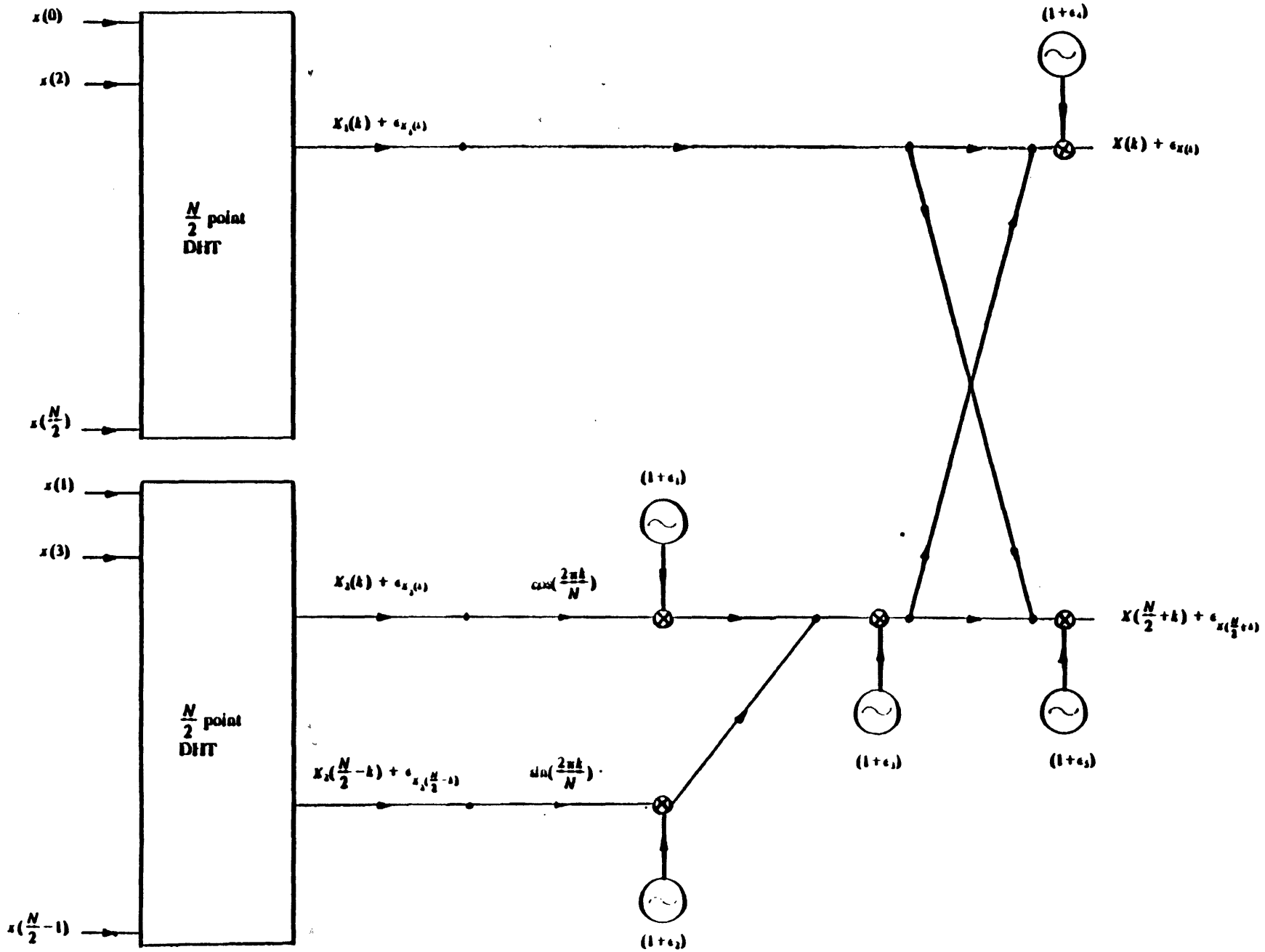


Figure 6.7 Statistical model for floating-point roundoff noise in a flow graph of the decimation in time decomposition of an N-point DHT computation into two N/2 point DHT computations using the DIT algorithm

DHTs of white sequences, the variance of $\epsilon_{X_1(k)}$ and $\epsilon_{X_2(k)}$ is given by $\sigma^2(\frac{N}{2},k)$ and the variance of $\epsilon_{X_2(\frac{N}{2}-k)}$ is given by $\sigma^2(\frac{N}{2},\frac{N}{2}-k)$. Therefore if we know the output noise variance distribution for $\frac{N}{2}$-point white sequences, we can easily find it for N-point white sequences. This way we can find the distribution of variance of error for arbitrarily long sequences in a recursive manner. More specifically, by inspection of figure 6.7 we get:

$$\sigma^2(N,k) = \begin{cases} \sigma^2(\frac{N}{2},k) + \cos^2(\frac{2\pi k}{N})\sigma^2(\frac{N}{2},k) + & 0 < k < \frac{N}{2}, k \neq \frac{N}{4} \\ \qquad \sin^2(\frac{2\pi k}{N})\sigma^2(\frac{N}{2},\frac{N}{2}-k) + 2N\,\sigma_\epsilon^2\sigma_{in}^2 & \\ 2\sigma^2(\frac{N}{2},k)+N\,\sigma_\epsilon^2\sigma_{in}^2 & k=0,\ \frac{N}{4} \end{cases} \tag{6.26a}$$

$$\sigma^2(N,k) = \sigma^2(N,k-\frac{N}{2}) \qquad \frac{N}{2} \le k < N \tag{6.26b}$$

Note that for $k = 0, \frac{N}{4}, \frac{N}{2}, \frac{3N}{4}$ the coefficients of the butterflies of figure 6.7 become 0 or 1. By taking care of these special cases, we have incorporated the noiselessness of these multipliers in our theoretical predictions. The distribution of the output noise variance of 256-point white sequences using equation (6.26) is shown in figure 6.8. Figure 6.9 shows the average output noise to signal ratio using the simplified analysis of equation (6.25) and the more complicated analysis shown in equation (6.26). Equation (6.26) results in lower noise to signal ratio as was predicted earlier. In chapter 8 the experimental results confirming the theoretical predictions of figures 6.8 and 6.9 will be presented.

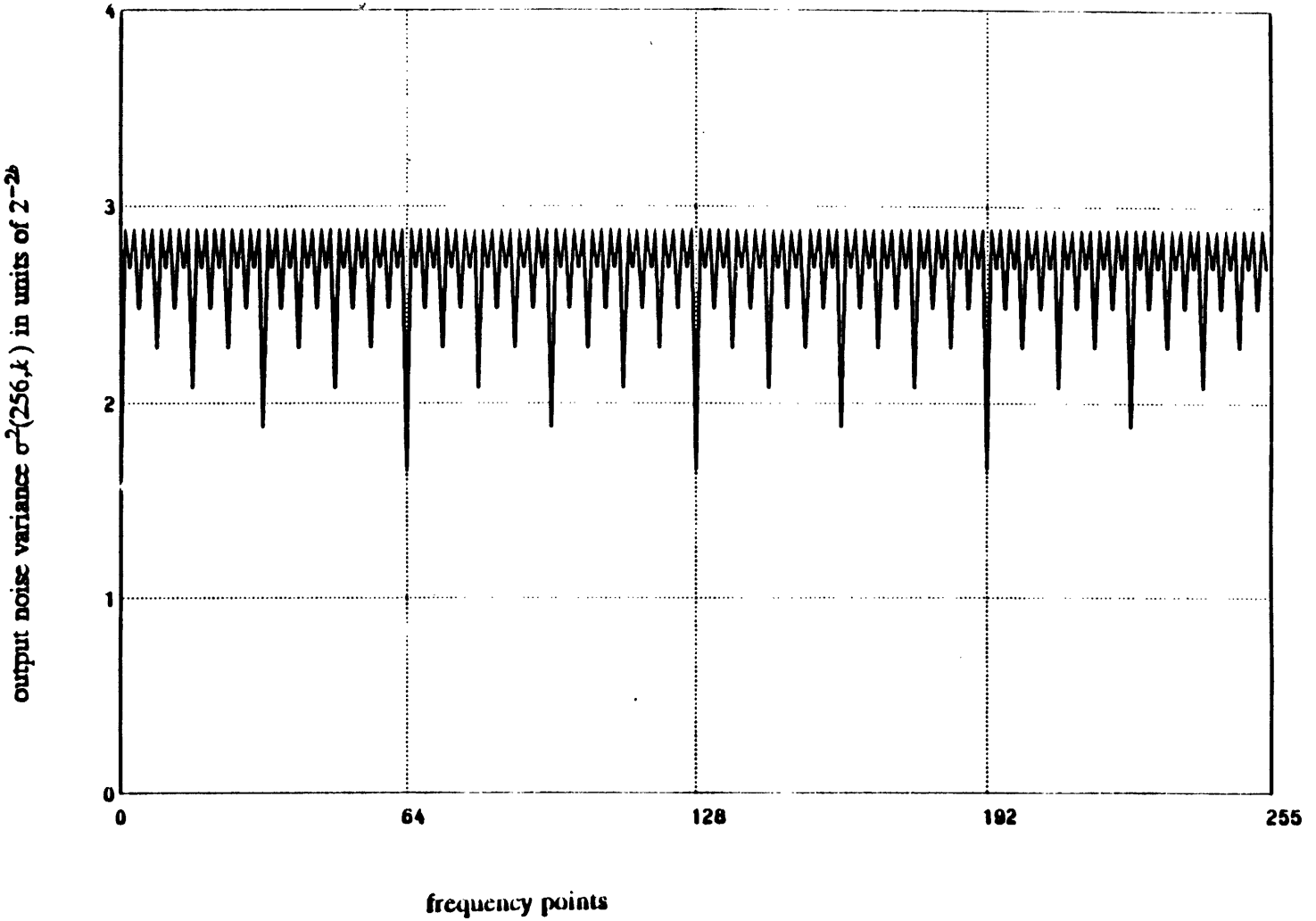


Figure 6.8 Distribution of the output noise variance of 256-point white sequences for floating-point implementation of the DT1 algorithm.

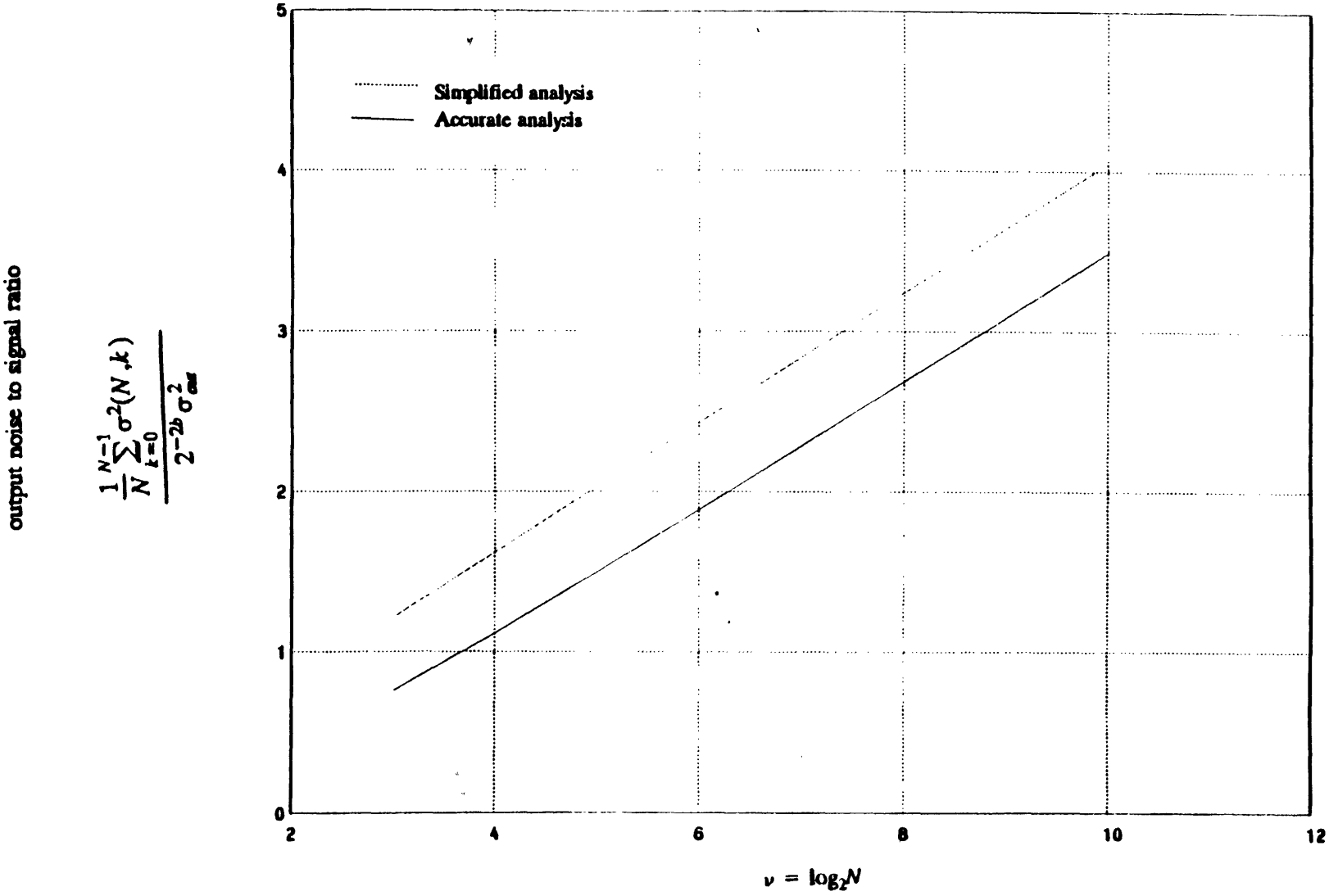


Figure 6.9 Theoretical noise to signal ratio for floating-point realization of DT1

6.2.2. Roundoff Noise in the MDT1 Algorithm

6.2.2.1. Roundoff Noise Analysis of the MDT1 Algorithm Using Fixed-Point Arithmetic

MDT1 algorithm was introduced in section 3.1.1. Although the basic idea behind it is the same as the DT1 algorithm, its error properties are essentially different. Because of the way the butterflies are computed in this algorithm, the error at the inputs of a given butterfly are correlated with each other. Thus the simplified analysis of section 6.2.1 which resulted in a closed form expression can not be applied for this algorithm. In this section the output noise variance of the MDT1 Algorithm using fixed-point arithmetic will be derived. Section 6.1.2.2 will deal with the error analysis of the floating-point implementation of the MDT1 algorithm. Note that the dynamic range considerations related to the MDT1 algorithm are identical to that of the DT1 algorithm.

Recall from chapter 3 that in the MDT1 algorithm, we decompose the problem of finding an N-point DHT into computing two $\frac{N}{2}$-point DHTs. This is shown pictorially in figure 6.10. $x(n)$ is the N-point sequence which we wish to transform. $X_1(k)$ and $X_2(k)$ are the $\frac{N}{2}$-point DHTs of the even and odd points of $x(n)$ respectively. The computed values of $w_1(k)$ and $w_2(k)$ the two intermediate quantities shown in figure 6.10 are given by

$$\bar{w}_1(k) = [\cos(\frac{2\pi k}{N}) + \sin(\frac{2\pi k}{N})][X_2(k) + \epsilon_{X_2(k)}] + \epsilon_1 \qquad (6.27a)$$

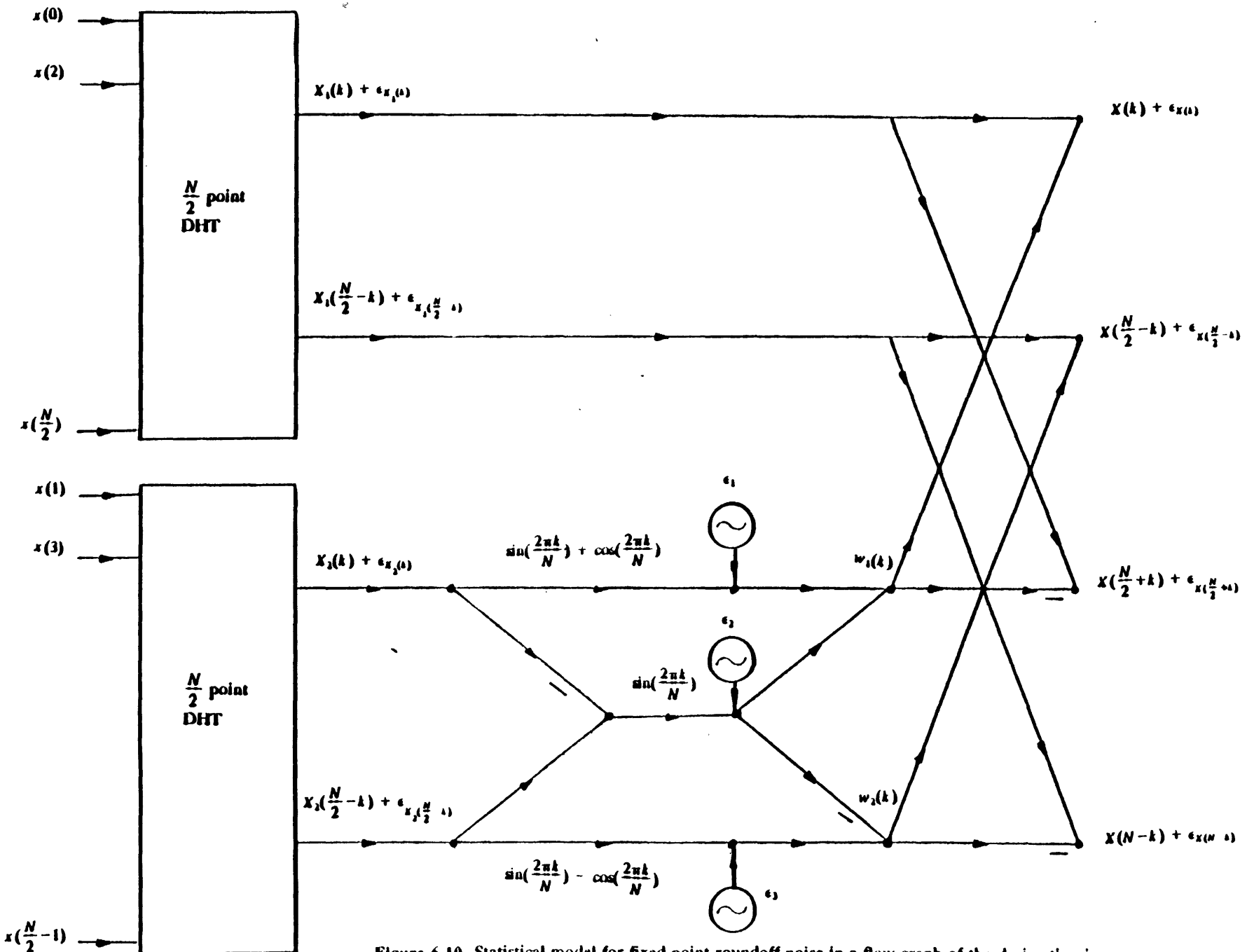


Figure 6.10 Statistical model for fixed-point roundoff noise in a flow graph of the decimation-in-time decomposition of an N-point DHT computation into two N/2-point DHT computations using the MDT1 algorithm

$$+ \sin(\frac{2\pi k}{N})[\ (X_2(\frac{N}{2}-k) + \epsilon_{X_2(\frac{N}{2}-k)}) - (X_2(k)+\epsilon_{X_2(k)})\ ] + \epsilon_2$$

$$\bar{w}_2(k) = [\sin(\frac{2\pi k}{N}) - \cos(\frac{2\pi k}{N})][X_2(\frac{N}{2}-k) + \epsilon_{X_2(\frac{N}{2}-k)}] + \epsilon_3 \qquad (6.27b)$$

$$- \sin(\frac{2\pi k}{N})[\ (X_2(\frac{N}{2}-k) + \epsilon_{X_2(\frac{N}{2}-k)}) - (X_2(k)+\epsilon_{X_2(k)})\ ] - \epsilon_2$$

where ϵ_i denotes the roundoff error due to fixed-point multiplications. Its variance denoted by σ_ϵ^2 was defined in section 6.1.1. $\epsilon_{X_i(k)}$ represents the total amount of error in computing $X_i(k)$. Since $X_i(k)$ is an $\frac{N}{2}$-point DHT of a white sequence, the variance of $\epsilon_{X_i(k)}$ is $\sigma^2(\frac{N}{2},k)$. Using equation (6.27) and figure 6.10 we can conclude that:

$$\sigma^2(N,k) = \begin{cases} 2\,\sigma_\epsilon^2 + \cos^2(\frac{2\pi k}{N})\sigma^2(\frac{N}{2},k) + \sigma^2(\frac{N}{2},k) + & 0<k<\frac{N}{4} \\ \qquad \sin^2(\frac{2\pi k}{N})\sigma^2(\frac{N}{2},\frac{N}{2}-k) + \sin(\frac{4\pi k}{N})cov_\epsilon(\frac{N}{2},k) & \\ 2\sigma^2(\frac{N}{2},k) & k=0,\frac{N}{4} \end{cases} \qquad (6.28a)$$

$$\sigma^2(N,\frac{N}{2}-k) = \begin{cases} 2\sigma_\epsilon^2 + \cos^2(\frac{2\pi k}{N})\sigma^2(\frac{N}{2},\frac{N}{2}-k) + \sigma^2(\frac{N}{2},\frac{N}{2}-k) + & 0<k<\frac{N}{4}, k\neq\frac{N}{8} \\ \qquad \sin^2(\frac{2\pi k}{N})\sigma^2(\frac{N}{2},k) - \sin(\frac{4\pi k}{N})\ cov_\epsilon(\frac{N}{2},k) & \\ \frac{3}{2}\sigma^2(\frac{N}{2},\frac{N}{2}-k) + \sigma_\epsilon^2 + \frac{1}{2}\sigma^2(\frac{N}{2},k) - & k=\frac{N}{8} \\ \qquad cov_\epsilon(\frac{N}{2},k) & \end{cases} \qquad (6.28b)$$

$$\sigma^2(N,k) = \sigma^2(N,k-\frac{N}{2}) \qquad \frac{N}{2} \le k < N \qquad (6.28c)$$

where

$$cov_\epsilon(N,k) = E[\epsilon_{X(k)}\epsilon_{X((N-k))_N}]$$

The special cases of $k=0,\frac{N}{4}$ in the above equation are indicative of the fact that

multiplications by zero and one are essentially error free. Also for $k=\frac{N}{8}$, $(\sin(\frac{2\pi k}{N}) - \cos(\frac{2\pi k}{N}))$ in equation (6.27b) becomes zero and therefore no error occurs in the process of subtracting zero from $\sin(\frac{2\pi k}{N})[X_2(k) - X_2(\frac{N}{2}-k)]$.

$cov_\epsilon(N,k)$ of equation (6.28) which is the covariance of error between the *kth* and $(N-k)th$ points of $X(k)$ can be found recursively in terms of $cov_\epsilon(\frac{N}{2},k)$ and $\sigma^2(\frac{N}{2},k)$. By inspection of figure 6.10 we get

$$cov_\epsilon(N,k) = \begin{cases} \cos(\frac{2\pi k}{N})\sin(\frac{2\pi k}{N})[\,\sigma^2(\frac{N}{2},\frac{N}{2}-k) - \sigma^2(\frac{N}{2},k)\,] + & 0 < k < \frac{N}{4}, k \neq \frac{N}{8} \\ \quad 2\cos^2(\frac{2\pi k}{N})cov_\epsilon(\frac{N}{2},k) + \sigma_\epsilon^2 & \\ 2\sigma^2(\frac{N}{2},k) & k = 0 \\ 0 & k = \frac{N}{4} \end{cases} \tag{6.29a}$$

$$cov_\epsilon(N,k) = cov_\epsilon(N,\tfrac{N}{2}-k) = cov_\epsilon(N,\tfrac{N}{2}+k) = cov_\epsilon(N,N-k) \qquad 0 \le k \le \tfrac{N}{4} \tag{6.29b}$$

From equations (6.28) and (6.29) we can conclude that $\sigma^2(N,k)$, the output noise variance of an N-point DHT and $cov_\epsilon(N,k)$, the covariance of error between its *kth* and $(N-k)th$ point can be computed from $\sigma^2(\frac{N}{2},k)$ and $cov_\epsilon(\frac{N}{2},k)$. The recursion is now complete.

The correlation between the errors at the *kth* and $(N-k)th$ point of the transform is due to the fact that the product $[X_2(k) - X_2(\frac{N}{2}-k)] \sin(\frac{2\pi k}{N})$ shown in figure 6.10 is used to compute both $X_2(k)$ and $X_2(\frac{N}{2}-k)$. This is not the

case in the DT1 algorithm. In fact comparing equations (6.28) and (6.15), we find that except for this covariance term, the noise variance expressions for these two algorithms are fairly similar.

Figure 6.11 shows the distribution of output noise variance among the frequency points for 256-point input sequences. This distribution is slightly different from that of figure 6.4 as we expected. Figure 6.12 shows the signal to noise ratio as a function of transform size for the MDT1 and DT1 algorithms. Although the number of multiplies in the MDT1 is two third of the number of multiplies in the DT1, their noise to signal ratio is almost identical. The experimental results confirming figures 6.11 and 6.12 will be shown in chapter 8.

6.2.2.2. Roundoff Noise Analysis of the MDT1 Using Floating-Point Arithmetic

The roundoff noise analysis using floating-point arithmetic is very similar to that of fixed-point arithmetic. We shall assume that the input signal is white with variance of σ_{in}^2 and insert multiplicative, signal independent white noise sources after each multiplier and adder. Figure 6.13 shows the decomposition used for an N-point sequence $x(n)$ with the noise sources injected for the multipliers and the adders. Referring to figure 6.13, the following expressions for $\bar{w}_1(k)$ and $\bar{w}_2(k)$ can be obtained:

$$\bar{w}_1(k) = (1+\epsilon_4)\left\{ [\cos(\frac{2\pi k}{N}) + \sin(\frac{2\pi k}{N})][X_2(k) + \epsilon_{X_2(k)}](1+\epsilon_1) + \right.$$

$$\left. (1+\epsilon_3)(1+\epsilon_2)\sin(\frac{2\pi k}{N})[(X_2(\frac{N}{2}-k) + \epsilon_{X_2(\frac{N}{2}-k)}) - (X_2(k)+\epsilon_{X_2(k)})] \right\}$$

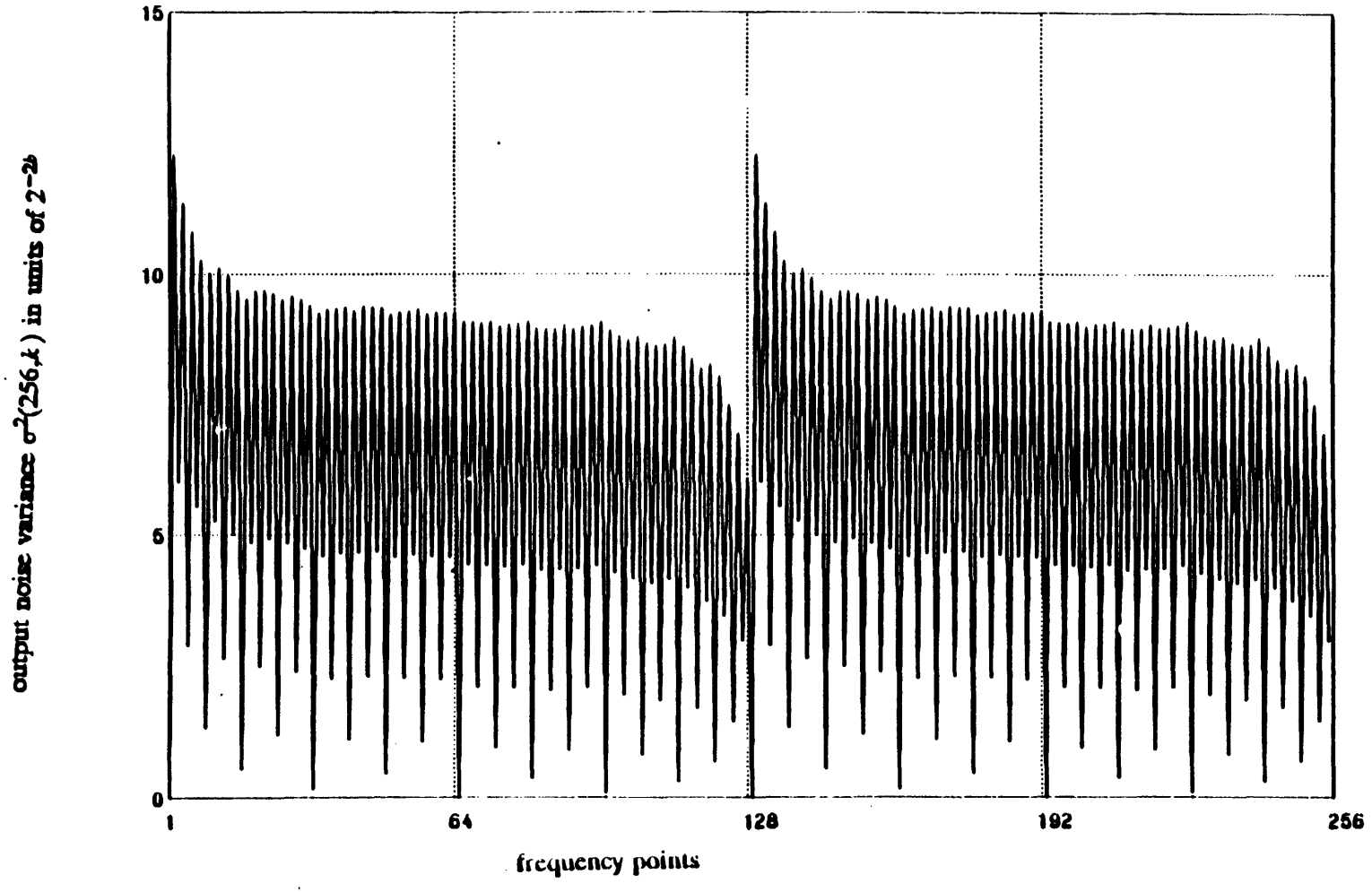


Figure 6.11 Distribution of the output noise variance of 256-point sequences for fixed-point implementation of the MDT1 algorithm.

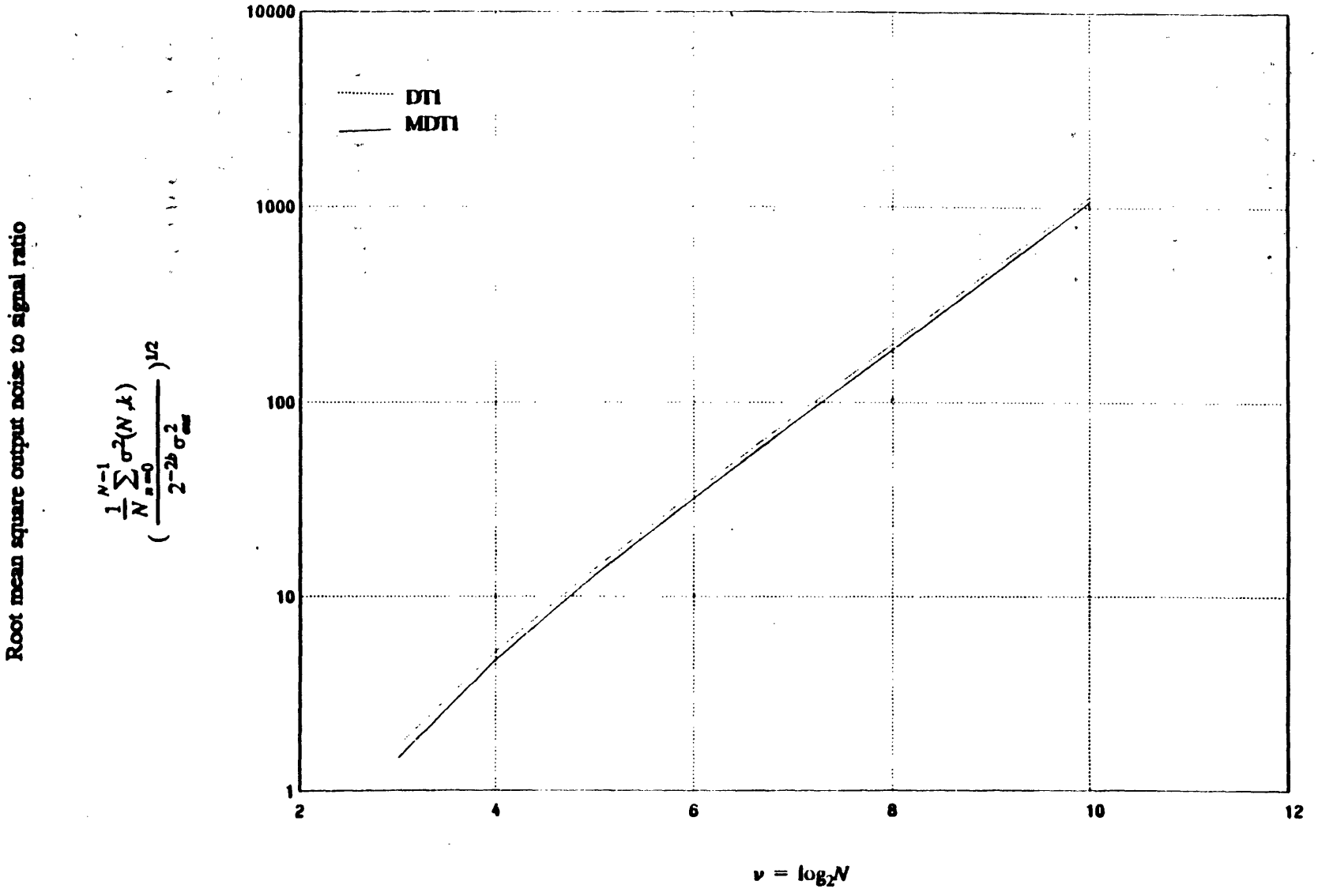


Figure 6.12 Theoretical noise to signal ratio for fixed-point realization of MDT1 and DT1

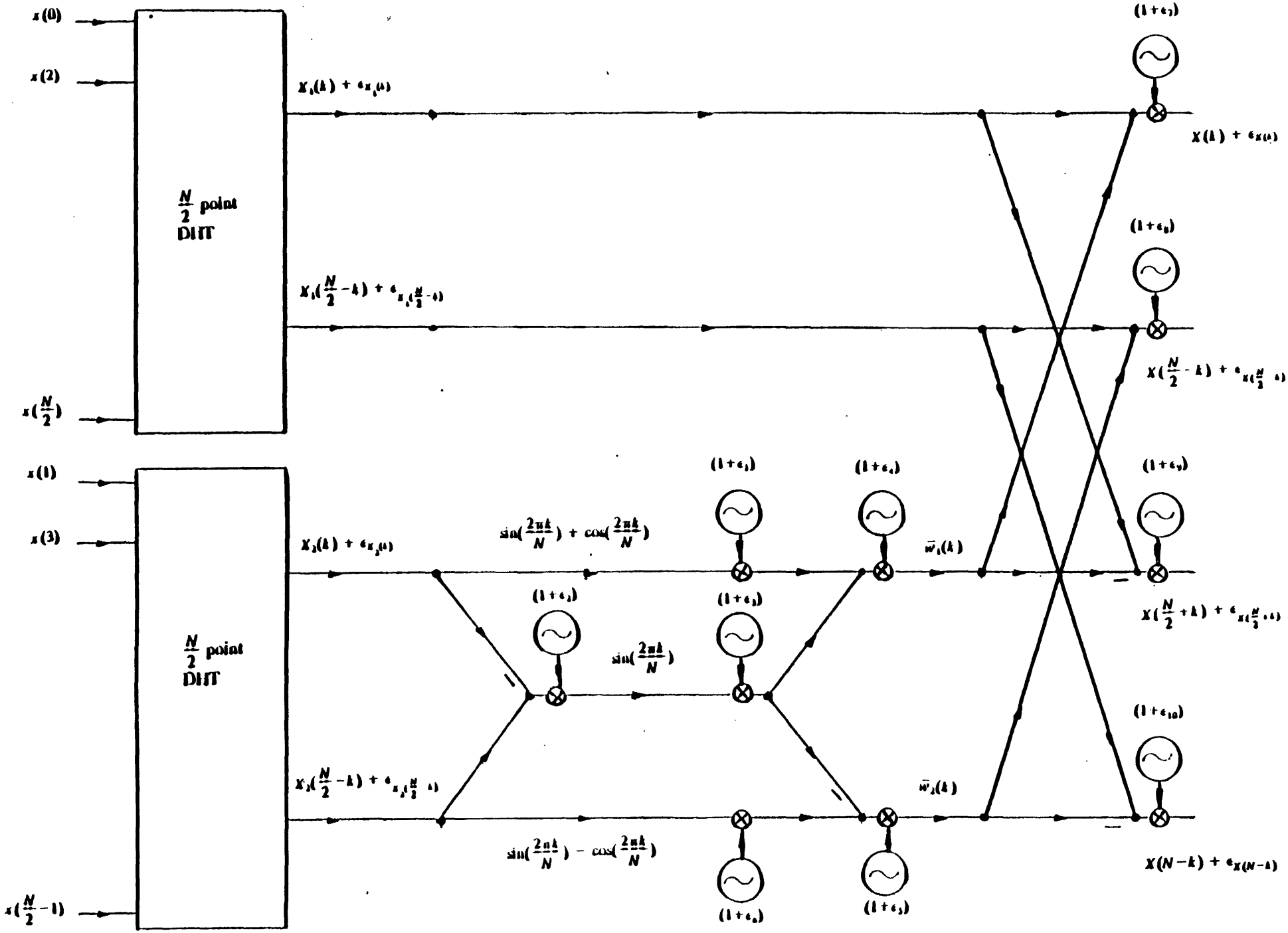


Figure 6.13 Statistical model for floating-point roundoff noise in a flow graph of the decimation-in-time decomposition of an N-point DHT computation into two N/2-point DHT computations using the MDHT algorithm

$$\bar{w}_2(k) = (1+\epsilon_5)\left\{ [\sin(\frac{2\pi k}{N}) - \cos(\frac{2\pi k}{N})][X_2(\frac{N}{2}-k) + \epsilon_{X_2(\frac{N}{2}-k)}](1+\epsilon_6) - \right.$$

$$\left. (1+\epsilon_3)(1+\epsilon_2)\sin(\frac{2\pi k}{N})[(X_2(\frac{N}{2}-k) + \epsilon_{X_2(\frac{N}{2}-k)}) - (X_2(k) + \epsilon_{X_2(k)})] \right\}$$

Since $X_1(k)$ and $X_2(k)$ are $\frac{N}{2}$-point DHTs of white sequences, the variance of $\epsilon_{X_1(k)}$ and $\epsilon_{X_2(k)}$ is simply $\sigma^2(\frac{N}{2},k)$. By inspection of figure 6.13 the variance of error for an N-point sequence is given by

$$\sigma^2(N,k) = \begin{cases} [2N + N\sin(\frac{2\pi k}{N})(\cos(\frac{2\pi k}{N}) + 2\sin(\frac{2\pi k}{N}))]\sigma_\epsilon^2\sigma_{in}^2 + & 0 < k < \frac{N}{4} \\ \quad (1+\cos^2(\frac{2\pi k}{N}))\sigma^2(\frac{N}{2},k) + \sin^2(\frac{2\pi k}{N})\sigma^2(\frac{N}{2},\frac{N}{2}-k) + & \\ \quad \sin(\frac{4\pi k}{N})cov_\epsilon(\frac{N}{2},k) & \\ 2\sigma^2(\frac{N}{2},k) + N\sigma_\epsilon^2\sigma_{in}^2 & k = 0 \\ 2\sigma^2(\frac{N}{2},k) + 2.5N\sigma_\epsilon^2\sigma_{in}^2 & k = \frac{N}{4} \end{cases} \quad (6.30a)$$

$$\sigma^2(N,\frac{N}{2}-k) = \begin{cases} [2N + N\sin(\frac{2\pi k}{N})(2\sin(\frac{2\pi k}{N}) - \cos(\frac{2\pi k}{N}))]\sigma_\epsilon^2\sigma_{in}^2 + & 0 < k < \frac{N}{4},\ k \neq \frac{N}{8} \\ \quad (1+\cos^2(\frac{2\pi k}{N}))\sigma^2(\frac{N}{2},\frac{N}{2}-k) + & \\ \quad \sin^2(\frac{2\pi k}{N})\sigma^2(\frac{N}{2},k) - \sin(\frac{4\pi k}{N})cov_\epsilon(\frac{N}{2},k) & \\ 1.5\sigma^2(\frac{N}{2},\frac{N}{2}-k) + \frac{1}{2}\sigma^2(\frac{N}{2},k) + 2N\sigma_\epsilon^2\sigma_{in}^2 - cov_\epsilon(\frac{N}{2},k) & k = \frac{N}{8} \end{cases} \quad (6.30b)$$

$$\sigma^2(N,k) = \sigma^2(N,k-\frac{N}{2}) \qquad \frac{N}{2} \le k < N \quad (6.30c)$$

where $cov_\epsilon(N,k)$ denotes the covariance of error between the k*th* and $(N-k)$*th* point of the DHT of an N-point white sequence. Note that the special cases of $k=0, \frac{N}{4}, \frac{N}{8}$ in equation (6.30) take care of noiseless multiplications by ones or zeros. Equation (6.30) states that $\sigma^2(N,k)$ can be obtained from $\sigma^2(\frac{N}{2},k)$ and $cov_\epsilon(\frac{N}{2},k)$. In order to complete the recursion we have to be able to obtain $cov_\epsilon(N,k)$

from $\sigma^2(\frac{N}{2},k)$ and $cov_\epsilon(\frac{N}{2},k)$. This can be done by finding the error in $X(k)$ and $X(N-k)$ and evaluating the expected value of their products. The final result is :

$$cov_\epsilon(N,k) = \begin{cases} \cos(\frac{2\pi k}{N}) \sin(\frac{2\pi k}{N}) [\, \sigma^2(\frac{N}{2},\frac{N}{2}-k) - \sigma^2(\frac{N}{2},k)\,] + & 0 < k < \frac{N}{4},\ k \neq \frac{N}{8} \\ \quad 2\cos^2(\frac{2\pi k}{N})\, cov_\epsilon(\frac{N}{2},k) + 2N \sin^2(\frac{2\pi k}{N})\, \sigma_\epsilon^2\sigma_{in}^2 & \\ 2\, cov_\epsilon(\frac{N}{2},k) - N\, \sigma_\epsilon^2\sigma_{in}^2 & k = \frac{N}{4} \end{cases} \tag{6.31a}$$

$$cov_\epsilon(N,k) = cov_\epsilon(N,\frac{N}{2}-k) = cov_\epsilon(N,\frac{N}{2}+k) = cov_\epsilon(N,N-k) \qquad 0 < k \leq \frac{N}{4} \tag{6.31b}$$

Note that $cov_\epsilon(N,k)$ for $k = 0, \frac{N}{2}$ is not shown in equation (6.31) because of the fact that these quantities are not used in computing equation (6.30). Therefore we have shown that the output noise variance of an N-point DHT of a white sequence $\sigma^2(N,k)$, and its covariance of error between its kth and $(N-k)th$ point $cov_\epsilon(N,k)$, can be computed from $\sigma^2(\frac{N}{2},k)$ and $cov_\epsilon(\frac{N}{2},k)$. The recursion is now complete.

The reason for $cov_\epsilon(N,k)$ being non zero is the fact that the error sources ϵ_2 and ϵ_3 of figure 6.13 both contribute to the error at the kth and $(N-k)th$ output points. This is clearly not the case in the DT1 algorithm. In fact, ignoring the $cov_\epsilon(\frac{N}{2},k)$ term, equations (6.26) and (6.30) look somewhat similar.

Figure 6.14 shows the distribution of output noise variance for 256-point white sequences using the MDT1 algorithm. It is clearly different from figure 6.8 where the distribution for the DT1 algorithm is shown. Figure 6.15 shows the noise to signal ratio of the DHTs of white sequences using the MDT1 and the DT1

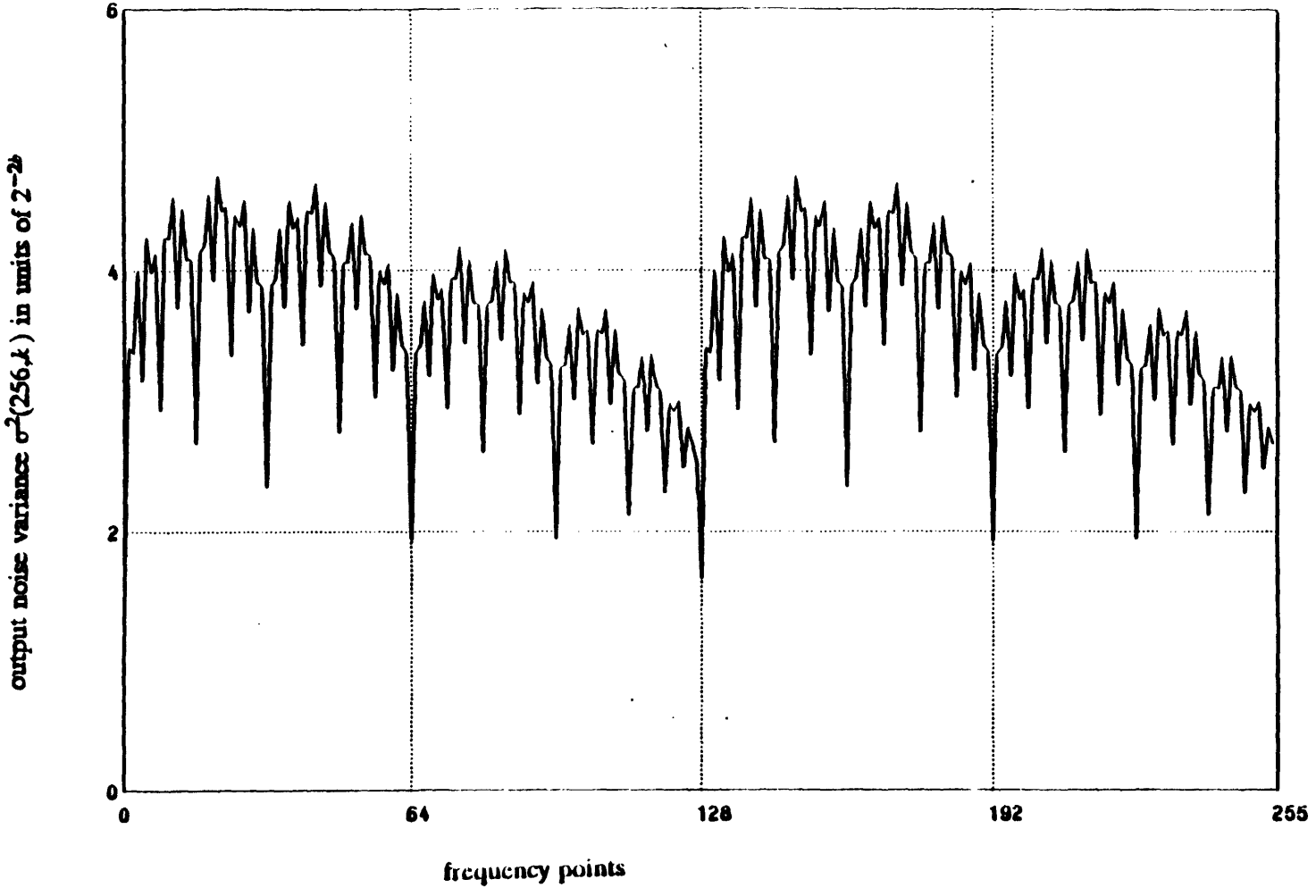


Figure 6.14 Distribution of the output noise variance of 256-point white sequences for floating-point implementation of the MDT1 algorithm.

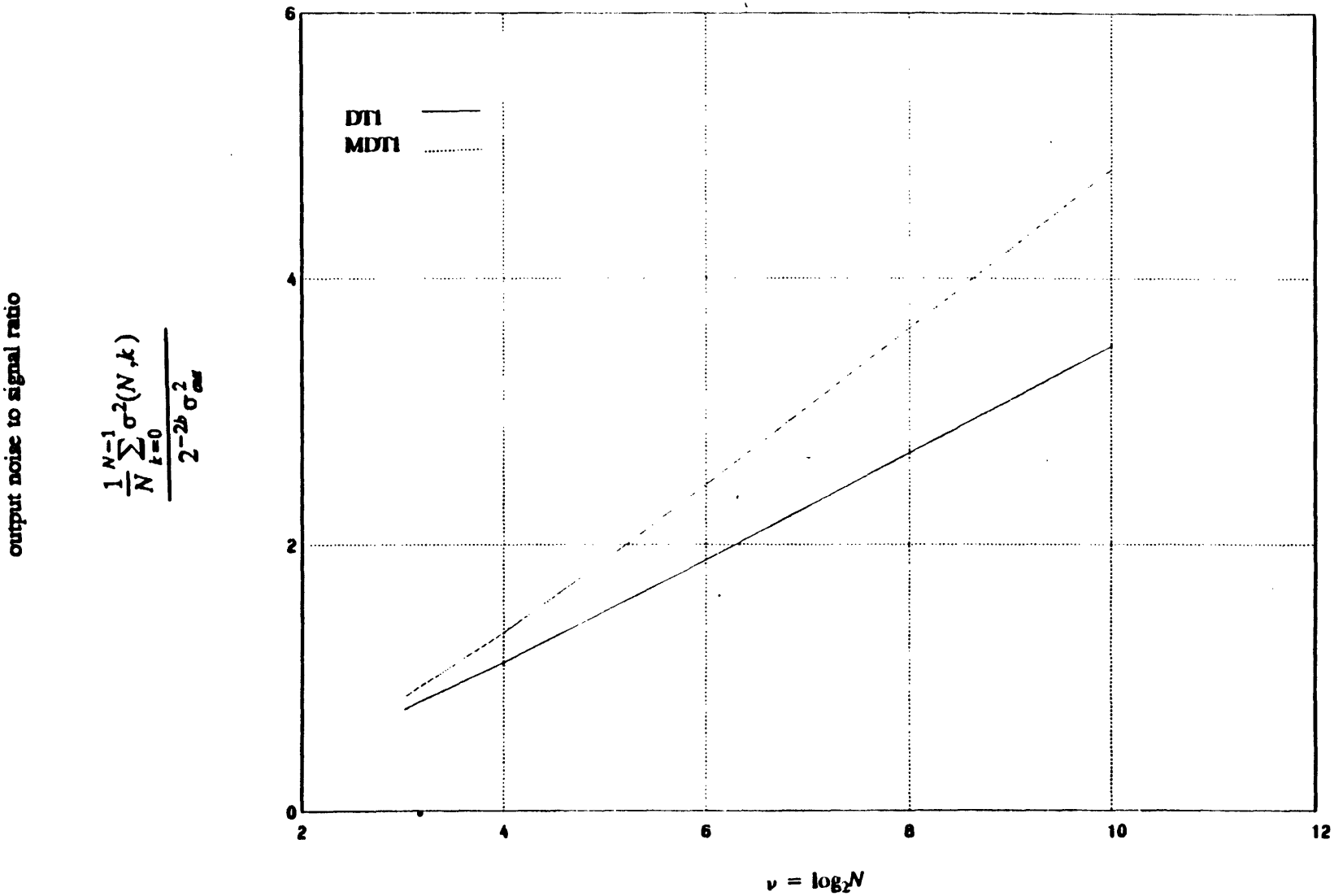


Figure 6.15 Theoretical noise to signal ratio for floating-point realization of MDT1 and DT1

algorithms. Although DT1 requires more multiplications than MDT1, its noise to signal ratio is lower than the MDT1. The experimental results supporting the theoretical predictions in figures 6.14 and 6.15 are included in chapter 8.

6.2.3. Roundoff Noise in the DF1 Algorithm

6.2.3.1. Roundoff Noise Analysis of the DF1 Algorithm Using Fixed-Point Arithmetic

The theoretical analysis for the DF1 algorithm is very similar to that of the DT1 algorithm. We shall insert additive, signal independent white noise after each multiplication in the signal flow graph of the algorithm and find their effects at the output of the transform.

The simplified way of deriving the output noise variance for this algorithm is as follows: The variance of error in the $(m+1)st$ array due to the roundoff noise of the butterflies in the *mth* stage of the algorithm is $2\sigma_\epsilon^2$ and doubles as we move from one stage to the next. Thus the variance of error at the output of the DHT (*vth* array) due to the errors introduced at the *mth* stage is $2^{\nu-m+1}\sigma_\epsilon^2$. Since the noise source generators of different stages are assumed to be independent of each other, summing up the effect of the noise due to various stages at the output of the DHT and taking into account that the last two stages of the algorithm are error free we get:

$$\sigma^2(N,k) = \sum_{m=1}^{\nu-2} 2^{\nu-m+1}\sigma_\epsilon^2 = (2^{\nu+1} - 8)\sigma_\epsilon^2 \qquad (6.32)$$

Equation (6.32) states that the output noise variance is uniformly distributed

among all frequency points and doubles (for large values of N) as the transform size is doubled.

The simplified analyses which have led us to equations (6.32) and (6.13) suggest that the output noise variance for an N-point sequence is proportional to $\frac{N}{2}$ for the decimation-in-time algorithm and $2N$ for the decimation-in-frequency algorithm. This might sound puzzling at first. However, it can be explained intuitively in the following manner; Recall that the butterflies with zero and one coefficients form the first two stages in the decimation-in-time implementation and the last two stages in the decimation-in-frequency implementation. Therefore in the decimation-in-time algorithm the presence or absence of these stages do not affect the output noise variance at all. However, in the decimation-in-frequency implementation, the variance of error due to the first $(\nu-2)$ stages of the algorithm double as they go through each of these two stages. This is the reason why the output noise variance for the DF1 algorithm is predicted to be four times that of the DT1 algorithm.

In order to simplify the analysis leading to equation (6.32) we have neglected some details; We have associated equal variance noise sources with all multipliers, including when the coefficients are zero or one (Note that in the simplified analysis we only took care of these butterflies which appeared in the last two stages. Other stages except the last two ones also have butterflies of this kind). The output noise variance will be less than the result in equation (6.32) if these cases are taken into account. To improve the accuracy of our analysis, an alternative way of

finding the output noise variance is suggested. Recall from chapter 3 that the idea behind the decimation-in-frequency algorithm is to decompose the problem of finding an N-point DHT into computing two $\frac{N}{2}$-point DHTs. This is demonstrated more clearly in equations (3.25) and (3.26). Pictorially, this can be shown in figure 6.16; $x(n)$ denotes the N-point input sequence which we wish to transform. It is decomposed into two $\frac{N}{2}$-point sequences $x_1(n)$ and $x_2(n)$ as shown in figure 6.16. The DHT of $x_1(n)$ denoted by $X_1(k)$ constitutes the even points of $X(k)$ and the DHT of $x_2(n)$ denoted by $X_2(k)$ constitutes the odd points of $X(k)$. Let $\epsilon_{x_i(n)}$ denote the error in $x_i(n)$. By inspection of figure 6.16 we have:

$$\epsilon_{x_1(n)} = 0 \tag{6.33a}$$

$$\epsilon_{x_2(n)} = \begin{cases} \epsilon_{n_1} + \epsilon_{n_2} & 0 < n < \frac{N}{2},\ n \neq \frac{N}{4} \\ 0 & n = 0,\ \frac{N}{4} \end{cases} \tag{6.33b}$$

where ϵ_{n_i} denotes the roundoff error due to multiplications in fixed-point arithmetic. Its variance is denoted by σ_ϵ^2 and was defined in section 6.1.1. The variance of error at the input of $x_2(n)$ is given by

$$Var[\epsilon_{x_2(n)}] = \begin{cases} 2 \cdot \sigma_\epsilon^2 & 0 < n < \frac{N}{2},\ n \neq \frac{N}{4} \\ 0 & n = 0,\ \frac{N}{4} \end{cases}$$

If $\epsilon_{x_2(n)}$ was the only source of error in $X_2(k)$ (i.e. the $\frac{N}{2}$-point DHT was done with infinite precision), the output noise variance at $X_2(k)$ due to the error in the input would have been given by

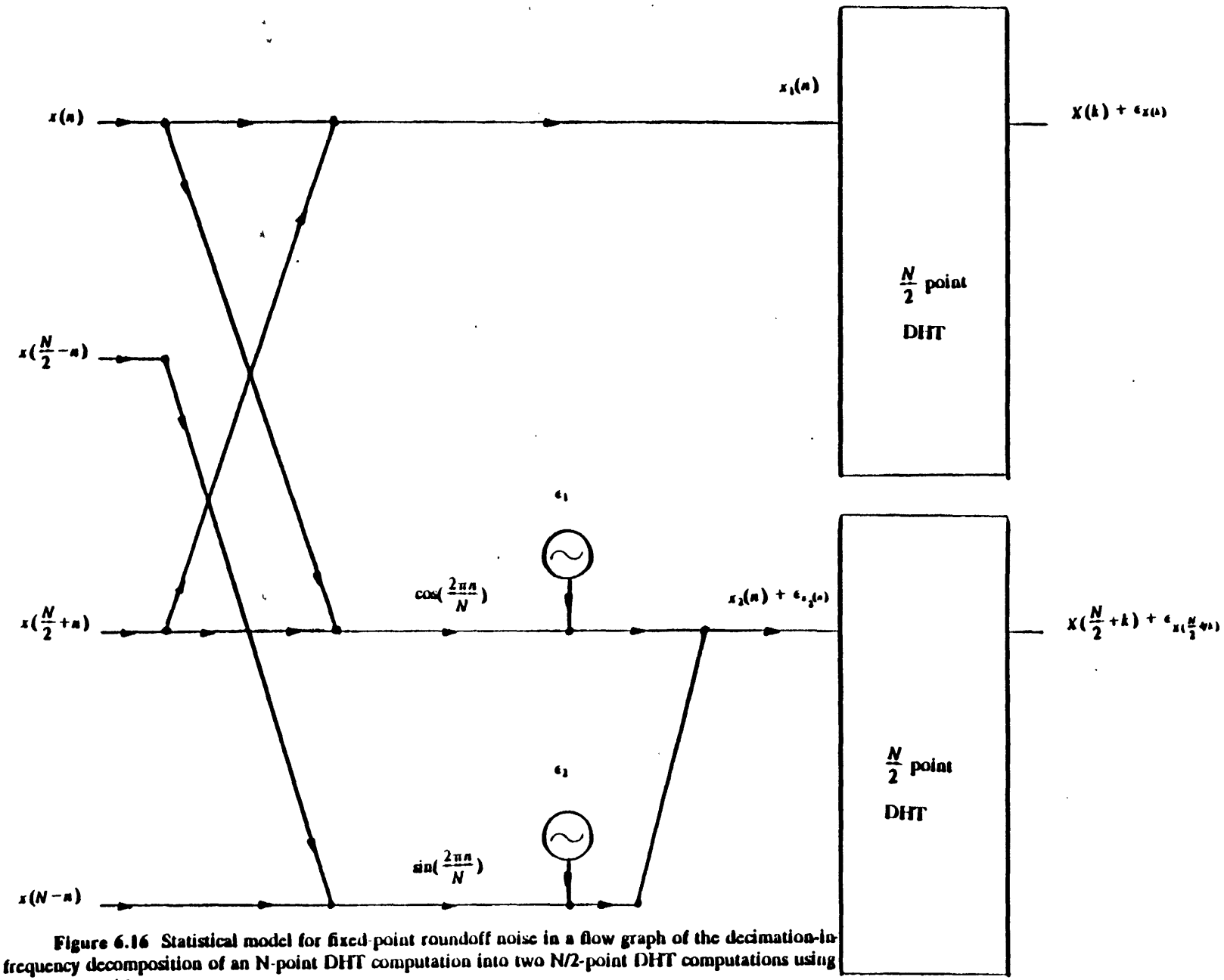


Figure 6.16 Statistical model for fixed-point roundoff noise in a flow graph of the decimation-in-frequency decomposition of an N-point DHT computation into two N/2-point DHT computations using the DFI algorithm

$$Var'[\epsilon_{X_2(k)}] = \sum_{n=0}^{\frac{N}{2}-1} Var[\epsilon_{x_2(n)}] cas^2(\frac{2\pi kn}{N/2}) = (N-4)\sigma_\epsilon^2 \qquad (6.34)$$

The error in $X_2(k)$ can be considered to be due to error in $x_2(n)$ and due to the errors introduced in the $\frac{N}{2}$-point DHT computation of $x_2(n)$. The variance of error due to the noise in $x_2(n)$ is shown in equation (6.34) and the variance of error due to computations in the $\frac{N}{2}$-point DHT is simply $\sigma^2(\frac{N}{2},k)$. Since these two errors are independent of each other, in order to find the variance of $\epsilon_{X_2(k)}$ we can simply add these two variances. That is

$$Var[\epsilon_{X_2(k)}] = (N-4)\sigma_\epsilon^2 + \sigma^2(\frac{N}{2},k) \qquad (6.35a)$$

Since no error has been introduced in computing $x_1(n)$, the variance of $\epsilon_{X_1(k)}$ is only due to the $\frac{N}{2}$-point DHT computation and therefore we get:

$$Var[\epsilon_{X_1(k)}] = \sigma^2(\frac{N}{2},k) \qquad (6.35b)$$

At the output of the algorithm we have:

$$\epsilon_{X(2k)} = \epsilon_{X_1(k)} \qquad (6.36a)$$

$$\epsilon_{X(2k+1)} = \epsilon_{X_2(k)} \qquad (6.36b)$$

Combining equations (6.35) and (6.36) we get:

$$\sigma^2(N,k) = \begin{cases} \sigma^2(\frac{N}{2},k) & k \text{ even} \\ \sigma^2(\frac{N}{2},k) + (N-4)\sigma_\epsilon^2 & k \text{ odd} \end{cases} \qquad (6.37)$$

Figure 6.17 shows the distribution of the variance of error for 256-point sequences using equation (6.37).

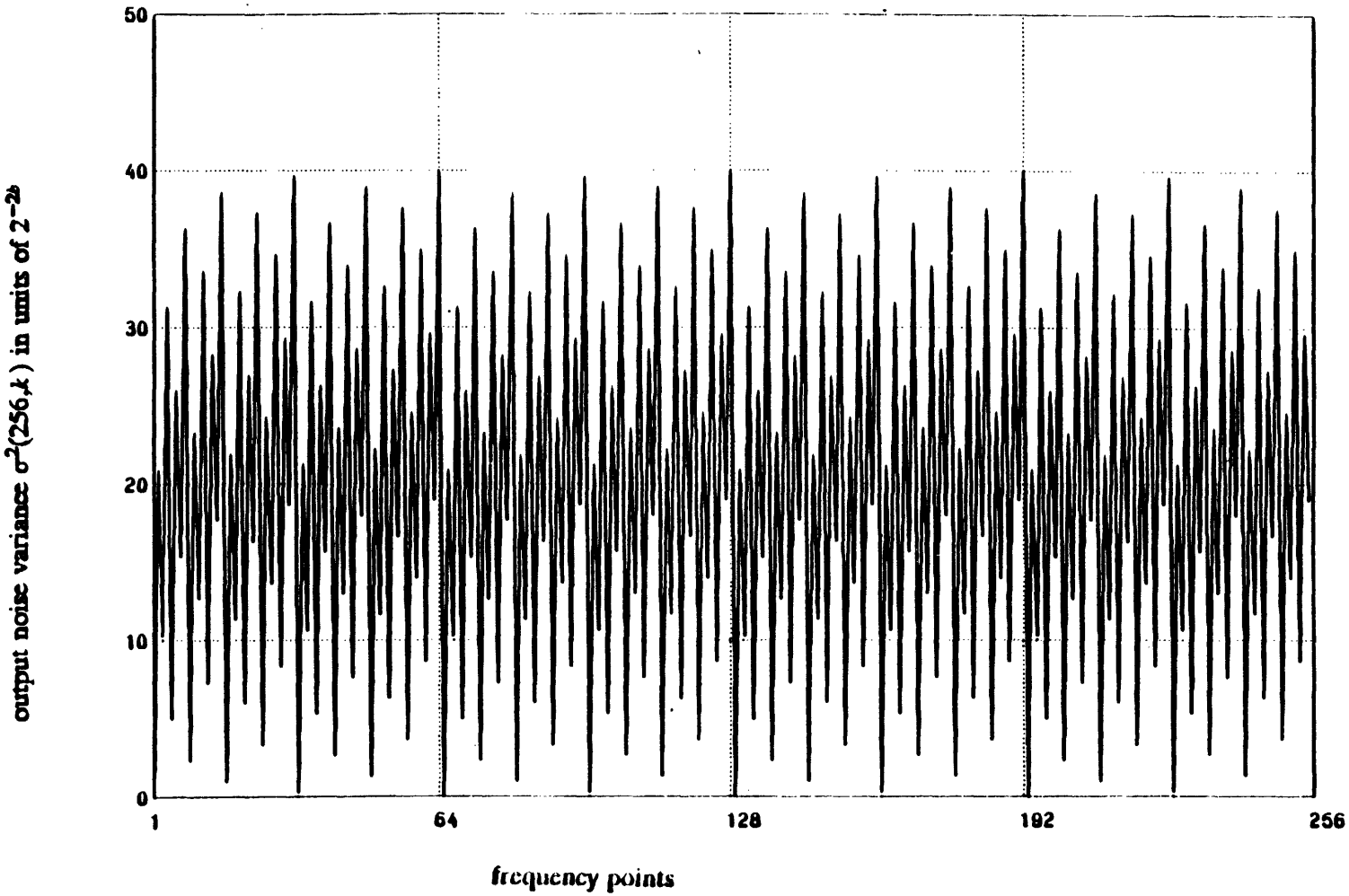


Figure 6.17 Distribution of the output noise variance of 256-point sequences for fixed-point implementation of the DF1 algorithm.

If we average $\sigma^2(N,k)$ of equation (6.37) over the frequency points, we can find a closed form expression for the average output noise variance of an $N=2^\nu$ sequence by solving the following difference equation

$$< \sigma^2(2^\nu,k) > = < \sigma^2(2^{\nu-1},k) > + \frac{1}{2}(2^\nu - 4) \qquad (6.38)$$

where $<.>$ is used to show averaging over frequency points. Since 2 and 4-point DHTs are essentially error free, we get:

$$< \sigma^2(2,k) > = < \sigma^2(4,k) > = 0$$

Solving equation (6.38) for $\nu > 2$, we obtain the following closed form expression for the mean noise variance:

$$< \sigma^2(2^\nu,k) > = (2^\nu - 2\,\nu)\sigma_e^2 \qquad (6.39)$$

Therefore the more accurate analysis which led us to equation (6.39) suggests that the output noise variance of an N-point sequence is proportional to N as opposed to $2N$ which was derived in our simplified analysis.

Figure 6.18 shows the average output noise variance of the decimation-in-time and frequency algorithms using equations (6.15) and (6.39). For large values of N the output noise variance for the decimation-in-time algorithm is double the corresponding quantity for the decimation-in-frequency algorithm. Equations (6.13) and (6.39) could have led us to same conclusion. These results will be verified experimentally in chapter 8.

Having discussed the output noise variance in detail, we now address the dynamic range considerations. We would like to obtain a formula for the output noise to signal ratio, by considering the overflow constraint in conjunction with our

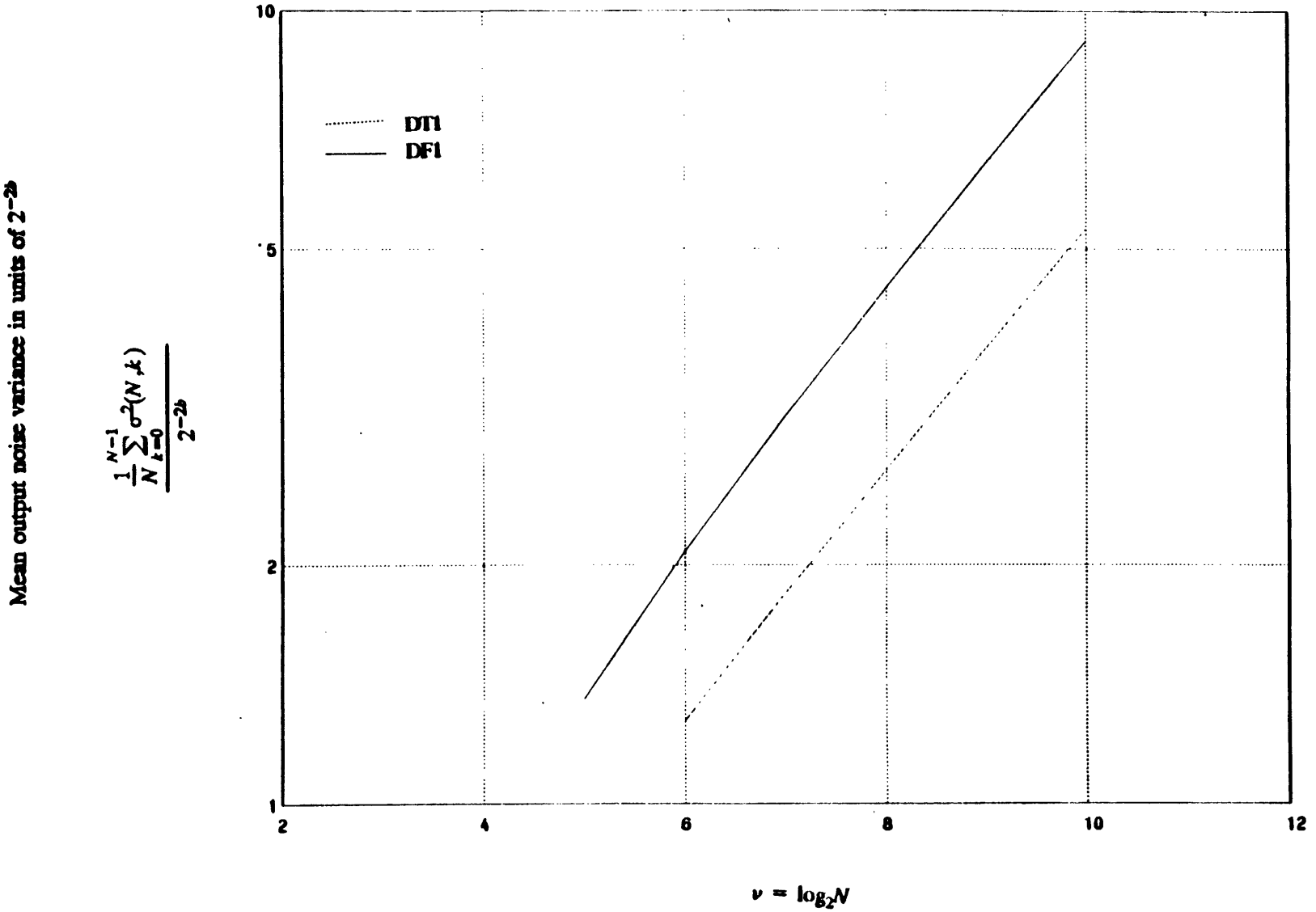


Figure 6.18 Output noise variance for fixed-point realization of DT1 and DF1 algorithms

noise analysis. Our approach here is to find a relationship between the maximum magnitude of elements of any two succeeding arrays in the algorithm. Let us denote the maximum magnitude of the input sequence by $|X_{in}|_{max}$ and the maximum magnitude of the elements of the *ith* array by $|X_i|_{max}$. Since a typical butterfly at the *nth* stage of the algorithm is given by

$$X_{n+1}(i) = X_n(i) + X_n(p+i)$$

$$X_{n+1}(i+p) = [\, X_n(i) - X_n(i+p) \,]\cos(\frac{2\pi i}{2p}) + [\, X_n(p-i) - X_n(2p-i) \,]\sin(\frac{2\pi i}{2p})$$

we get

$$|X_{n+1}|_{max} \leq 2 \,.\, \sqrt{2}\ |X_n|_{max}$$

Since there are ν stages of butterfly computations in an $N = 2^{\nu}$ point transform, we get

$$|X_{out}|_{max} < (\, 2 \,.\, \sqrt{2})^{\nu}\ |X_{in}|_{max}$$

Therefore, in order to guarantee that there will be no overflows at any stage of the algorithm (i.e. no intermediate quantity exceeds unity) we should choose the input as follows:

$$|X_{in}|_{max} < \frac{1}{(\, 2 \,.\, \sqrt{2})^{\nu}} \qquad (6.40)$$

To obtain an explicit expression for output signal variance, we assume $x(n)$ to be white and uniformly distributed in $(-\frac{1}{(\, 2 \,.\, \sqrt{2})^{\nu}}\ ,\ \frac{1}{(\, 2 \,.\, \sqrt{2})^{\nu}})$. Thus the variance of the input will be $\frac{1}{3\,(\, 2 \,.\, \sqrt{2})^{2\nu}}$ and the variance of the output signal can be written as:

$$\sigma^2_{out} = \frac{2^{\nu}}{3\,(2\,.\,\sqrt{2})^{2\nu}} \tag{6.41}$$

The upper bound on maximum magnitude of input shown in equation (6.40) can be tightened further using a numerical approach described in appendix A. This tighter upper bound is shown in the second column of table 6.2. For large values of N, doubling the transform size scales down the maximum magnitude of the input by a factor of 2.553. As it is mentioned in appendix A, the numerical technique used to generate table 6.2 results in the sufficient but not the necessary condition for the input to prevent overflow. The third column of table 6.2 shows the output signal variance provided the input is chosen to be white and uniformly distributed in the range specified by the upper bound in the second column.

Comparing tables 6.1 and 6.2 we see that maximum magnitude of the input which guarantees no overflows, is larger for the decimation- in-time algorithm than it is for the decimation-in-frequency algorithm. The same conclusion would have been reached, if we had used the results from equations (6.41) and (6.17). Thus for the decimation-in-frequency implementation, not only is the output noise variance higher, the dynamic range of the input signal which results in no overflows is less than that of the decimation-in-time algorithms. Figure 6.19 shows the average noise to signal ratio for the decimation-in-time and frequency algorithms DT2 and DF2 using equations (6.39) and (6.15) and tables 6.1 and 6.2. As it is expected the decimation-in-time algorithm has a more favorable error properties than the decimation-in-frequency implementations. Figures 6.17 and 6.19 will be verified experimentally in chapter 8.

Transform size N	Maximum input $\lvert X_{in} \rvert_{max}$	Variance of output $\frac{N \lvert X_{in} \rvert^2_{max}}{3}$
8	1.0×10^{-1}	2.9×10^{-2}
16	4.1×10^{-2}	9.0×10^{-3}
32	1.6×10^{-2}	2.7×10^{-3}
64	6.3×10^{-3}	8.4×10^{-4}
128	2.5×10^{-3}	2.6×10^{-4}
256	9.6×10^{-4}	7.9×10^{-5}
512	3.8×10^{-4}	2.4×10^{-5}
1024	1.5×10^{-4}	7.5×10^{-6}

Table 6.2 The upper bound on maximum magnitude of the input to ensure against overflow in fixed point implementation of the DF1 algorithm.

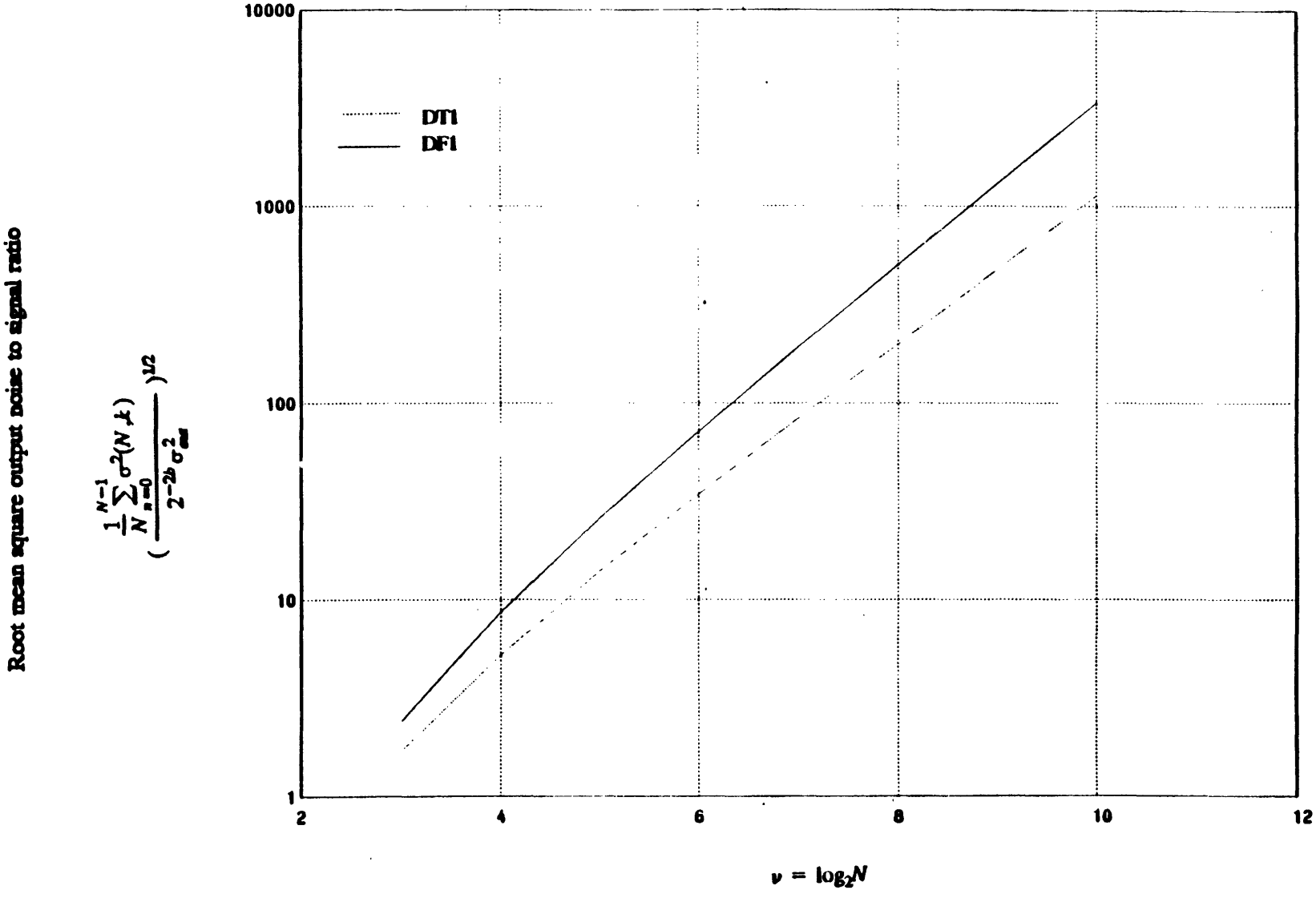


Figure 6.19 Theoretical noise to signal ratio for fixed-point realization of DF1 and DT1

6.2.3.2. Roundoff Noise Analysis of the DF1 Algorithm Using Floating-Point Arithmetic

The roundoff noise analysis of the DF1 algorithm using floating-point arithmetic is similar to that of the DT1 algorithm. There are two ways of finding the output noise variance. The approach taken in the first method which is more intuitive and results in a closed form expression is very similar to the intuitive approach described in section 6.2.1.2 for the decimation-in-time algorithm. Using this approach the output noise to signal ratio for an $N=2^{\nu}$ point white input sequence can be shown to be $2\nu\sigma_{\epsilon}^2$. The second method which is more accurate, is slightly different from the one discussed in section 6.2.1.2 and will be presented here.

We will assume that our input signal is zero mean white with variance σ_{in}^2. As usual we will be inserting white noise sources after each multiplier and adder in the signal flow graph. Figure 6.20 shows the decomposition used in computing an N-point Hartley transform using the decimation-in-frequency algorithm after the noise sources have been injected. Using the notation introduced earlier, the error in the *nth* term of the sequences $x_1(n)$ and $x_2(n)$ are given by

$$\epsilon_{x_1(n)} = [\, x(n) + x(\frac{N}{2}+n) \,](1 + \epsilon_{n_0})$$

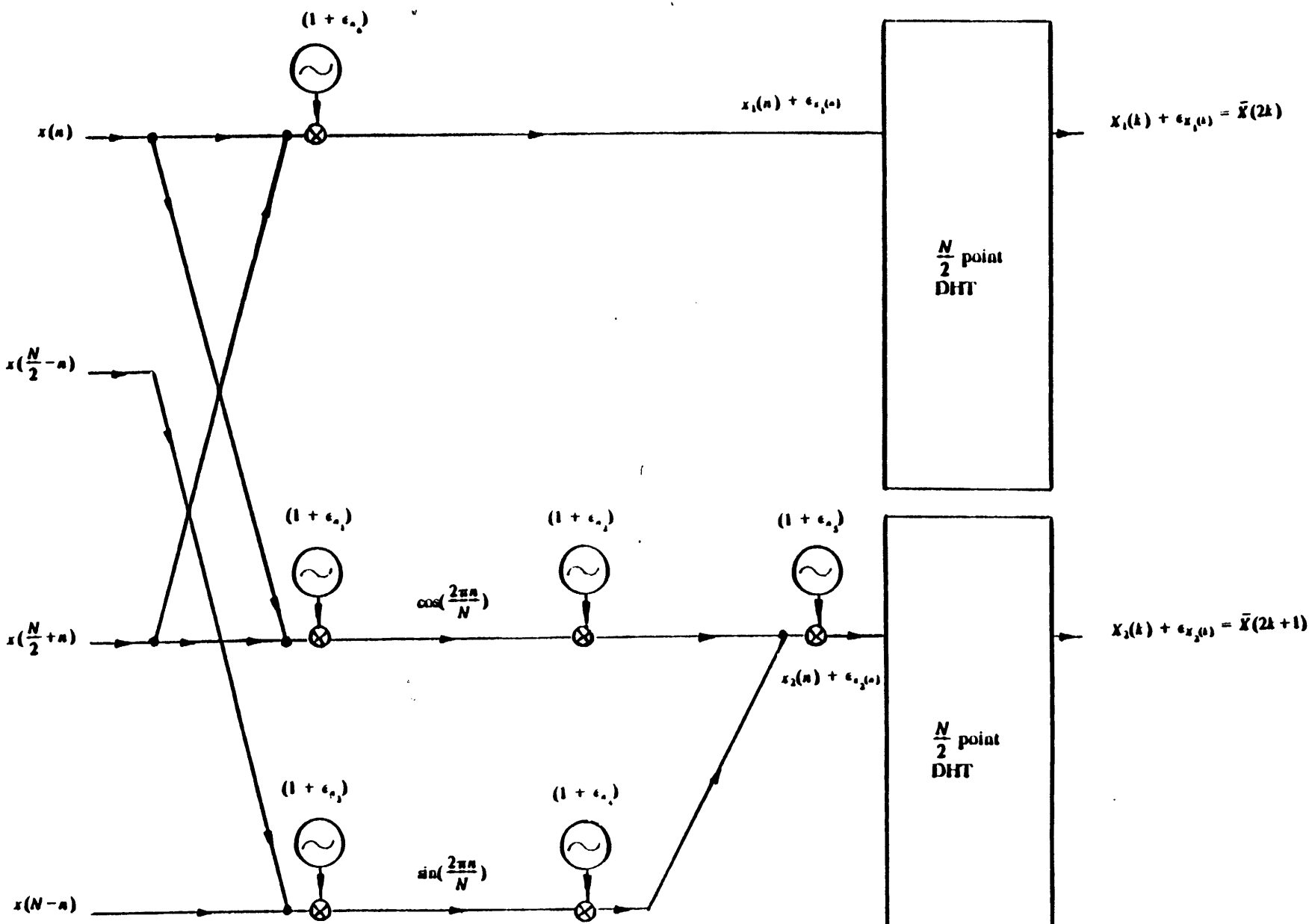


Figure 6.20 Statistical model for floating-point roundoff noise in a flow graph of the decimation-in-frequency decomposition of an N-point DHT computation into two N/2-point DHT computations using the DF1 algorithm

$$\epsilon_{x_2(n)} = \begin{cases} \Big\{ [\, x(n) - x(\frac{N}{2}+n) \,] \cos(\frac{2\pi n}{N}) \,(1+\epsilon_{n_1})\,(1+\epsilon_{n_2}) + & 0 < n < \frac{N}{2},\ n \neq \frac{N}{4} \\ \quad [\, x(\frac{N}{2}-n) - x(N-n) \,] \sin(\frac{2\pi n}{N}) \,(1+\epsilon_{n_3})\,(1+\epsilon_{n_4}) \Big\} (1+\epsilon_{n_5}) & \\ [\, x(n) - x(\frac{N}{2}+n) \,]\,(1+\epsilon_{n_7}) & n = 0 \\ [\, x(\frac{N}{2}-n) - x(N-n) \,]\,(1+\epsilon_{n_8}) & n = \frac{N}{4} \end{cases}$$

where ϵ_{n_i} denotes the roundoff error due to floating-point multiplications or additions. Its variance is denoted by σ_ϵ^2 and was defined in section 6.1.2. The variance of error at the input of $x_1(n)$ and $x_2(n)$ is given by

$$Var\,[\epsilon_{x_1(n)}] = 2\,\sigma_{in}^2\,\sigma_\epsilon^2 \tag{6.42a}$$

$$Var\,[\epsilon_{x_2(n)}] = \begin{cases} 6\,\sigma_{in}^2\,\sigma_\epsilon^2 & 0 < n < \frac{N}{2},\ n \neq \frac{N}{4} \\ 2\,\sigma_{in}^2\,\sigma_\epsilon^2 & n = 0,\ \frac{N}{4} \end{cases} \tag{6.42b}$$

If the $\frac{N}{2}$-point DHTs of $x_1(n)$ and $x_2(n)$ were computed with infinite precision, the variance of error at $X_1(k)$ and $X_2(k)$ due to the errors to the input sequences would have been given by

$$Var'[\epsilon_{X_1(k)}] = \frac{N}{2}\,Var\,[\epsilon_{x_1(n)}] = N\,\sigma_{in}^2\,\sigma_\epsilon^2 \tag{6.43a}$$

$$Var'[\epsilon_{X_2(k)}] = \sum_{n=0}^{\frac{N}{2}-1} Var\,[\epsilon_{x_2(n)}]\,cas^2(\frac{2\pi nk}{N/2}) = (3N-8)\sigma_{in}^2\,\sigma_\epsilon^2 \tag{6.43b}$$

Indeed the $\frac{N}{2}$-point DHT computations shown in figure 6.20 are not error free and they also contribute to the error in $X_1(k)$ and $X_2(k)$. Let $\sigma^2(\frac{N}{2},k)$ denote the output noise variance of the *kth* point of an $\frac{N}{2}$-point white sequence with variance of σ_{in}^2. Since $x_1(n)$ and $x_2(n)$ are white sequences with variance of $2\,\sigma_{in}^2$, the

variance of error due to $\frac{N}{2}$-point DHT computations at $X_1(k)$ and $X_2(k)$ are simply $2\,\sigma^2(\frac{N}{2},k)$. In order to find the total noise variance at $X_1(k)$ and $X_2(k)$ we add the variance of error due to the $\frac{N}{2}$-point DHT computations to the variance of error due to the input noise which was derived in equation (6.43). (Note that we are taking advantage of the fact that these two errors are statistically independent of each other). Thus we get:

$$Var\,[\epsilon_{X_1(k)}] = N\,\sigma_{in}^2\,\sigma_\epsilon^2 + 2\,\sigma^2(\frac{N}{2},k) \tag{6.44a}$$

$$Var\,[\epsilon_{X_2(k)}] = (3N-8)\,\sigma_{in}^2\,\sigma_\epsilon^2 + 2\,\sigma^2(\frac{N}{2},k) \tag{6.44b}$$

Since $X_1(k)$ and $X_2(k)$ constitute the even and odd points of $X(k)$, the distribution of output noise variance for N-point white sequences is given by

$$\sigma^2(N,k) = \begin{cases} 2\,\sigma^2(\frac{N}{2},k) + N\,\sigma_{in}^2\,\sigma_\epsilon^2 & k\ even \\ 2\,\sigma^2(\frac{N}{2},k) + (3N-8)\,\sigma_{in}^2\,\sigma_\epsilon^2 & k\ odd \end{cases} \tag{6.45}$$

The distribution of the variance of error for 256-point white sequences using equation (6.45) is shown in figure 6.21. It is clearly different from figure 6.8 where the distribution for the decimation-in-time algorithm is shown.

If $\sigma^2(N,k)$ of equation (6.45) is averaged over the frequency points, we can find a closed form expression for the mean output noise variance of an $N = 2^\nu$ sequence by solving the following difference equation

$$< \sigma^2(2^\nu,k) > = 2 < \sigma^2(2^{\nu-1},k) > + 2^{\nu+1} - 4 \tag{6.46}$$

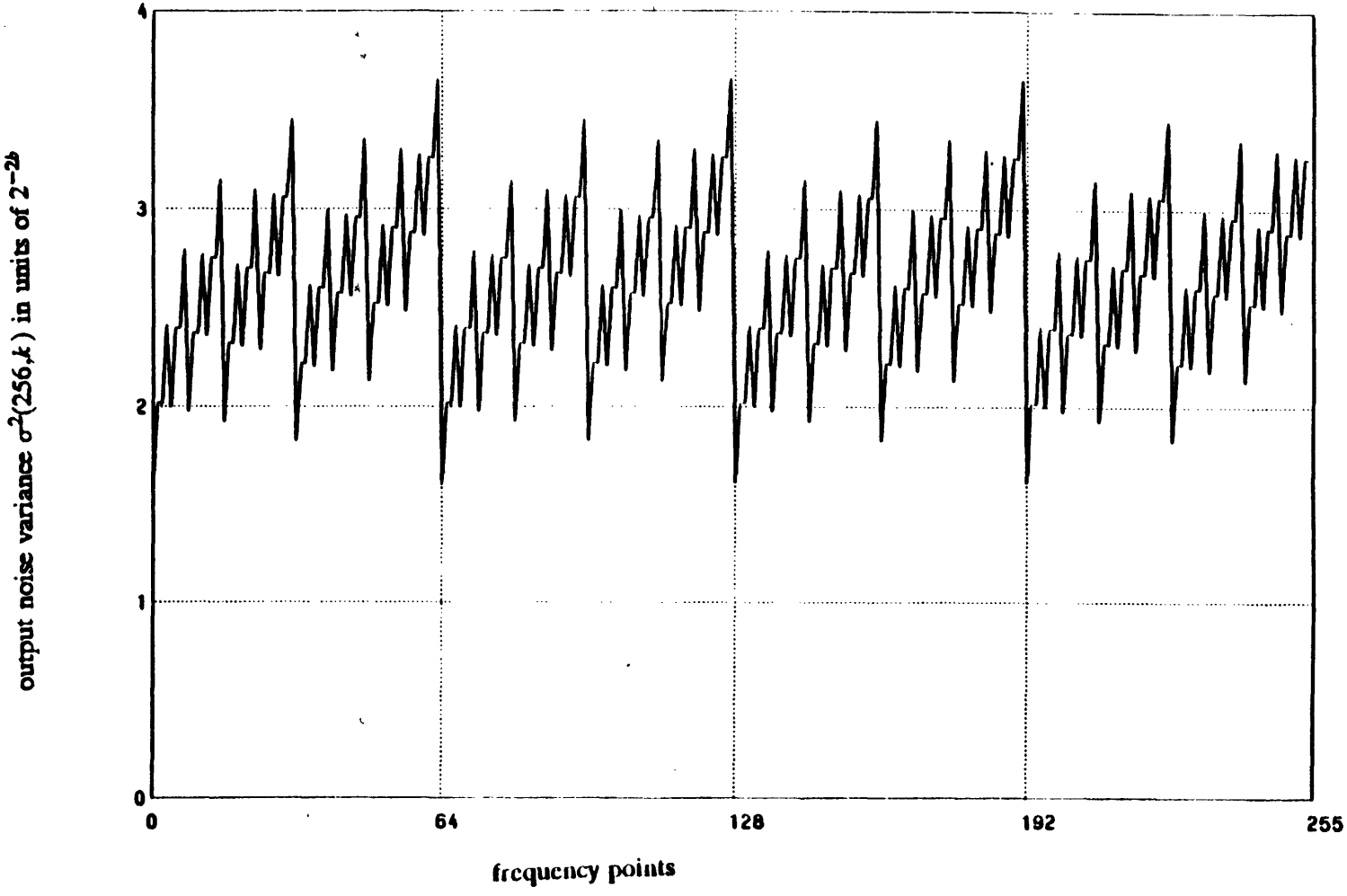


Figure 6.21 Distribution of the output noise variance of 256-point white sequences for floating-point implementation of the DF1 algorithm.

where <.> is used to show averaging over frequency points. Since a 4-point DHT only involves additions and subtractions, we get:

$$< \sigma^2(4,k) > = 8\, \sigma_{in}^2\, \sigma_\epsilon^2$$

Solving equation (6.46) the closed form expression for the mean output noise variance for $\nu > 2$ is given by

$$< \sigma^2(2^\nu,k) > = (2\, \nu\, N - 3N + 4)\sigma_\epsilon^2 \qquad (6.47)$$

Thus the more accurate analysis which led us to equation (6.47) is slightly different from the simplified one and their difference becomes negligible for large values of N.

The output noise to signal ratio for the decimation-in-time and frequency algorithms as a function of transform size are shown in figure 6.22. Although the distribution of variance of error among the frequency points of these two algorithms are different, their average output noise to signal ratio is almost identical. This is in contrast with the fixed-point case where the output noise to signal ratio for the decimation-in-time and frequency algorithms were different from each other. The experimental results verifying the theoretical predictions in figures 6.21 and 6.22 will be shown in chapter 8.

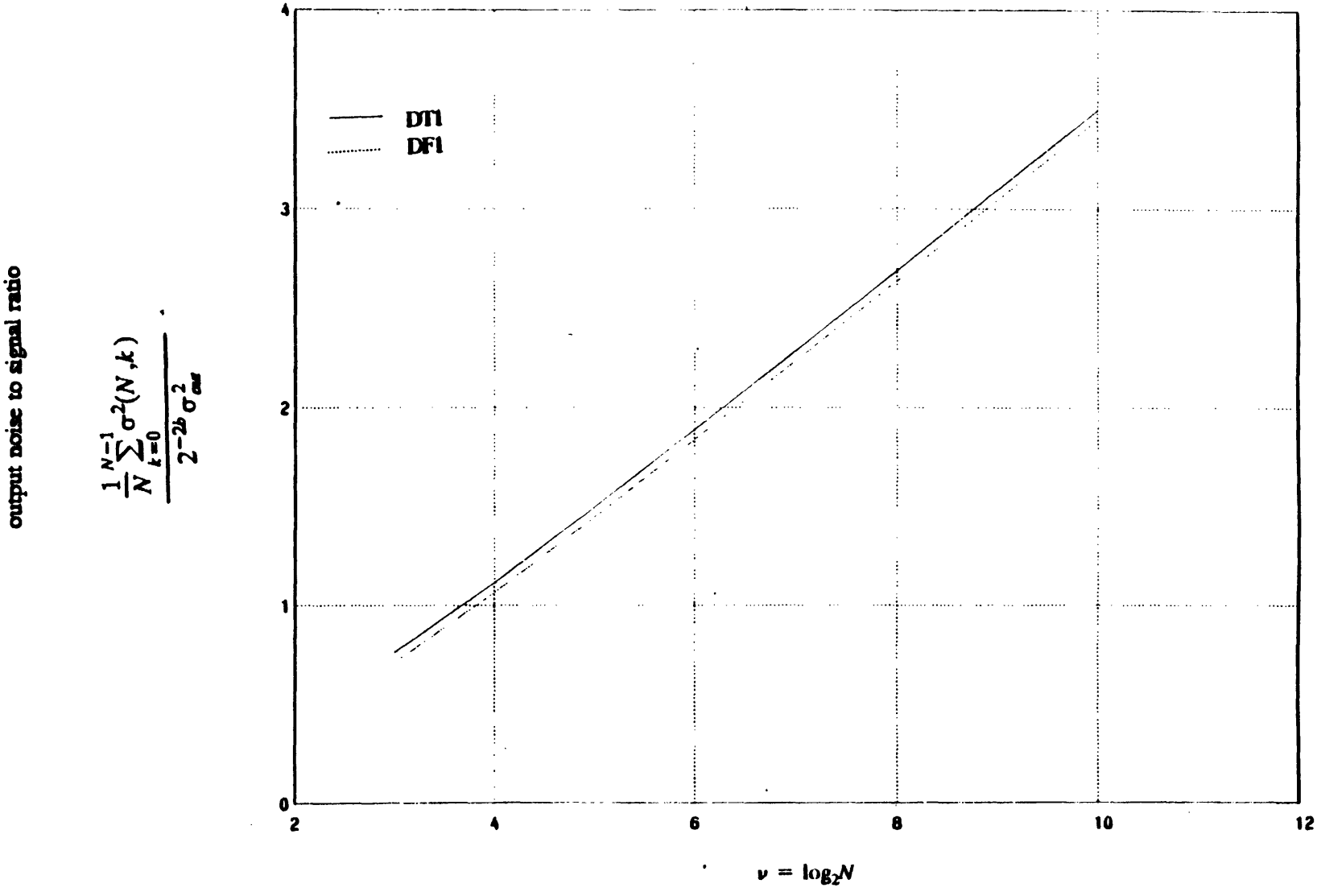


Figure 6.22 Theoretical noise to signal ratio for floating-point realization of DF1 and DT1

CHAPTER 7: Theoretical Noise Analysis of the DT2 and DF2 Algorithms

In chapter 6, we analyzed the statistical error properties of the DHT algorithms of chapter 3. This chapter discusses the effects of floating-point and fixed-point roundoff errors in the algorithms presented in chapter 5. More specifically, sections 7.1 and 7.2 will describe the error characteristics of the DT2 and DF2 algorithms respectively. Our approach in this chapter is very similar to that of chapter 6; we will assume that the roundoff error due to multiplications and additions are uncorrelated with each other and with the input. We then associate noise source generators for every multiplier and adder that appears in the flow graph of the specific algorithm under investigation and analyze their effects on the output. In chapter 8, the experimental results supporting the theoretical derivations of this chapter and chapter 6 will be presented.

7.1. Roundoff Noise in the DT2 Algorithm

7.1.1. Roundoff Noise Analysis of the DT2 Algorithm Using Fixed-Point Arithmetic

In this section we will analyze the effects of roundoff errors due to fixed-point implementation of the DT2 algorithm described in section 5.1.1. At first we ignore the overflow constraint and derive the output noise variance analytically. Then the dynamic range issues will be considered.

We will insert additive, signal independent, white noise sources after each multiplier in the signal flow graph of the algorithm. Our approach in finding the output

noise variance is again a recursive one. Recall that using the DT2 algorithm, the problem of finding an N-point DHT is decomposed into that of finding two $\frac{N}{2}$-point DHTs. This is shown pictorially in figure 7.1 where $x(n)$ an N-point sequence is decomposed into two $\frac{N}{2}$-point sequences $x_1(n)$ and $x_2(n)$. From equations (5.2) and (5.4) we can conclude that $x(n)$ is related to $x_1(n)$ and $x_2(n)$ in the following manner:

$$x_1(n) = x(2n) \qquad 0 \le k < \frac{N}{2} \tag{7.1a}$$

$$x_2(n) = x(2n+1) + x(2n-1) \qquad 0 \le k < \frac{N}{2} \tag{7.1b}$$

where

$$x(-1) \equiv x(N-1)$$

Since additions in fixed-point arithmetic are error free, no error is introduced in the process of forming the two subsequences $x_1(n)$ and $x_2(n)$ from $x(n)$. $X(k)$, the desired DHT of $x(n)$ is computed from $X_1(k)$ and $X_2(k)$ the transforms of the subsequences $x_1(n)$ and $x_2(n)$ in the following manner:

$$X(k) = \begin{cases} X_1(k) + \dfrac{1}{2\cos(\frac{2\pi k}{N})} X_2(k) & 0 \le k < N,\ k \ne \frac{N}{4},\ k \ne \frac{3N}{4} \\ X_1(k) + \sum\limits_{n=0}^{\frac{N}{2}-1} x(2n+1)(-1)^{n+\frac{4k-N}{2N}} & k = \frac{N}{4},\ \frac{3N}{4} \end{cases} \tag{7.2}$$

Using the above equation we can find the output noise variance of the N-point sequence $x(n)$ from the output noise variance of the $\frac{N}{2}$-point sequences $x_1(n)$ and $x_2(n)$. More specifically, by inspection of figure 7.1 the error in $X(k)$ denoted by

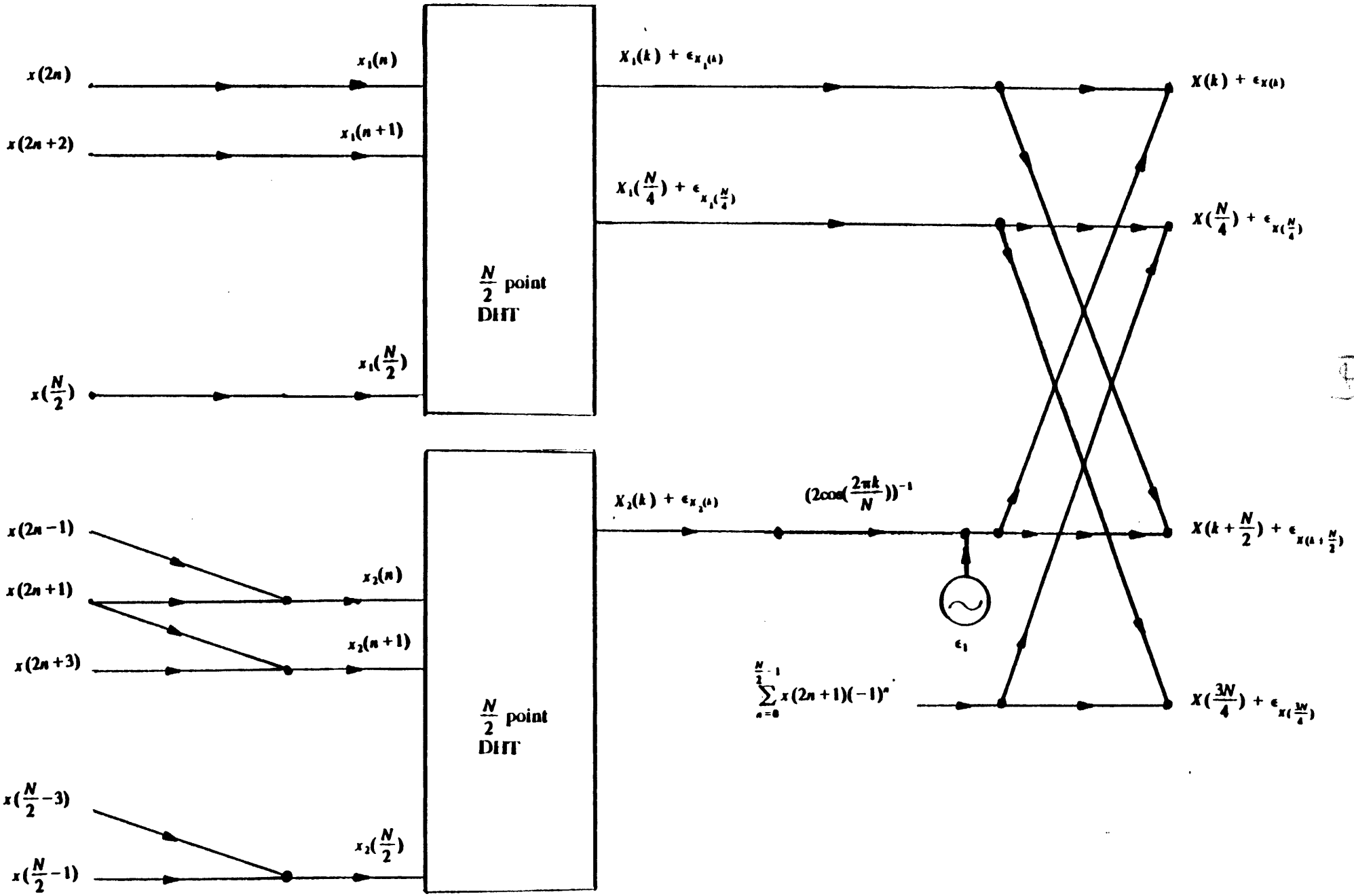


Figure 7.1 Statistical model for fixed-point roundoff noise in a flow graph of the decimation-in-time decomposition of an N-point DHT computation into two N/2-point DHT computations using the DIT2 algorithm

$\epsilon_{X(k)}$ can be written as:

$$\epsilon_{X(k)} = \begin{cases} \epsilon_{X_1(k)} + \dfrac{1}{2\cos(\frac{2\pi k}{N})}\epsilon_{X_2(k)} + \epsilon_1 & 0 \le k < N,\ k \ne \dfrac{N}{4}, \dfrac{3N}{4} \\ \epsilon_{X_1(k)} & k = \dfrac{N}{4}, \dfrac{3N}{4} \end{cases} \quad (7.3)$$

where ϵ_1 denotes the error due to the multiplication of $X_2(k)$ with $\dfrac{1}{2\cos(\frac{2\pi k}{N})}$ in equation (7.2). Its variance σ_ϵ^2 was defined in section 6.1.1. Using equation (7.3) the variance of error at the *kth* point of an N-point DHT denoted by $\sigma^2(N,k)$ can be written as:

$$\sigma^2(N,k) = \begin{cases} [\,1 + \dfrac{1}{4\cos^2(\frac{2\pi k}{N})}\,]\,\sigma^2(\frac{N}{2},k) + \sigma_\epsilon^2 & 0 < k < \dfrac{N}{2},\ k \ne \dfrac{N}{4} \\ [\,1 + \dfrac{1}{4\cos^2(\frac{2\pi k}{N})}\,]\,\sigma^2(\frac{N}{2},k) + 0.75\,\sigma_\epsilon^2 & k = 0 \quad (7.4a) \\ \sigma^2(\frac{N}{2},k) & k = \dfrac{N}{4} \end{cases}$$

$$\sigma^2(N,k) = \sigma^2(N,k-\frac{N}{2}) \qquad \frac{N}{2} \le k < N \qquad (7.4b)$$

Note that for $k = 0$, $X_2(k)$ of equation (7.2) is going to be multiplied by $\frac{1}{2}$. When a fixed-point number of b bits is multiplied by $\frac{1}{2}$, the magnitude of error is either zero or $\frac{1}{2}2^{-b}$ depending upon whether the number is even or odd. Thus for the special case of $k = 0$, the error model of section 6.1.1 has to be slightly modified. Therefore the variance of error due to multiplication by $\frac{1}{2}$ is going to be $\frac{3}{4}\sigma_\epsilon^2$ where σ_ϵ^2 is defined in section 6.1.1.

Equation (7.4) is the result we were looking for. It says that the output noise variance distribution for N-point sequences can be obtained from the output noise variance for $\frac{N}{2}$-point sequences. The distribution of error for the DHT of 256-point sequences using equation (7.4) is shown in figure 7.2. There are two peaks at $k = \frac{N}{4} \pm 1$ and $k = \frac{3N}{4} \pm 1$. This is due to the fact that $\frac{1}{2\cos(\frac{2\pi k}{N})}$ attains its highest value at these values of k (Note that $k = \frac{N}{4}, \frac{3N}{4}$ are the special cases and are treated separately as shown in equation (7.2)). Thus, whatever error that has been accumulated in computing $X_2(k)$ gets multiplied by a relatively large factor in the last stage of the algorithm, causing peaks in the output noise variance.

Having discussed the output noise variance, let us now consider the dynamic range issues. Numerical techniques shown in appendix A were used to find an upper bound on the magnitude of the input signal in order to prevent overflow. These bounds are shown in the second column of table 7.1. For large values of N, doubling the transform size scales down the maximum magnitude of the input by a factor of 4.0. As it is mentioned in appendix A, the numerical technique used to generate table 7.1 results in the sufficient but not the necessary condition for the input to prevent overflows. The third column of table 7.1 indicates the variance of output signal given that the input is white and uniformly distributed in the range specified by the upper bound shown in the second column. The values at the third column together with equation (7.4) can be used to find the output noise to signal variance for the DT2 algorithm as a function of N. The noise to signal ratio for the

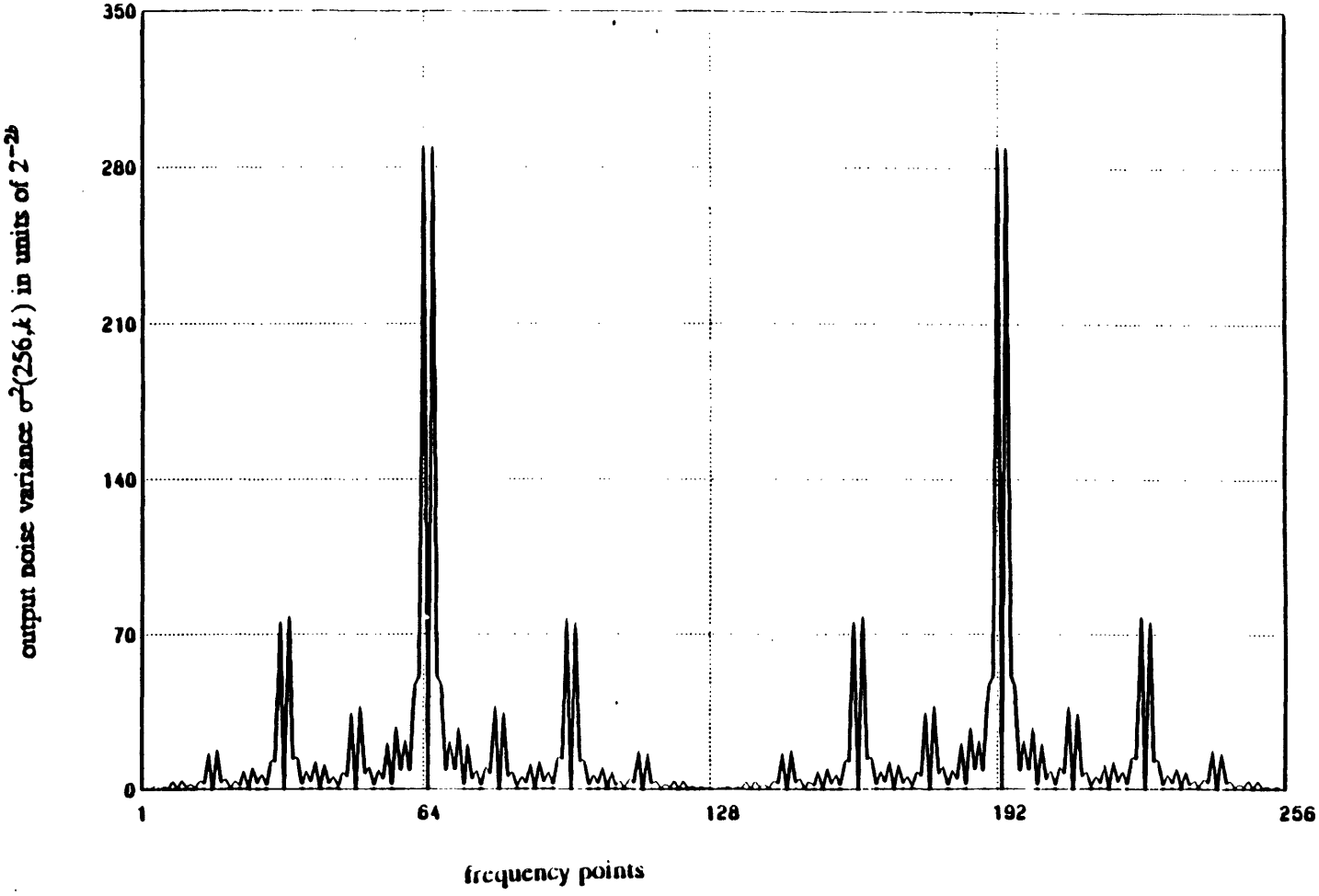


Figure 7.2 Distribution of the output noise variance of an 256 point sequence for fixed-point realization of DT2

Transform size N	Maximum input $\lvert X_{in} \rvert_{max}$	Variance of output $\frac{N \lvert X_{in} \rvert^2_{max}}{3}$
8	1.0×10^{-1}	2.9×10^{-2}
16	5.0×10^{-2}	1.3×10^{-2}
32	1.2×10^{-2}	1.6×10^{-3}
64	2.7×10^{-3}	1.5×10^{-4}
128	6.4×10^{-4}	1.8×10^{-5}
256	1.5×10^{-4}	2.0×10^{-6}
512	3.8×10^{-5}	2.5×10^{-7}
1024	9.5×10^{-6}	3.1×10^{-8}

Table 7.1 The upper bound on maximum magnitude of the input to ensure against overflow in fixed point implementation of the DT2 algorithm.

DT1 and DT2 algorithms are shown in figure 7.3. The theoretical results of figures 7.2 and 7.3 will be verified experimentally in chapter 8.

7.1.2. Roundoff Noise Analysis of the DT2 Algorithm Using Floating-Point Arithmetic

The roundoff noise analysis using floating-point arithmetic is somewhat similar to that of the fixed-point arithmetic. We shall assume that the input signal is white with variance of σ_{in}^2 and insert multiplicative, signal independent white noise sources after each multiplier and adder. Our approach in finding the output noise variance is a recursive one.

The decomposition used for computing the N-point DHT of a white sequence $x(n)$ with variance of σ_{in}^2 is shown in figure 7.4. Using equation (7.2) and figure 7.4, $X(k)$ the DHT of $x(n)$ can be written as:

$$X(k) = \begin{cases} (1+\epsilon_{k_3})\left\{[X_1(k)+\epsilon_{X_1(k)}]+(1+\epsilon_{k_2})\dfrac{X_2(k)+\epsilon_{X_2(k)}}{2\cos(\frac{2\pi k}{N})}\right\} & 0 \le k < N,\ k \neq \frac{N}{4}, \frac{3N}{4} \\ (1+\epsilon_{k_3})\left\{[X_1(k)+\epsilon_{X_1(k)}]+\sum_{n=0}^{\frac{N}{2}-1} x(2n+1)(-1)^{n+\frac{4k-N}{2N}}\prod_{i=n}^{\frac{N}{2}-1}(1+\epsilon_i)\right\} & k = \frac{N}{4}, \frac{3N}{4} \end{cases} \quad (7.5)$$

where ϵ_i and ϵ_{k_i} denote the roundoff error due to fixed-point additions or multiplications. Their variance is denoted by σ_ϵ^2 and was defined in section 6.1.2. $\epsilon_{X_i(k)}$ denotes the error in computation of $X_i(k)$. Let $\sigma^2(\frac{N}{2},k)$ denote the output noise variance at the *kth* point of an $\frac{N}{2}$-point white sequence with variance σ_{in}^2. Since $x_1(n)$ is the even points of $x(n)$, it is white and its variance is σ_{in}^2. Thus the

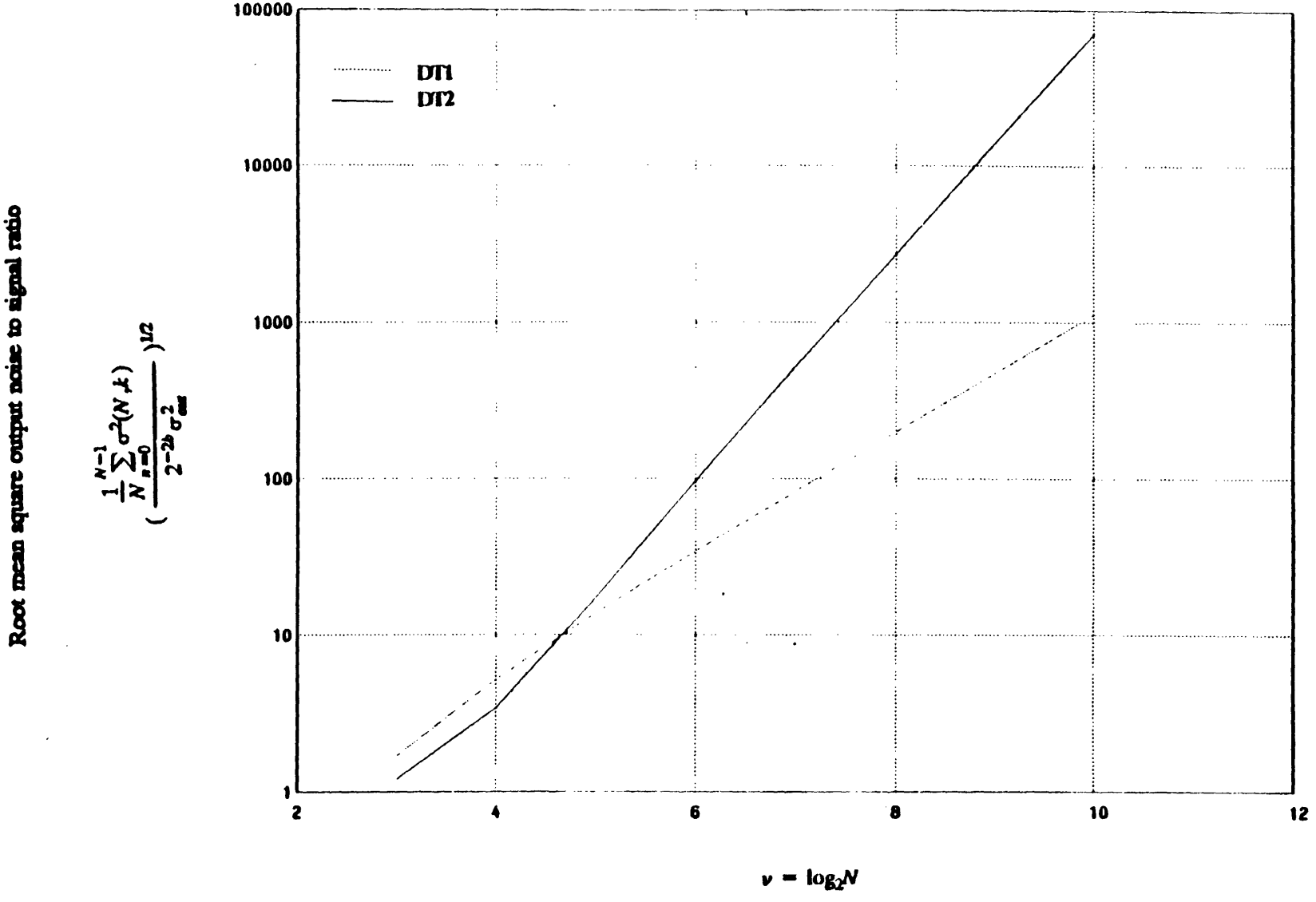


Figure 7.3 Theoretical noise to signal ratio for fixed-point realization of DT2 and DT1

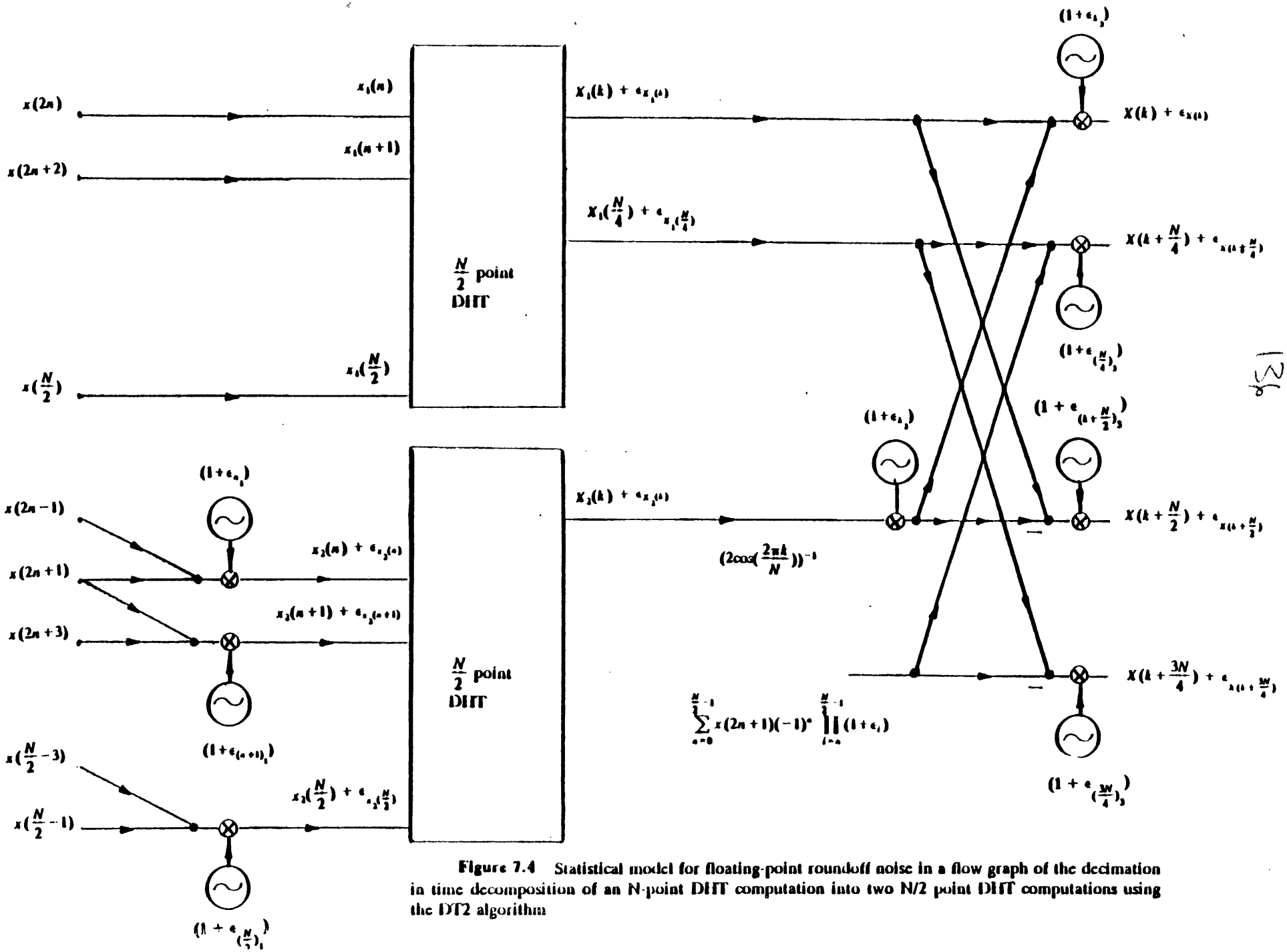


Figure 7.4 Statistical model for floating-point roundoff noise in a flow graph of the decimation in time decomposition of an N-point DFT computation into two N/2 point DFT computations using the DIT2 algorithm

variance of error at $X_1(k)$ is given by

$$Var[\epsilon_{X_1(k)}] = \sigma^2(\frac{N}{2},k) \tag{7.6a}$$

The sequence $x_2(n)$ is given by equation (7.1b). Therefore the variance of error at $x_2(n)$ is

$$\epsilon_{x_2(n)} = 2\,\sigma_{in}^2\,\sigma_\epsilon^2$$

and the variance of error at $X_2(k)$ due to noise in $x_2(n)$ (i.e. ignoring the noise due to $\frac{N}{2}$-point DHT computation of $X_2(k)$) is

$$Var'[\,\epsilon_{X_2(k)}\,] = N\,\sigma_{in}^2\,\sigma_\epsilon^2$$

Since the total error in $X_2(k)$ is due to the noise in $x_2(n)$ and due to roundoff error introduced in $\frac{N}{2}$-point DHT computation of $X_2(k)$, we get

$$Var[\,\epsilon_{X_2(k)}\,] = N\,\sigma_{in}^2\,\sigma_\epsilon^2 + \sigma'^2(\frac{N}{2},k) \tag{7.6b}$$

where $\sigma'^2(\frac{N}{2},k)$ denotes the variance of error at the *kth* point of the DHT of an $\frac{N}{2}$-point sequence with similar statistical characteristics to $x_2(n)$, i.e. sequences for which the variance of each point is given by $2\sigma_{in}^2$ and the covariance between any two adjacent points is σ_{in}^2. In other words

$$E\,[\,x_2(n)\,x_2(n+1)\,] = \sigma_{in}^2 \qquad\qquad 0 \le n < \frac{N}{2}$$

where

$$x_2(\frac{N}{2}) \equiv x_2(0)$$

Using equations (7.5) and (7.6) the variance of error at the *kth* point of the white sequence $x(n)$ can be easily found in terms of $\sigma'^2(\frac{N}{2},k)$ and $\sigma^2(\frac{N}{2},k)$ in

the following manner:

$$\sigma^2(N,k) = \begin{cases} \frac{3}{2}N\sigma_{in}^2\sigma_\epsilon^2 + \sigma^2(\frac{N}{2},k) + & 0 < k < \frac{N}{2},\ k \neq \frac{N}{4} \\ \quad \frac{1}{4\cos^2(\frac{2\pi k}{N})}[\,N\,\sigma_{in}^2\sigma_\epsilon^2 + \sigma'^2(\frac{N}{2},k) & \\ N\sigma_{in}^2\sigma_\epsilon^2 + \sigma^2(\frac{N}{2},k) + & k = 0 \quad (7.7a) \\ \quad \frac{1}{4\cos^2(\frac{2\pi k}{N})}[\,N\,\sigma_{in}^2\sigma_\epsilon^2 + \sigma'^2(\frac{N}{2},k) & \\ N\sigma_{in}^2\sigma_\epsilon^2 + \sigma^2(\frac{N}{2},k) + [\,\frac{N}{4}(\frac{N}{2}+1) - 1\,]\sigma_{in}^2\sigma_\epsilon^2 & k = \frac{N}{4} \end{cases}$$

$$\sigma^2(N,k) = \sigma^2(N,k-\frac{N}{2}) \qquad \frac{N}{2} \le k < N \qquad (7.7b)$$

$\sigma^2(N,0)$ is considered separately in the above equation because for $k = 0$, $\frac{1}{2\cos(\frac{2\pi k}{N})}$, the factor by which $X_2(k)$ is multiplied becomes equal to $\frac{1}{2}$. Since in floating-point arithmetic multiplication by $\frac{1}{2}$ corresponds to decrementing the exponent by 1, it is basically error free. Also $k = \frac{N}{4}$ is the special case in our algorithm which is computed in a different manner from the rest of the points as shown in equation (7.2).

Equation (7.7) shows the way $\sigma^2(N,k)$ can be found in terms of $\sigma^2(\frac{N}{2},k)$ and $\sigma'^2(\frac{N}{2},k)$. To complete the recursion we have to find a way of expressing $\sigma'^2(N,k)$ in terms of $\sigma^2(\frac{N}{2},k)$ and $\sigma'^2(\frac{N}{2},k)$. To obtain $\sigma'^2(N,k)$ consider an N-point white sequence $y(n)$ with variance $2\,\sigma_{in}^2$ and the covariance between its adjacent

points σ_{in}^2. By definition, the output noise variance for $Y(k)$ is $\sigma'^2(N,k)$. We are interested in finding $\sigma'^2(N,k)$ in terms of $\sigma'^2(\frac{N}{2},k)$ and $\sigma^2(\frac{N}{2},k)$ (the output noise variance of an $\frac{N}{2}$-point white sequence with variance σ_{in}^2). In order to compute $Y(k)$, we will split $y(n)$ into two $\frac{N}{2}$-point sequences $y_1(n)$ and $y_2(n)$ in a manner shown in equation (7.1). Although $y(n)$ is not white, because of its special structure, $y_1(n)$ which consists of the even points of $y(n)$ is white with variance $2\sigma_{in}^2$. Thus the output noise variance for $Y_1(k)$ is $2\sigma^2(\frac{N}{2},k)$. The sequence $y_2(n)$ is given by:

$$y_2(n) = [\, y(2n+1) + y(2n-1) \,](\, 1 + \epsilon_n \,) \qquad 0 \le n < \frac{N}{2}$$

where

$$y(-1) \equiv y(N-1)$$

and ϵ_n is the roundoff error due to floating-point addition. The variance of $y_2(n)$ is $4\sigma_{in}^2$ and the covariance between its neighboring points is $2\sigma_{in}^2$. Thus by definition the variance of output error at $Y_2(k)$ only due to $\frac{N}{2}$-point DHT computation of $Y_2(k)$ is given by $2\sigma'^2(\frac{N}{2},k)$. Similarly, the variance of error at $Y_2(k)$ only due to the error in forming $y_2(n)$ is given by $2\,N\,\sigma_\epsilon^2\sigma_{in}^2$. Since these two sources of errors are independent of each other, the total variance of error at $Y_2(k)$ denoted by $\epsilon_{Y_2(k)}$ is:

$$Var[\epsilon_{Y_2(k)}] = 2\sigma'^2(\frac{N}{2},k) + 2\,N\,\sigma_\epsilon^2\,\sigma_{in}^2$$

Using the fact that $Y_1(k)$ and $Y_2(k)$ can be recombined to form $Y(k)$ and taking into account that:

$$Var[Y(k)] = 4N \cos^2(\frac{\pi k}{N}) \sigma_{in}^2$$

$$Var[Y_1(k)] = N \sigma_{in}^2$$

$$Var[Y_2(k)] = 4 \cos^2(\frac{2\pi k}{N}) \sigma_{in}^2$$

the variance of error for $Y(k)$ denoted by $\sigma'^2(N,k)$ can be easily written in terms of $\sigma^2(\frac{N}{2},k)$ and $\sigma'^2(\frac{N}{2},k)$:

$$\sigma'^2(N,k) = \begin{cases} N \sigma_{in}^2 \sigma_\epsilon^2 [4 \cos^2(\frac{\pi k}{N}) + 1] + 2\sigma^2((\frac{N}{2},k))_{\frac{N}{2}} + \dfrac{\sigma'^2((\frac{N}{2},k))_{\frac{N}{2}} + N}{2\cos^2(\frac{2\pi k}{N})} & 0 < k < N, k \neq \frac{N}{4}, \frac{3N}{4} \\ 4N \cos^2(\frac{\pi k}{N}) \sigma_{in}^2 \sigma_\epsilon^2 + 2\sigma^2((\frac{N}{2},k))_{\frac{N}{2}} + \dfrac{\sigma'^2((\frac{N}{2},k))_{\frac{N}{2}} + N}{2\cos^2(\frac{2\pi k}{N})} & k = 0 \\ 4N \cos^2(\frac{\pi k}{N}) \sigma_{in}^2 \sigma_\epsilon^2 + 2\sigma^2((\frac{N}{2},k))_{\frac{N}{2}} + [\frac{N}{2}(\frac{N}{2}+1) - 2]\sigma_{in}^2 \sigma_\epsilon^2 & k = \frac{N}{4}, \frac{3N}{4} \end{cases} \quad (7.8)$$

Equations (7.7) and (7.8) show the way to compute $\sigma^2(N,k)$ and $\sigma'^2(N,k)$ in terms of $\sigma^2(\frac{N}{2},k)$ and $\sigma'^2(\frac{N}{2},k)$. The recursion is now complete and the distribution of the output noise variance for arbitrarily long sequences can be found recursively. The output noise variance for the DHT of 256-point white sequences is shown in figure 7.5. There are two peaks at $k = \frac{N}{4} \pm 1$ and $k = \frac{3N}{4} \pm 1$. The reason for these peaks is explained in section 7.1.1. Figure 7.6 shows the output noise to signal ratio for white sequences as a function of their lengths for the DT2 and DT1 algorithms. The experimental results verifying the theoretical predictions of

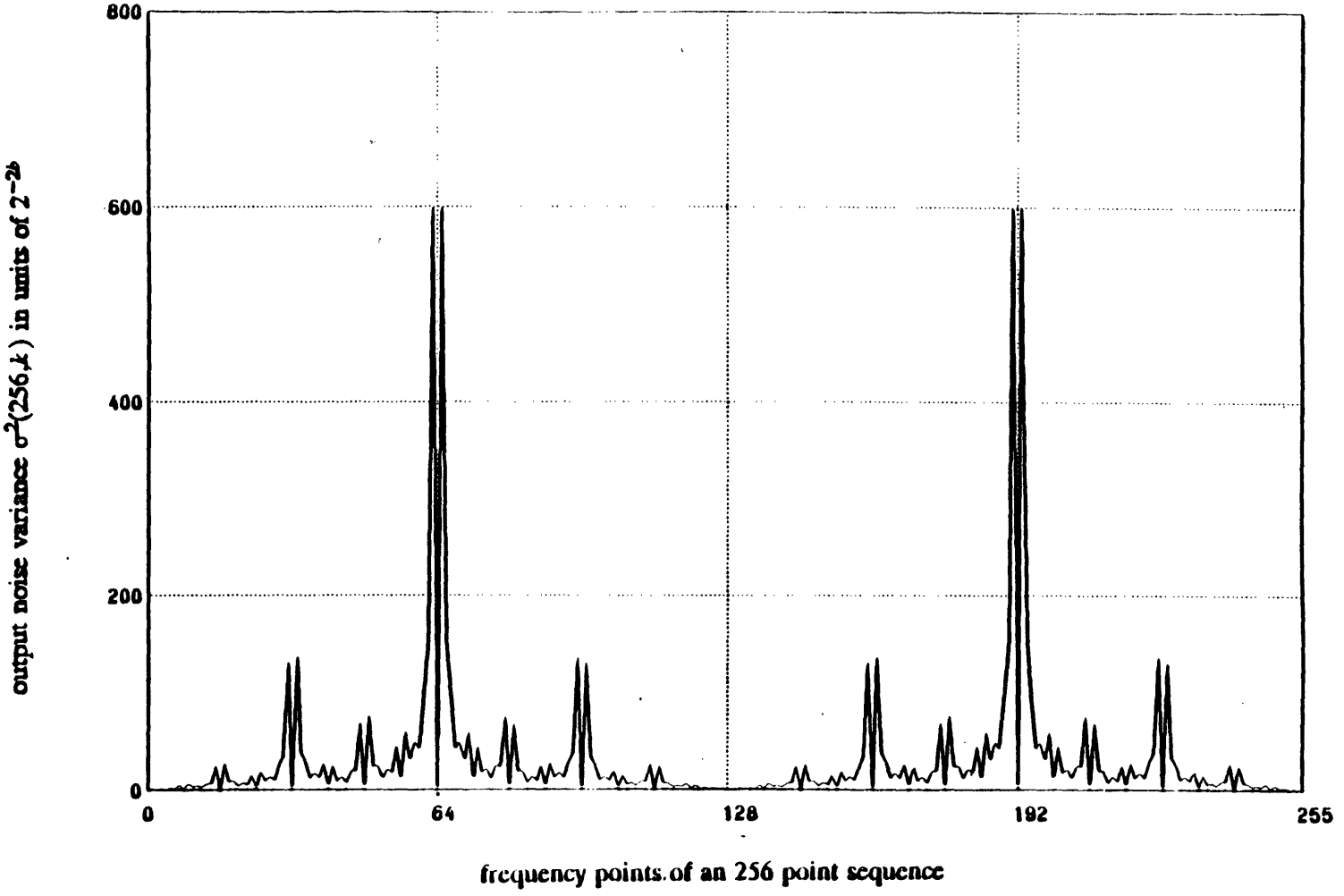


Figure 7.5 Distribution of the output noise variance of 256-point white sequences for floating-point implementation of the DT2 algorithm.

output noise to signal ratio

$$\frac{\frac{1}{N}\sum_{k=0}^{N-1}\sigma^2(N,k)}{2^{-2b}\sigma_{out}^2}$$

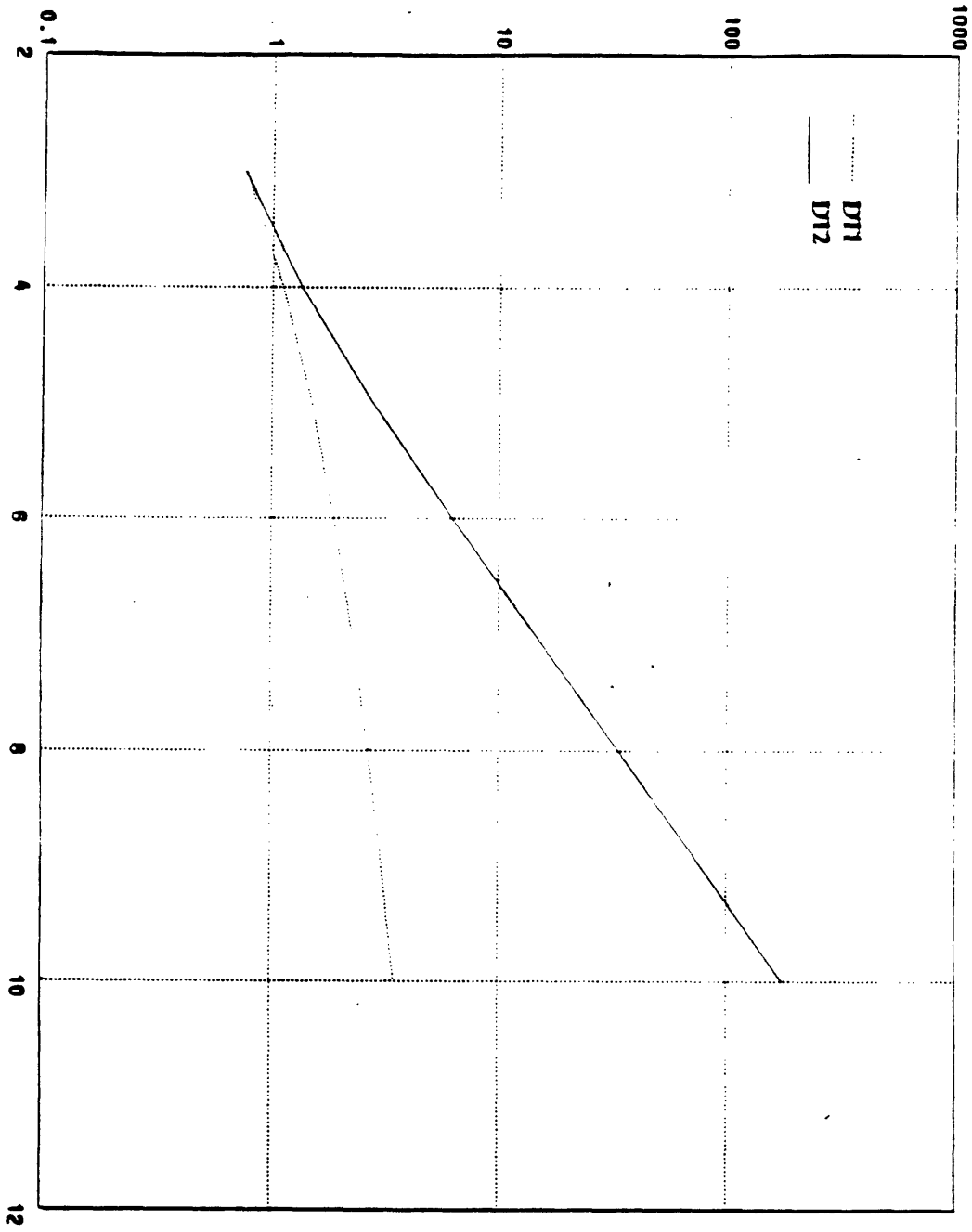


Figure 7.6 Theoretical noise to signal ratio for floating-point realization of DT2 and DT1

this section will be presented in chapter 8.

7.2. Roundoff Noise in the DF2 Algorithm

7.2.1. Roundoff Noise Analysis of the DF2 algorithm Using Fixed-Point Arithmetic

In this section we will analyze the effects of roundoff errors due to fixed-point implementation of the DF2 algorithm described in section 5.1.4. At first we ignore the overflow constraint and derive the output noise variance analytically. Then the dynamic range issues will be considered.

We will insert additive, signal independent, white noise sources after each multiplier in the signal flow graph of the algorithm. Our approach in finding the output noise variance is again a recursive one. Recall that using the DF2 algorithm, the problem of finding an N-point DHT is decomposed into that of finding two $\frac{N}{2}$-point DHTs. This is shown pictorially in figure 7.7 where $x(n)$ is the N-point sequence which is going to be transformed. $x_1(n)$ and $x_2(n)$ denote the two $\frac{N}{2}$-point subsequences that $x(n)$ is decomposed into. From equations (5.15) through (5.19) we can conclude that $x(n)$ is related to $x_1(n)$ and $x_2(n)$ in the following manner:

$$x_1(n) = x(n) + x(\frac{N}{2} + n) \qquad 0 \le k < \frac{N}{2} \qquad (7.9a)$$

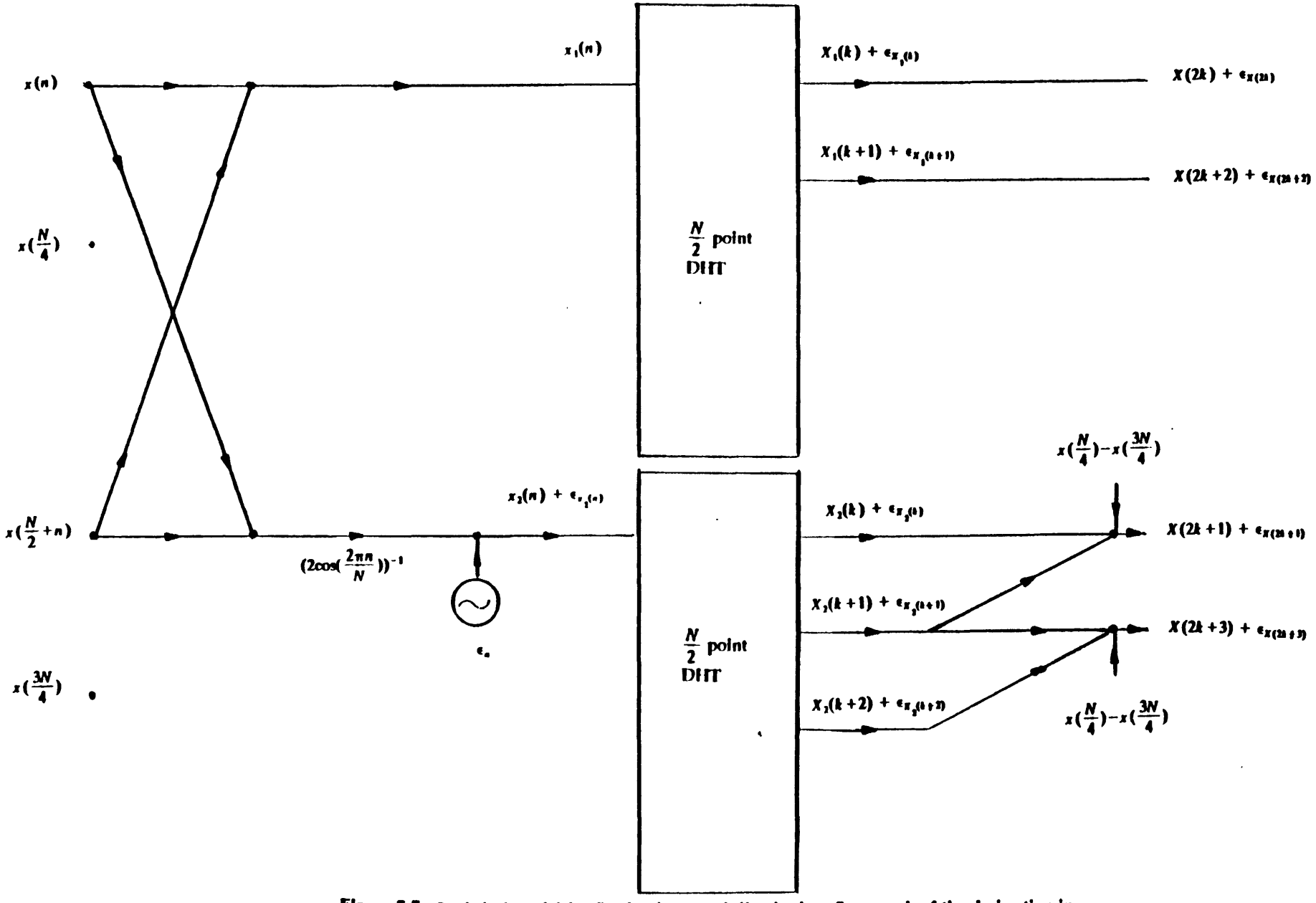


Figure 7.7 Statistical model for fixed-point roundoff noise in a flow graph of the decimation-in-frequency decomposition of an N-point DHT computation into two N/2-point DHT computations using the DF2 algorithm

$$x_2(n) = \begin{cases} \dfrac{x(n) - x(\frac{N}{2} + n)}{2\cos(\frac{2\pi n}{N})} & 0 \le n < \frac{N}{2}, n \neq \frac{N}{4} \\ 0 & n = \frac{N}{4} \end{cases} \qquad (7.9b)$$

Since additions in fixed-point arithmetic are error free, no error is occurred in the process of forming $x_1(n)$. Let $\bar{x}_2(n)$ denote the computed value of $x_2(n)$. Then we get:

$$\bar{x}_2(n) = x_2(n) + \epsilon_n$$

where ϵ_n denotes the roundoff error due to the multiplications in forming the *nth* point of $x_2(n)$. Its variance for most values of n is σ_ϵ^2 which was defined in section 6.1.1. The variance of ϵ_n is not quite the same for all values of n. For example $\epsilon_{\frac{N}{4}}$ is zero because $x_2(\frac{N}{4})$ is defined to be zero in equation (7.9b). Also for $n = 0$ the factor by which $(x(n) - x(n + \frac{N}{2}))$ of equation (7.9b) gets multiplied by becomes $\frac{1}{2}$. As mentioned in earlier sections, the error due to multiplication of fixed-point numbers by $\frac{1}{2}$ has a different probability distribution from what we have assumed in section 6.1.1. In short, the variance of error at $x_2(n)$ can be written as:

$$E[\epsilon_n^2] = \begin{cases} \sigma_\epsilon^2 & 0 < n < \frac{N}{2}, n \neq \frac{N}{4} \\ 1.5\,\sigma_\epsilon^2 & n = 0 \\ 0 & n = \frac{N}{4} \end{cases} \qquad (7.10)$$

On the other hand, $X(k)$, the desired DHT of $x(n)$ is computed from $X_1(k)$ and $X_2(k)$ the transforms of the subsequences $x_1(n)$ and $x_2(n)$ in the following manner:

$$X(2k) = X_1(k) \qquad 0 \le k < \frac{N}{2} \tag{7.11a}$$

$$X(2k+1) = X_2(k) + X_2(k+1) + (-1)^k \left[x(\frac{N}{4}) - x(\frac{3N}{4}) \right] \qquad 0 \le k < \frac{N}{2} \tag{7.11b}$$

where

$$X_2(\frac{N}{2}) = X_2(0)$$

Using the above equation we can find the output noise variance of the N-point sequence $x(n)$ from the output noise variance of the $\frac{N}{2}$-point sequences $x_1(n)$ and $x_2(n)$. More specifically, by inspection of figure 7.7 the error in $X(k)$ denoted by $\epsilon_{X(k)}$ can be written as:

$$\epsilon_{X(2k)} = \epsilon_{X_1(k)} \qquad 0 \le k < \frac{N}{2} \tag{7.12a}$$

$$\epsilon_{X(2k+1)} = \epsilon_{X_2(k)} + \epsilon_{X_2(k+1)} \qquad 0 \le k < \frac{N}{2} \tag{7.12b}$$

where $\epsilon_{X_i(k)}$ denotes the error in computing $X_i(k)$. From (7.9a) we can conclude that

$$Var[\epsilon_{X_1(k)}] = \sigma^2(\frac{N}{2}, k) \tag{7.13a}$$

The error in $X_2(k)$ is due to the error in forming $x_2(n)$ from $x(n)$ and due to the error in $\frac{N}{2}$-point DHT computation of $X_2(k)$. Thus using equation (7.10) the total variance of $\epsilon_{X_2(k)} + \epsilon_{X_2(k+1)}$ can be written as:

$$Var[\epsilon_{X_2(k)} + \epsilon_{X_2(k+1)}] = \sigma^2(\frac{N}{2},k) + \sigma^2(\frac{N}{2},k+1) + \sum_{n=0}^{\frac{N}{2}-1} E[\epsilon_n^2]\,(cas(\frac{2\pi nk}{N/2}) + cas(\frac{2\pi n(k+1)}{N/2}))^2 \quad (7.13b)$$

Note that the summation in equation (7.13b) is the contribution of the error in forming $x_2(n)$ to $\epsilon_{X_2(k)} + \epsilon_{X_2(k+1)}$ and the first two terms in equation (7.13b) are due to the roundoff errors involved in $\frac{N}{2}$-point DHT computation of $X_2(k) + X_2(k+1)$.

Using the equations (7.12) and (7.13) the variance of error at $X(k)$ denoted by $\sigma^2(N,k)$ can be written as:

$$\sigma^2(N,2k) = \sigma^2(\frac{N}{2},k) \qquad 0 \le k < \frac{N}{2} \quad (7.14a)$$

$$\sigma^2(N,2k+1) = \sigma^2(\frac{N}{2},k) + \sigma^2(\frac{N}{2},k+1) + \sum_{n=0}^{\frac{N}{2}-1} E[\epsilon_n^2]\,(cas(\frac{2\pi nk}{N/2}) + cas(\frac{2\pi n(k+1)}{N/2}))^2 \qquad 0 \le k < \frac{N}{2} \quad (7.14b)$$

Equation (7.14) is the basic result we have been looking for. It says that the output noise variance of an N-point sequence, $\sigma^2(N,k)$ can be easily obtained from $\sigma^2(\frac{N}{2},k)$. The distribution of the output error variance of 256-point sequences using the DF2 algorithm is shown in figure 7.8. As it is seen there are no major peaks similar to the ones in figure 7.2 which shows the distribution of output noise variance for the DT2 algorithm. Recall that the peaks at $k = \frac{N}{4} \pm 1, \frac{3N}{4} \pm 1$ for DT2 are due to the fact that in the last stage of the butterflies, the $(\frac{N}{4} \pm 1)st$ and $(\frac{3N}{4} \pm 1)st$ elements of the $(v-1)st$ array are multiplied by the largest coefficient

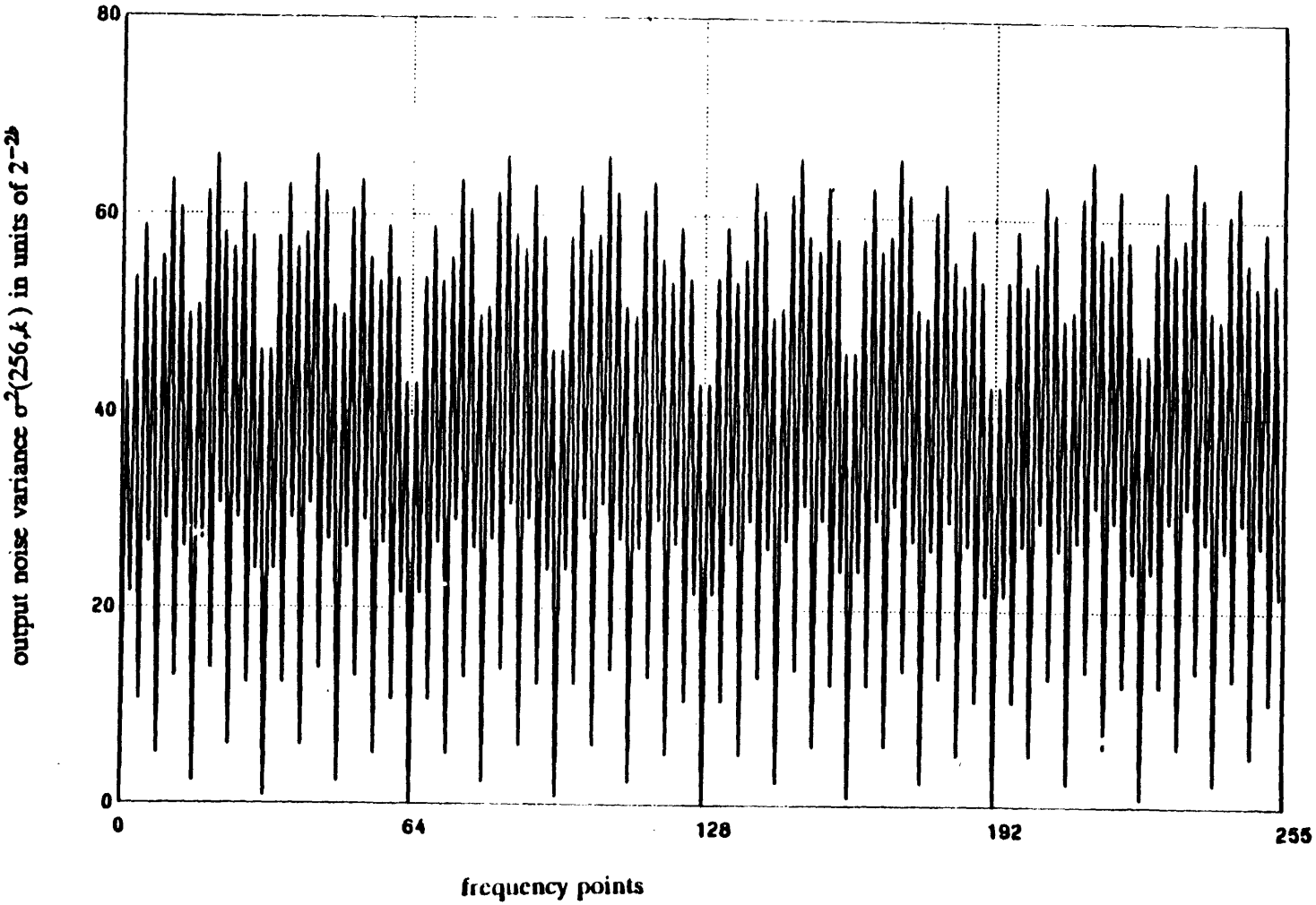


Figure 7.8 Distribution of the output noise variance of 256-point sequences for fixed-point implementation of the DF2 algorithm.

of the entire algorithm (i.e. $\frac{1}{2\sin(\frac{2\pi}{N})}$). In the DF2 algorithm, since multiplications by the largest coefficient (i.e. $\frac{1}{2\sin(\frac{2\pi}{N})}$) appear in the first stage of the algorithm, their effect is somewhat reduced. Besides, since the input to the first stage of the algorithm is error free, multiplication by large factors can not magnify any error. Figure 7.9 shows the output noise variance of the DT2 and DF2 algorithms as a function of the transform size.

Having found the output noise variance, we should now consider the dynamic range issues. The maximum allowable input to ensure against overflow was found numerically using the technique discussed in appendix A. The second column of table 7.2 shows the maximum input as a function of transform size. For large values of N, doubling the transform size scales down the maximum magnitude of the input by a factor of 3.46. As it is mentioned in appendix A, the numerical technique used to generate table 7.2, results in the sufficient but not the necessary condition for the input to prevent overflows. The third column of the table 7.2 shows the variance of the output signal provided the input is a white sequence with probability density function of each point uniformly distributed between its maximum allowable range. Comparing tables 7.1 and 7.2 we can see that the maximum magnitude of the input which guarantees no overflows, is larger for the decimation-in-frequency algorithm than it is for the decimation-in-time algorithm. Figure 7.10 shows the average noise to signal ratio for the DF2 and DT2 algo-

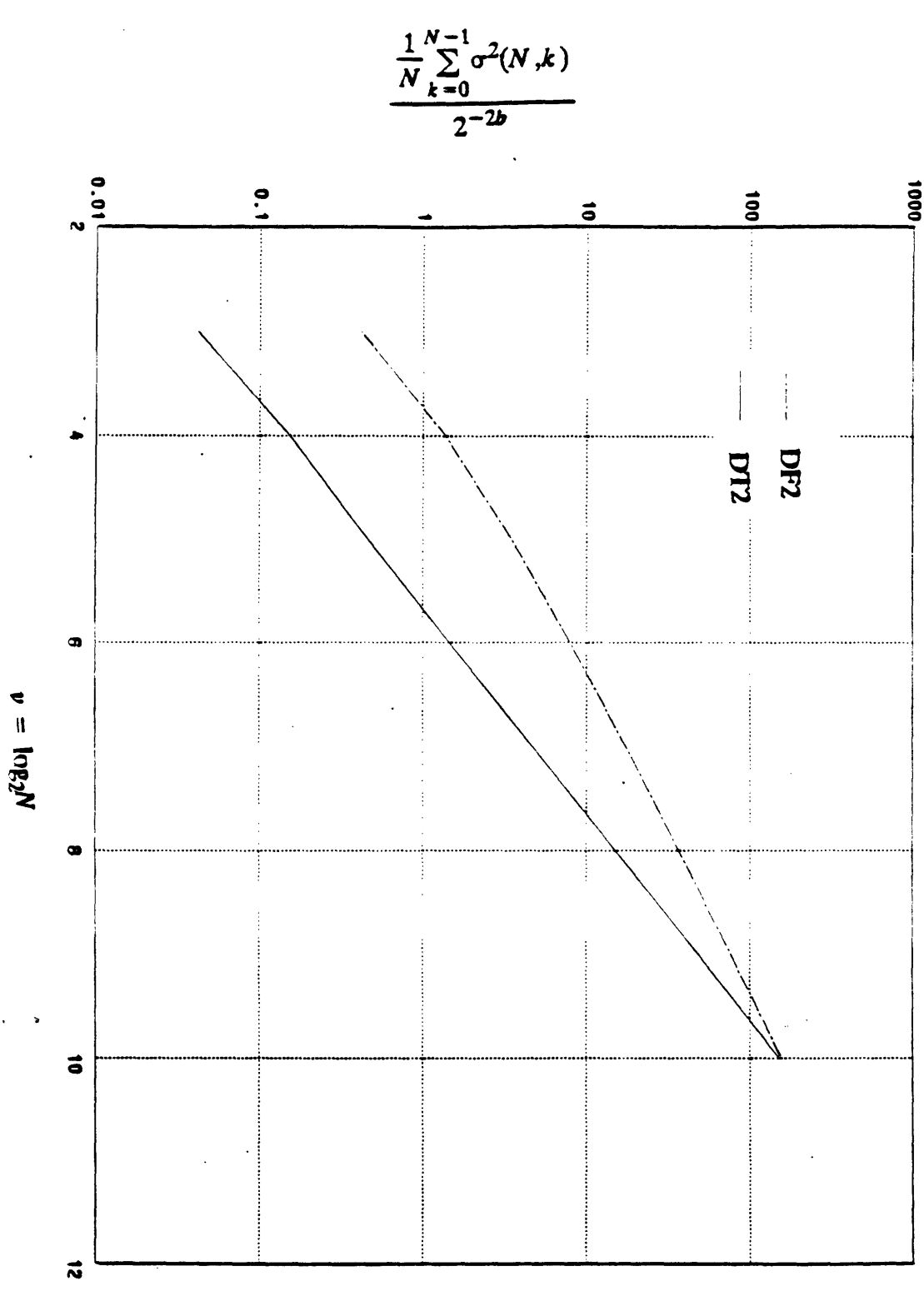


Figure 7.9 Output noise variance for fixed-point realization of DT2 and DF2 algorithms

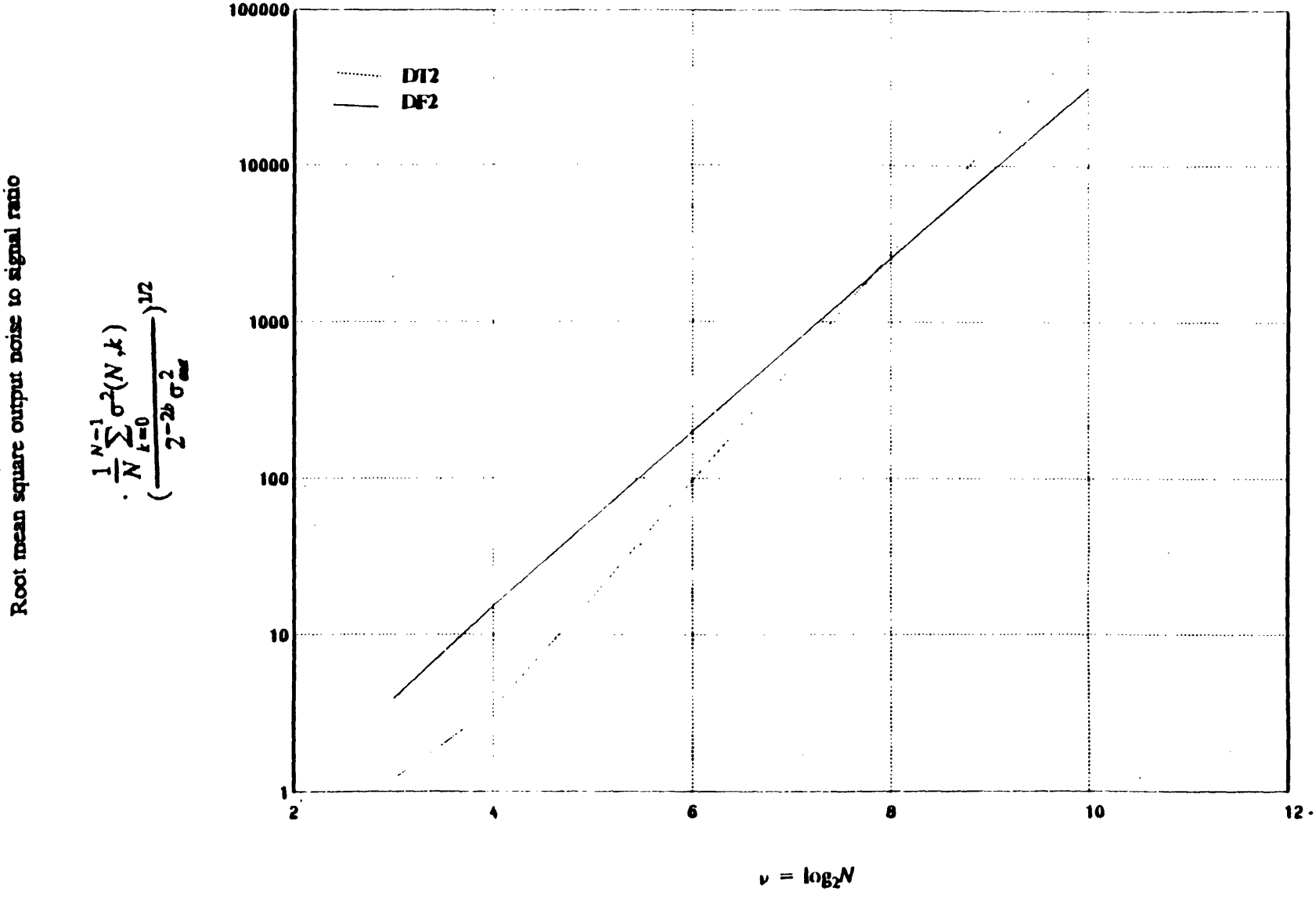


Figure 7.10 Theoretical noise to signal ratio for fixed-point realization of DF2 and DT2

Transform size N	Maximum input $\lvert X_{in} \rvert_{max}$	Variance of output $\frac{N \lvert X_{in} \rvert^2_{max}}{3}$
8	1.0×10^{-1}	2.9×10^{-2}
16	3.3×10^{-2}	5.9×10^{-3}
32	1.0×10^{-2}	1.2×10^{-3}
64	3.1×10^{-3}	2.0×10^{-4}
128	8.9×10^{-4}	3.4×10^{-5}
256	2.7×10^{-4}	6.3×10^{-6}
512	7.4×10^{-5}	9.5×10^{-7}
1024	2.2×10^{-5}	1.6×10^{-7}

Table 7.2 The upper bound on maximum magnitude of the input to ensure against overflow in fixed point implementation of the DF2 algorithm.

rithms. The experimental results verifying the theoretical curves of figures 7.8 and 7.10 will be presented in chapter 8.

7.2.2. Roundoff Noise Analysis of the DF2 Using Floating-Point Arithmetic

Our approach to roundoff noise analysis of the DF2 algorithm using floating-point arithmetic is a recursive one. However, as we will see, it will be slightly different from the previous cases.

Consider an $\frac{N}{2}$-point zero mean sequence $w(n)$ with all its points statistically independent with variance of the *nth* point denoted by $\sigma'^{2}_{in}(\frac{N}{2},n)$. Suppose we have an algorithm which given $\sigma'^{2}_{in}(\frac{N}{2},n)$ computes the variance of error at $W(k)$ the DHT of $w(n)$. Consider another N-point zero mean sequence $x(n)$ with all its points statistically independent and the variance of its *nth* point given by $\sigma^{2}_{in}(N,n)$. We will show that given an algorithm to compute the variance of error for $W(k)$, we can find a way to compute the variance of error for $X(k)$ the DHT of $x(n)$.

Recall from section 5.1.4 that using the decimation-in-frequency algorithm of section 5.1.4 an N-point DHT can be decomposed into two $\frac{N}{2}$-point DHTs. This is shown in figure 7.11 and equation (7.9). where the N-point sequence $x(n)$ is decomposed into $x_1(n)$ and $x_2(n)$. Using equation (7.12) and figure 7.11 $X_1(k)$ and $X_2(k)$, the DHTs of $x_1(n)$ and $x_2(n)$ are combined in order to form $X(k)$. Having inserted the signal independent noise generators after each multiplier and

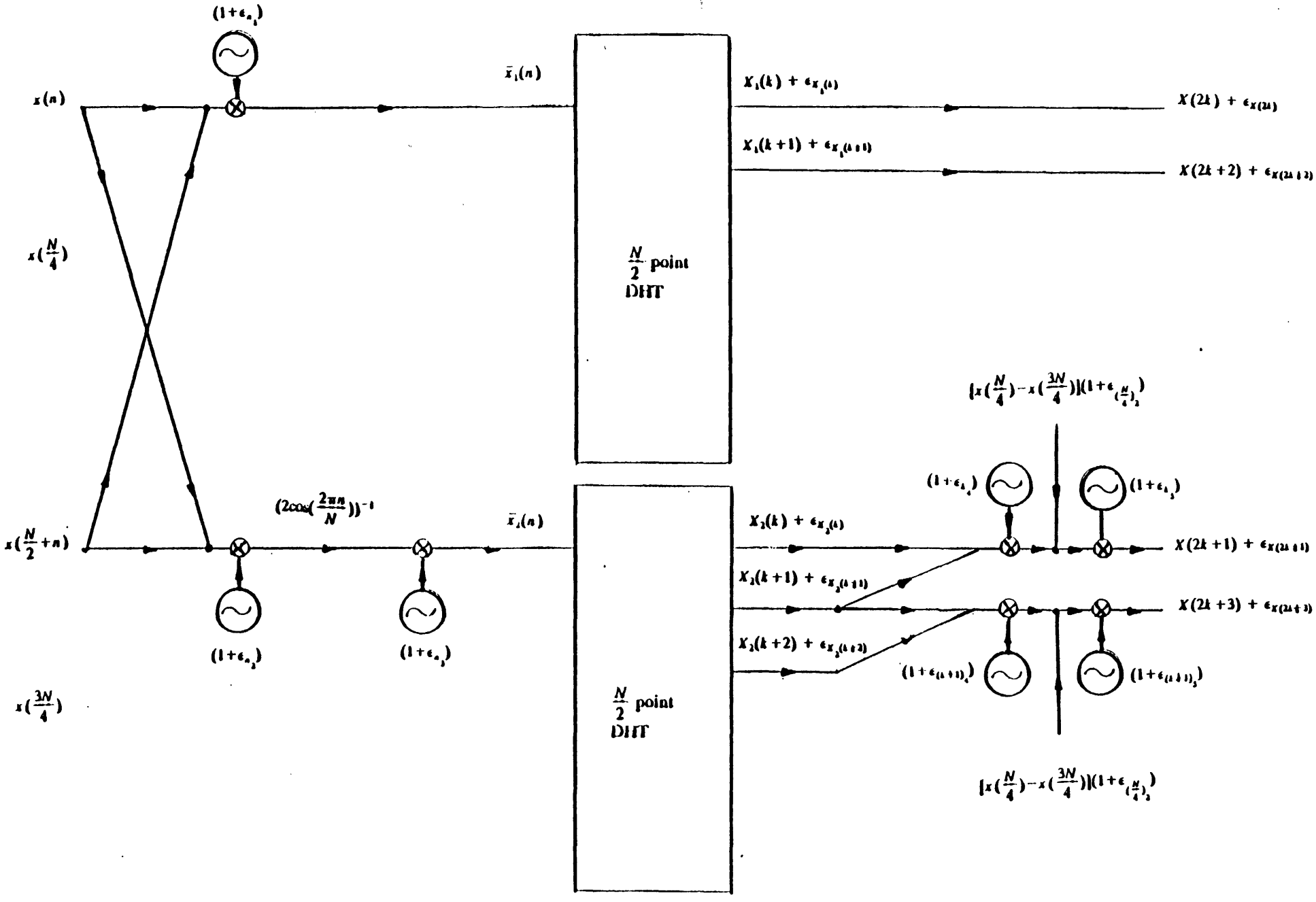


Figure 7.11 Statistical model for floating-point roundoff noise in a flow graph of the decimation-in-frequency decomposition of an N-point DHT computation into two N/2-point DHT computations using the DF2 algorithm

adder in the signal flow graph of the algorithm, the computed values of $x_1(n)$ and $x_2(n)$ denoted by $\bar{x}_1(n)$ and $\bar{x}_2(n)$ can be written as:

$$\bar{x}_1(n) = [\, x(n) + x(\frac{N}{2}+n)\,][\,1 + \epsilon_{n_1}\,] \tag{7.15a}$$

$$\bar{x}_2(n) = \begin{cases} \dfrac{x(n) - x(\frac{N}{2}+n)}{2\cos(\frac{2\pi n}{N})}\,[\,1+\epsilon_{n_2}\,][\,1+\epsilon_{n_3}\,] & 0 \le n < \frac{N}{2}\,,\ n \neq \frac{N}{4} \\ 0 & n = \frac{N}{4} \end{cases} \tag{7.15b}$$

where ϵ_{n_i} denotes the roundoff error due to floating-point multiplication and additions for computing the *nth* term of $x_1(n)$ or $x_2(n)$.

The error in computing various points of $x_1(n)$ are uncorrelated with each other and the variance of the error at $\bar{x}_1(n)$ is given by

$$Var\ [\,\epsilon_{x_1(n)}\,] = \sigma_\epsilon^2\,[\,\sigma_{in}^2(N,n) + \sigma_{in}^2(N,n+\frac{N}{2})\,]$$

If the $\frac{N}{2}$-point DHT of $x_1(n)$ were to be computed with infinite precision, the only source of error at $X_1(k)$ would have been due to the error in $x_1(n)$. That is

$$Var'[\epsilon_{X_1(k)} = \sigma_\epsilon^2 \sum_{n=0}^{\frac{N}{2}-1} [\,\sigma_{in}^2(N,n) + \sigma_{in}^2(N,n+\frac{N}{2})\,]\, cas^2(\frac{2\pi nk}{N/2}) \tag{7.16}$$

Recall that $x_1(n)$ is an $\frac{N}{2}$-point zero mean sequence with all its points statistically independent and the variance of its *nth* point given by $\sigma_{in}^2(N,n) + \sigma_{in}^2(N,n+\frac{N}{2})$. Therefore we can use our assumed algorithm in order to find $Var''(\epsilon_{X_1(k)})$ the variance of error at $X_1(k)$ only due to the $\frac{N}{2}$-point DHT computational errors . Thus the total variance of error at $X_1(k)$ is given by

$$Var[\epsilon_{X_1(k)}] = Var''[\epsilon_{X_1(k)}] + Var'[\epsilon_{X_1(k)}] \tag{7.17a}$$

A similar argument can be used to show that the variance of error at $X_2(k)$ is given by

$$Var[\epsilon_{X_2(k)}] = Var''[\epsilon_{X_2(k)}] + Var'[\epsilon_{X_2(k)}] \tag{7.17b}$$

where

$$Var'[\epsilon_{X_2(k)}] = \sigma_t^2 \sum_{n=0}^{\frac{N}{2}-1} \frac{\sigma_{in}^2(N,n) + \sigma_{in}^2(N,n+\frac{N}{2})}{2\cos^2(\frac{2\pi n}{N})} \cos^2(\frac{2\pi nk}{N/2})$$

is the variance of error at $X_2(k)$ only due to the error in forming $x_2(n)$ and $Var''[\epsilon_{X_2(k)}]$ is the variance of error at $X_2(k)$ only due to the $\frac{N}{2}$-point DHT computation of $X_2(k)$. $Var''[\epsilon_{X_2(k)}]$ can be obtained via our assumed algorithm which uses the variance of $x_2(n)$ (i.e $\frac{\sigma_{in}^2(N,n) + \sigma_{in}^2(N,n+\frac{N}{2})}{4\cos(\frac{2\pi n}{N})}$) as input.

Now let us see how the quantities that we have found can be used to find the output noise variance for $x(n)$. Referring to figure 7.11, the computed value of $X(2k)$ and $X(2k+1)$ can be written as:

$$\bar{X}(2k) = X_1(k) + \epsilon_{X_1(k)} \tag{7.18a}$$

$$\bar{X}(2k+1) = [1 + \epsilon_{k_5}] \left\{ [(X_2(k) + \epsilon_{X_2(k)}) + (X_2(k+1) + \epsilon_{X_2(k+1)})][1 + \epsilon_{k_4}] + \right. \tag{7.18b}$$

$$\left. (-1)^k [x(\frac{N}{4}) - x(\frac{3N}{4})](1 + \epsilon_{(\frac{N}{4})_2}) \right\}$$

where ϵ_{k_7} denotes the error due to floating-point multiplication and addition for computing $X(2k+1)$ and $\epsilon_{X_1(k)}$ is the error which has been accumulated in computing

$X_i(k)$. From equation (7.18) the variance of error at the kth point of the output of the N-point DHT denoted by $\sigma^2(N,k)$ can be written as:

$$\sigma^2(N,2k) = Var\ [\ \epsilon_{X_1(k)}\ ] \tag{7.19a}$$

$$\sigma^2(N,2k+1) = E\ [\ (\ \epsilon_{X_2(k)} + \epsilon_{X_2(k+1)}\ )^2\ ] + \sigma_t^2 \left\{ 2\,E[\ (\ X_2(k) + X_2(k+1)\ )^2\ ] + \right. \tag{7.19b}$$

$$\left. E[\ (\ X(2k+1)\ )^2\ ] + \sigma_{in}^2(N,\frac{N}{4}) + \sigma_{in}^2(N,\frac{3N}{4}) \right\} .$$

Note that $\sigma^2(N,2k)$ of equation (7.19a) has already been computed in equation (7.17a). We now have to show ways of obtaining the terms in equation (7.19b).

The first term in equation (7.19b) is given by

$$E\ [\ (\ \epsilon_{X_2(k)} + \epsilon_{X_2(k+1)}\ )^2\ ] = Var[\epsilon_{X_2(k)}] + Var[\epsilon_{X_2(k+1)}] +$$

$$2\,\sigma_t^2 \sum_{\substack{n=0 \\ n\neq\frac{N}{4}}}^{\frac{N}{2}-1} \frac{\sigma_{in}^2(N,n) + \sigma_{in}^2(N,n+\frac{N}{2})}{2\cos^2(\frac{2\pi n}{N})}\, cas(\frac{2\pi nk}{N/2})\, cas(\frac{2\pi n(k+1)}{N/2})$$

The summation in the above equation is due to the error in computing $x_2(n)$ which causes some correlation between the error at the output points $X_2(k)$ and $X_2(k+1)$. Similarly, $Var[\ \epsilon_{X_2(k)}\ ]$ and $Var[\ \epsilon_{X_2(k+1)}\ ]$ of the above equation can be found from equation (7.17b).

The second term in equation (7.19b) involves computing $E[\ (\ X(2k+1)\ )^2]$ and $E[(\ X_2(k) + X_2(k+1)\ )^2]$ which are given by

$$E\ [\ (\ X(2k+1)\ )^2\ ] = \sum_{n=0}^{N-1} \sigma_{in}^2(N,n)\, cas^2(\frac{2\pi nk}{N})$$

$$E\left[\left(X_2(k)+X_2(k+1)\right)^2\right]=\sum_{n=0,\,n\neq\frac{N}{4}}^{\frac{N}{2}-1}\frac{\sigma_{in}^2(N,n)+\sigma_{in}^2(N,\frac{N}{2}+n)}{4\cos^2(\frac{2\pi n}{N})}\left[cas(\frac{2\pi nk}{N/2})+cas(\frac{2\pi n(k+1)}{N/2})\right]^2$$

The recursion is now complete. We have shown that all the terms in equations (7.19a) and (7.19b) can be computed using our assumed algorithm.

To summarize, we started off with an algorithm which could compute the output noise variance of an $\frac{N}{2}$-point zero mean sequence of uncorrelated input points. Then we showed that the using this algorithm we can find the output noise variance of an N-point sequence with uncorrelated input points. Thus we can recursively find the output noise variance of arbitrarily long white sequences.

Figure 7.12 shows the distribution of the variance of error as a function of frequency for 256-point white sequences. Figure 7.13 shows the average output noise to signal ratio of white sequences using the DF2 and DT2 algorithms. As it is expected the average output noise to signal variance for the decimation-in-frequency algorithm is roughly the same as the decimation-in-time version. Comparing figures 7.13, 7.6 and 6.22 we can conclude that overall Bracewell's algorithm has more desirable error properties than the new algorithm. The experimental results verifying the theoretical predictions of this section will be presented in chapter 8.

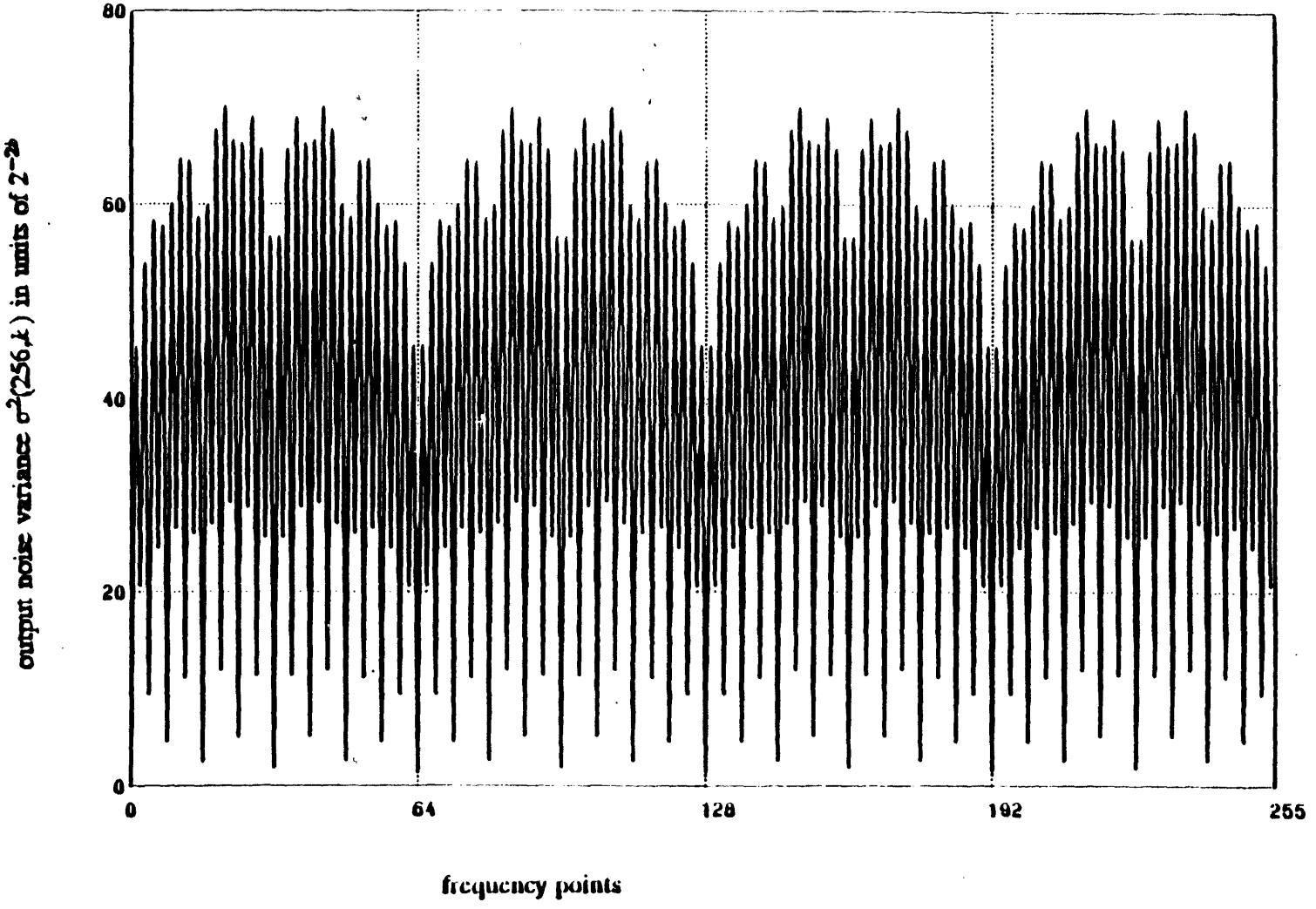


Figure 7.12 Distribution of the output noise variance of 256-point white sequences for floating-point implementation of the DF2 algorithm.

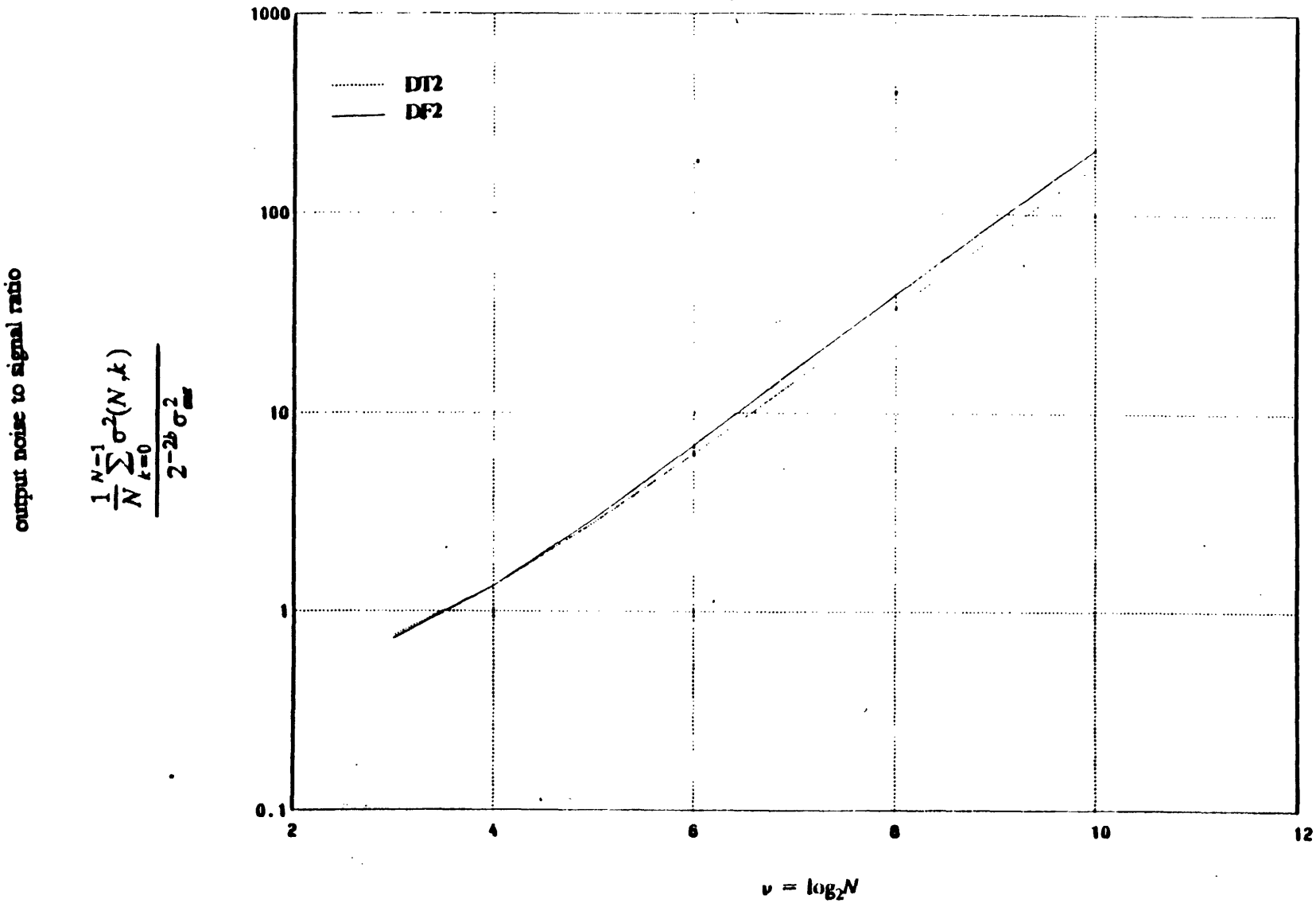


Figure 7.13 Theoretical noise to signal ratio for floating-point realization of DF2 and DT2

CHAPTER 8: Experimental Results

In chapters 6 and 7, we investigated the theoretical error properties of some of the algorithms in chapters 3, 4 and 5. In this chapter, the experimental results regarding the roundoff noise in these algorithms will be examined. Section 8.1 contains the detailed description of the experimental procedure which was used. Sections 8.2 and 8.3 will report on the experimental error properties of the DHT algorithms. In chapter 9 we will compare various algorithms discussed throughout the thesis in terms of their computational efficiencies and error properties.

8.1. The Experimental Procedure

The experiments for roundoff noise analysis consisted of two parts; In the first part, zero mean white input sequences were generated using a random number generator routine. The probability density function (pdf) for each point of these sequences was uniformly distributed around the origin. Clearly, the width of this pdf does not affect the noise to signal ratio for floating-point implementations. However, in the case of fixed-point arithmetic, the width was chosen in such a way to guarantee no overflows in the output or in intermediate computations.

Having generated the test sequences, they were then transformed three times; once using rounded fixed-point arithmetic with word length of 15 bits (excluding the sign bit); the second time using floating-point arithmetic with 23 bits of mantissa (excluding the hidden bit and the sign bit) and the third time using double precision floating-point arithmetic with 55 bits of mantissa (excluding the sign

bit and the hidden bit). The double precision computation was assumed to be exact in comparison with the other two. Thus the roundoff error due to fixed-point implementation is the difference between the third and first computations and the error due to floating-point implementation is the difference between the third and second computations.

The above procedure was repeated with 1000 independent input sequences in order to find a stable estimate of the variance of error for each frequency point of an N-point transform. The estimator used is of the form

$$\hat{\sigma}^2(N,k) = \frac{1}{1000}\sum_{i=1}^{1000}[\, e_i(N,k) - \hat{m}_e(N,k)\,]^2 \tag{8.1a}$$

$$\hat{m}_e(N,k) = \frac{1}{1000}\sum_{i=0}^{1000} e_i(N,k) \tag{8.1b}$$

where $\hat{m}_e(N,k)$ and $\hat{\sigma}^2(N,k)$ denote the estimated mean and variance of error in the *kth* point of an N-point transform. Similarly, $e_i(N,k)$ of equation (8.1) denotes the *ith* experimental value of the error at the *kth* frequency point of an N-point transform. The reason behind choosing 1000 as the number of experiments and the characteristics of the estimator shown in equation (8.1) are discussed in appendix B.

In order to find an estimate of the mean output noise variance we have to average $\hat{\sigma}^2(N,k)$ over the frequency points k . That is

$$\hat{\sigma}_E^2 = \frac{1}{N}\sum_{k=0}^{N-1}\hat{\sigma}^2(N,k) \tag{8.2}$$

Since the input signal is zero mean and white and its pdf is uniformly distributed we can easily find the output signal variance. Thus using equation (8.2) we can

obtain an estimate of noise to signal ratio for the algorithms under investigation.

Recall from section 6.1.2 that in our theoretical analyses we used the parameter σ_ϵ^2 to denote the variance of error due to floating-point multiplications and additions i.e

$$Q(xy) = xy\ (1 + \epsilon)$$

$$Q(x + y) = (x + y)(1 + \epsilon)$$

$$Var[\ \epsilon\ ] = \sigma_\epsilon^2$$

For floating-point implementation with b bits of mantissa, if we assume that ϵ is uniformly distributed in the range $(-2^{-b}, +2^{-b})$ then $\sigma_\epsilon^2 = \frac{1}{3}2^{-2b}$. Experiments have shown that the variance of error due to multiplications and additions are slightly different from each other and that the distribution for ϵ is not quite uniform [13]. (Note that in fixed-point implementation, uniform distribution for ϵ in the range $(-\frac{1}{2}2^{-b}, +\frac{1}{2}2^{-b})$ has been verified experimentally [13]). However, the variance of roundoff error in floating-point arithmetic has been verified to be proportional to 2^{-2b} [13]. That is

$$\sigma_\epsilon^2 = \alpha\ 2^{-2b}$$

where α is a constant for a given algorithm which depends on the number of multiplies and adds and the order in which they are performed in that algorithm. Therefore, for the results related to floating-point implementations, we will use an empirical value for α obtained by matching the theoretical and experimental noise to signal ratio curves.

To clarify the experimental procedure used, we should define carefully the convention used to round the results of floating-point additions and multiplications. The results were rounded to the closest binary number (for a b-bit mantissa) and if a result of an addition or a multiplication was midway between two binary numbers, a random choice was made as to whether to round up or down. It turns out that in floating-point addition of two numbers of the same order of magnitude where the unrounded mantissas are very often only one bit longer than the rounded mantissas, this situation occurs quite frequently. Always rounding up (or down) rather than randomly up or down in this situation introduces a correlation between roundoff error and signal sign. This contradicts the assumption that round-off errors are signal independent. For instance, as we will see, in implementing the algorithms of chapter 3, if one merely rounds up in situations where the mantissa is exactly between two binary numbers, the experimental noise to signal ratios will be significantly higher than the theoretical predictions.

8.2. Experimental Results

8.2.1. Experimental Verification of the Theoretical Results

8.2.1.1. Fixed-point Implementation Results

In this section the experimental results verifying the theoretical analyses of chapter 6 will be presented. Figure 8.1 shows the experimental and the theoretical noise to signal ratio for fixed-point realization of the DT1 and DF1 algorithms. There seems to be an excellent agreement between the predicted and actual values

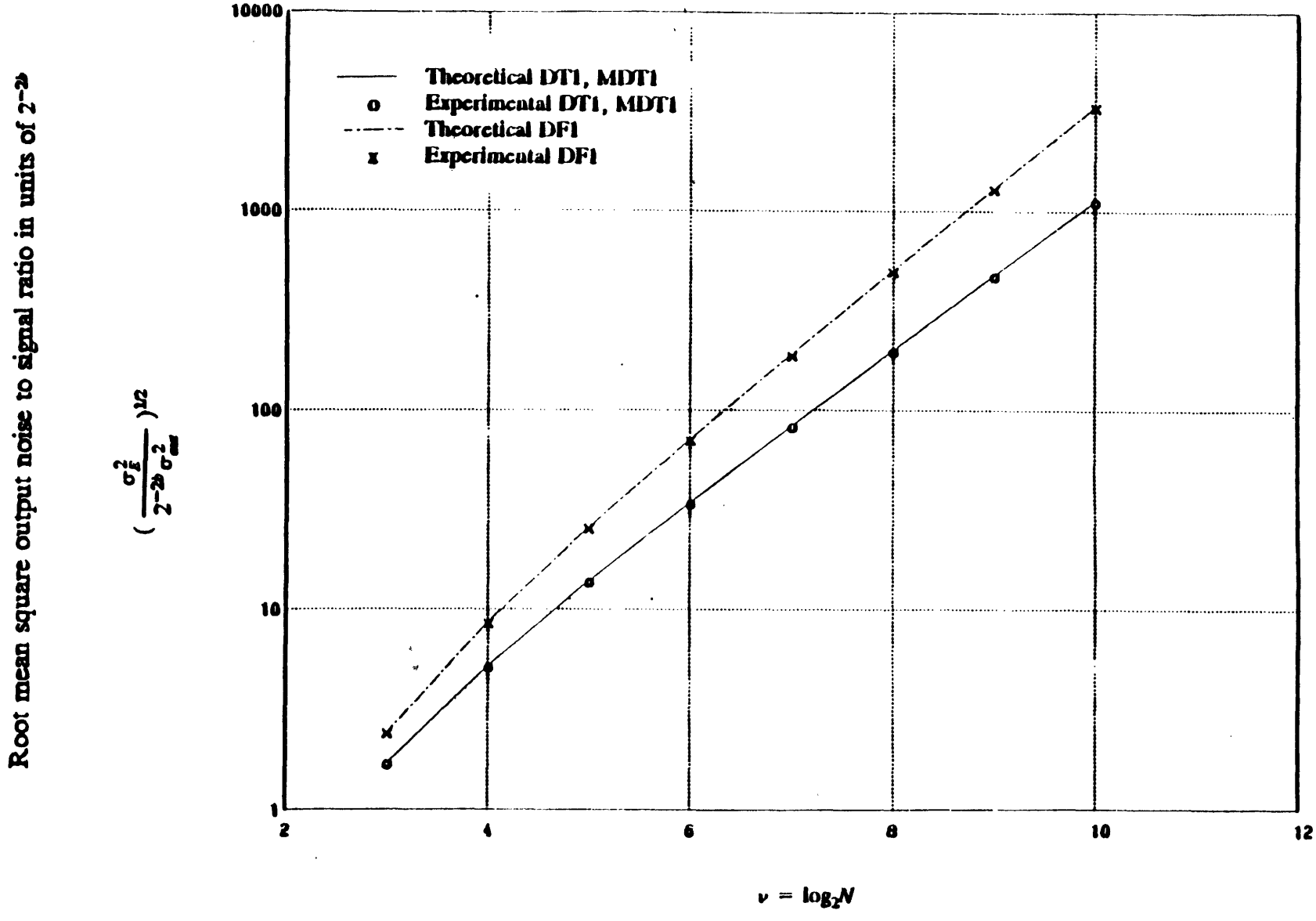


Figure 8.1 Output noise to signal ratio for fixed-point realization of DT1 and DF1 algorithms

of noise to signal ratio. Equations (6.15) and (6.37) were used in order to find the theoretical mean output noise variance given by

$$\sigma_E^2 = <\sigma^2(N,k)> = \frac{1}{N}\sum_{k=0}^{N-1}\sigma^2(N,k)$$

The output signal variance σ_{out}^2 was obtained by assuming that the input signal is white and that the pdf for each point is uniformly distributed around the origin. The width of this pdf was determined by the maximum magnitude of the input that would guarantee no overflows for the particular algorithm under investigation. The upper bound for the maximum magnitude of input for DT1 and DF1 are shown in tables 6.1 and 6.2 respectively. Not only is the output noise variance for the DF1 algorithm larger than that of the DT1 algorithm (see figure 6.18), but also the dynamic range constraint on the input signal is more severe for the DF1 algorithm. This explains the gap between the two curves shown in figure 8.1. We can fit the following equations to the data shown in figure 8.1:

$$\sqrt{\frac{\sigma_E^2}{\sigma_{out}^2\, 2^{-2b}}} = 0.20\, N^{1.25} \qquad DT1 \qquad (8.3a)$$

$$\sqrt{\frac{\sigma_E^2}{\sigma_{out}^2\, 2^{-2b}}} = 0.24\, N^{1.38} \qquad DF1 \qquad (8.3b)$$

Although the multiplication count for the MDT1 algorithm is $\frac{2}{3}$ of DT1, as it is shown in figure 8.1, the noise to signal ratio for the two algorithms are almost identical. As explained in section 6.2.2.1, this is due to the fact that the errors at the input to a given butterfly in the MDT1 algorithm are correlated with each other.

Figure 8.2 shows the output noise to signal ratio for the DT2 and DF2 algorithms. Again, There is excellent agreement between the predicted and actual values. Although the output noise variance for DT2 and DF2 are fairly close to each other, comparing tables 7.1 and 7.2 we realize that the overflow constraints, allow a larger output signal variance for the decimation-in-frequency algorithm. Indeed, this is one of the reasons why the noise to signal ratio of the DF2 algorithm is increasing at a lower than that of the DT2 algorithm in figure 8.2. We can fit the following equations to the data shown in figure 8.2:

$$\sqrt{\frac{\sigma_{\hat{e}}^2}{\sigma_{out}^2\, 2^{-2b}}} = 0.0057\ N^{2.36} \qquad DT2 \qquad (8.4a)$$

$$\sqrt{\frac{\sigma_{\hat{e}}^2}{\sigma_{out}^2\, 2^{-2b}}} = 0.10\ N^{1.8} \qquad DT2 \qquad (8.4b)$$

Figures 8.3 through 8.7 show the theoretical and experimental distribution of output noise variance as a function of frequency for 256-point sequences. Again the theoretical results seem to be in good agreement with the experiments. As mentioned in chapter 7, the peaks in figure 8.6 at frequency points $k = \frac{N}{4} \pm 1, \frac{3N}{4} \pm 1$ are due to the fact that in the last stage of the algorithm, the error which has been accumulated in the $(\nu-1)st$ array is multiplied by $\frac{1}{2\cos(\frac{2\pi k}{N})}$ which attains its highest value (i.e. $\frac{1}{2\sin(\frac{2\pi}{N})}$) at $k = \frac{N}{4} \pm 1, \frac{3N}{4} \pm 1$. This highest value $\frac{1}{2\sin(\frac{2\pi}{N})}$ is also the largest coefficient of all the butterflies in the entire algorithm.

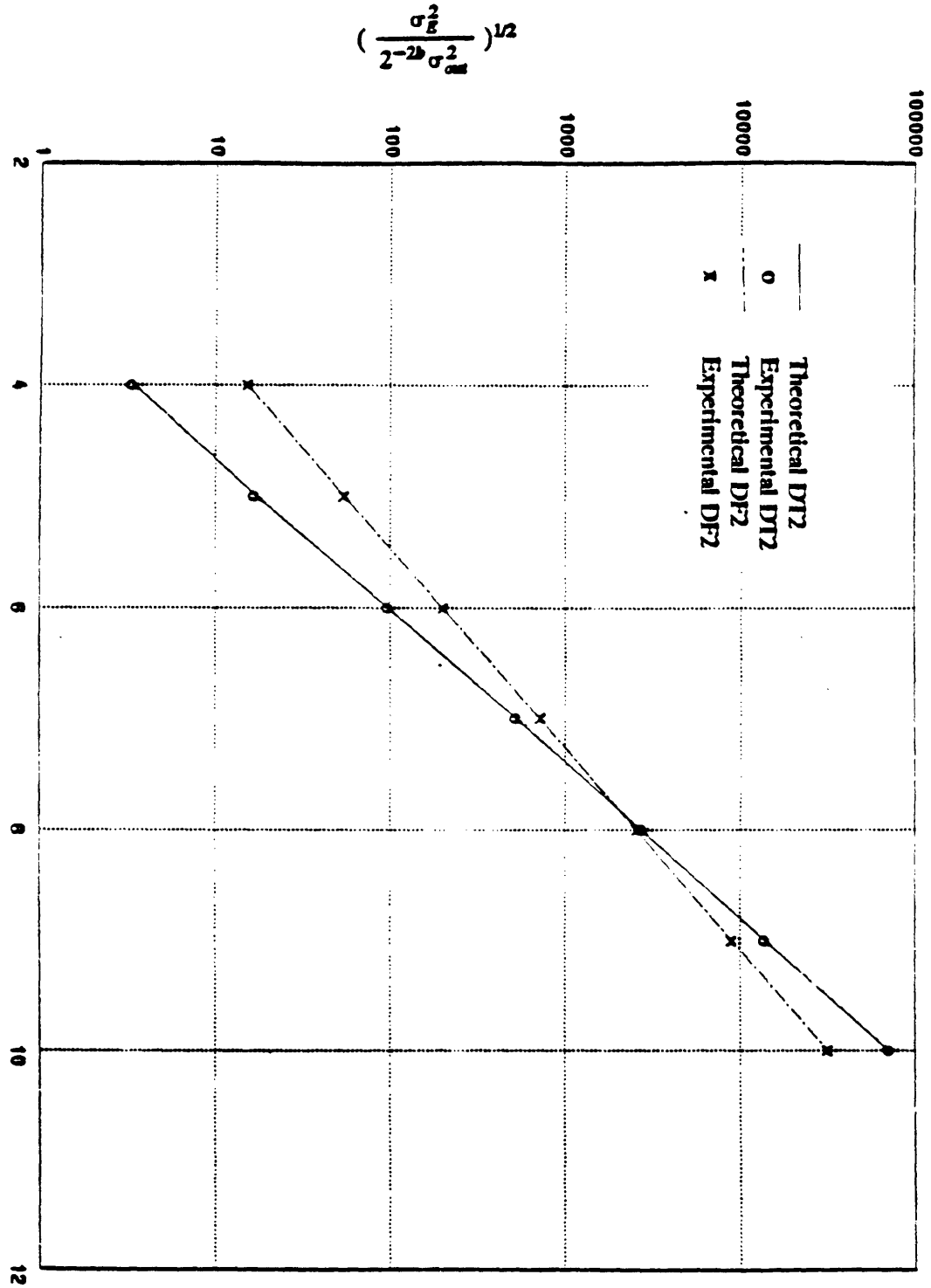


Figure 8.2 Output noise to signal ratio for fixed-point realization of DT2 and DF2 algorithms

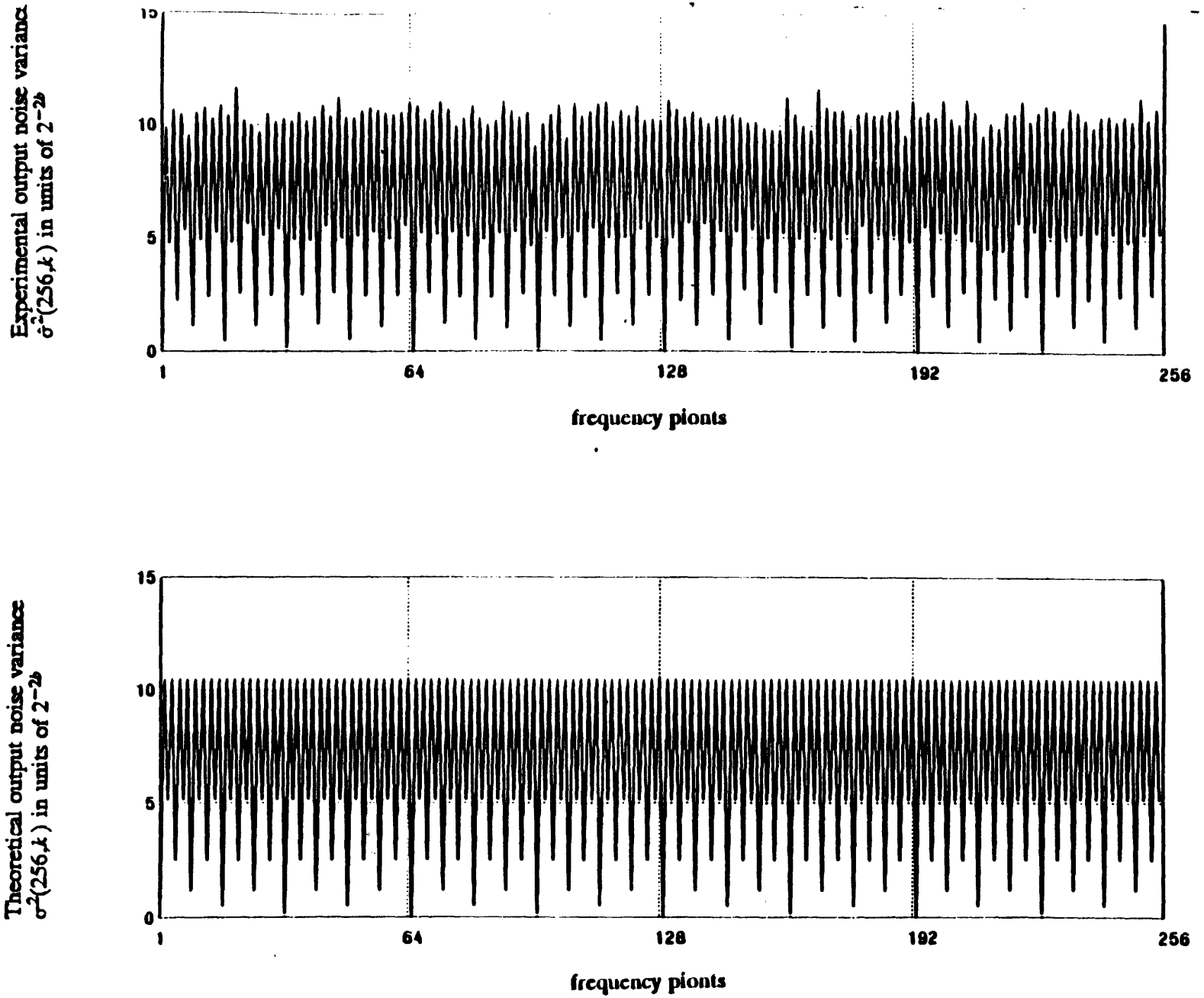


Figure 8.3 Distribution of the output noise variance for 256-point sequences for fixed-point realization of DT1: (a) experimental; (b) theoretical.

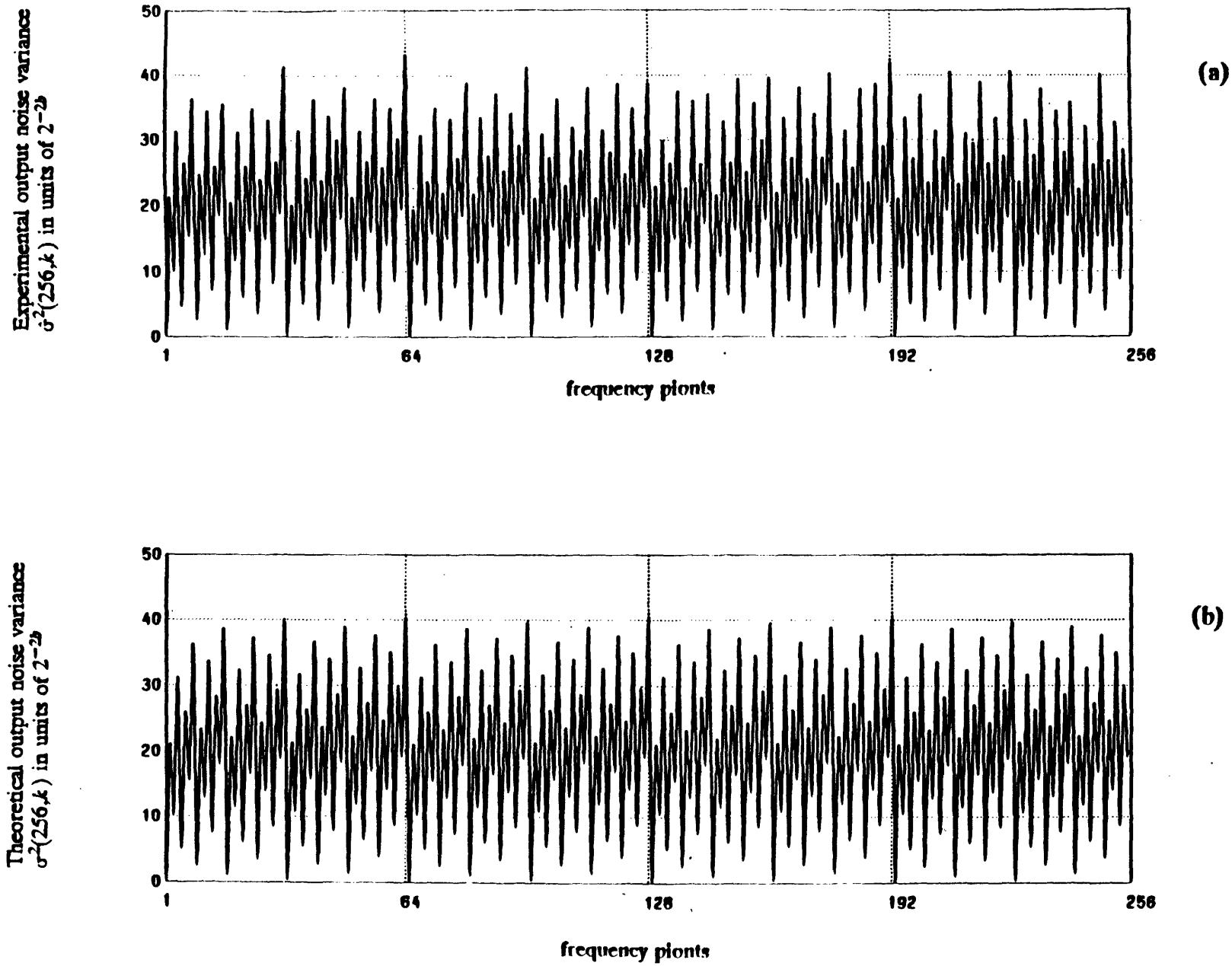


Figure 8.4 Distribution of the output noise variance for 256-point sequences for fixed-point realization of MDT1: (a) experimental; (b) theoretical.

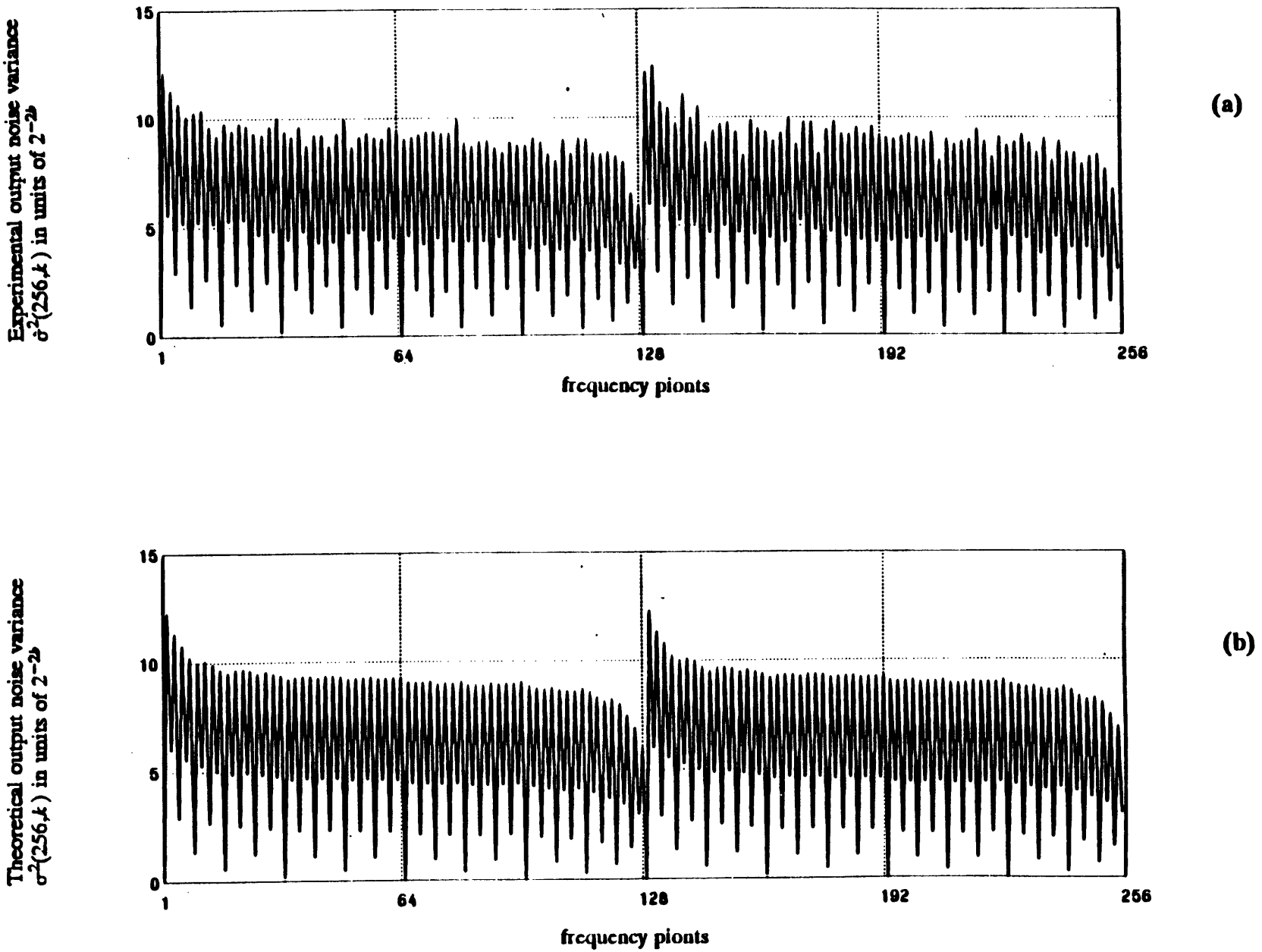
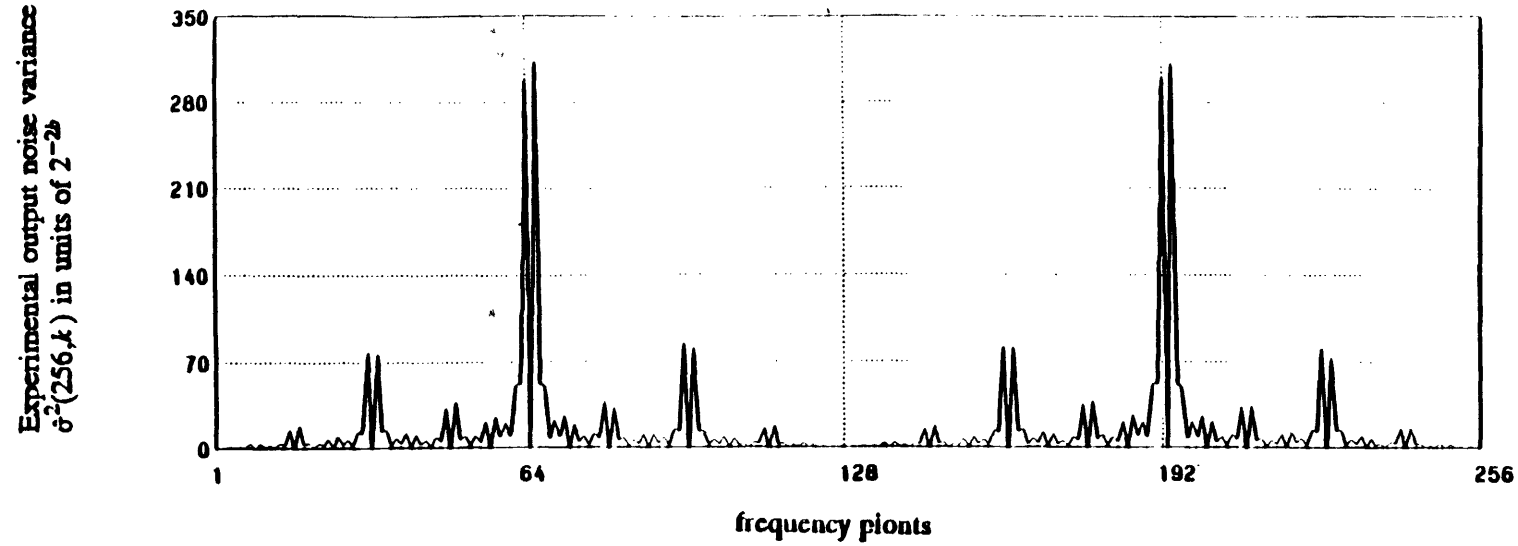


Figure 8.5 Distribution of the output noise variance for 256-point sequences for fixed-point realization of DF1: (a) experimental; (b) theoretical.



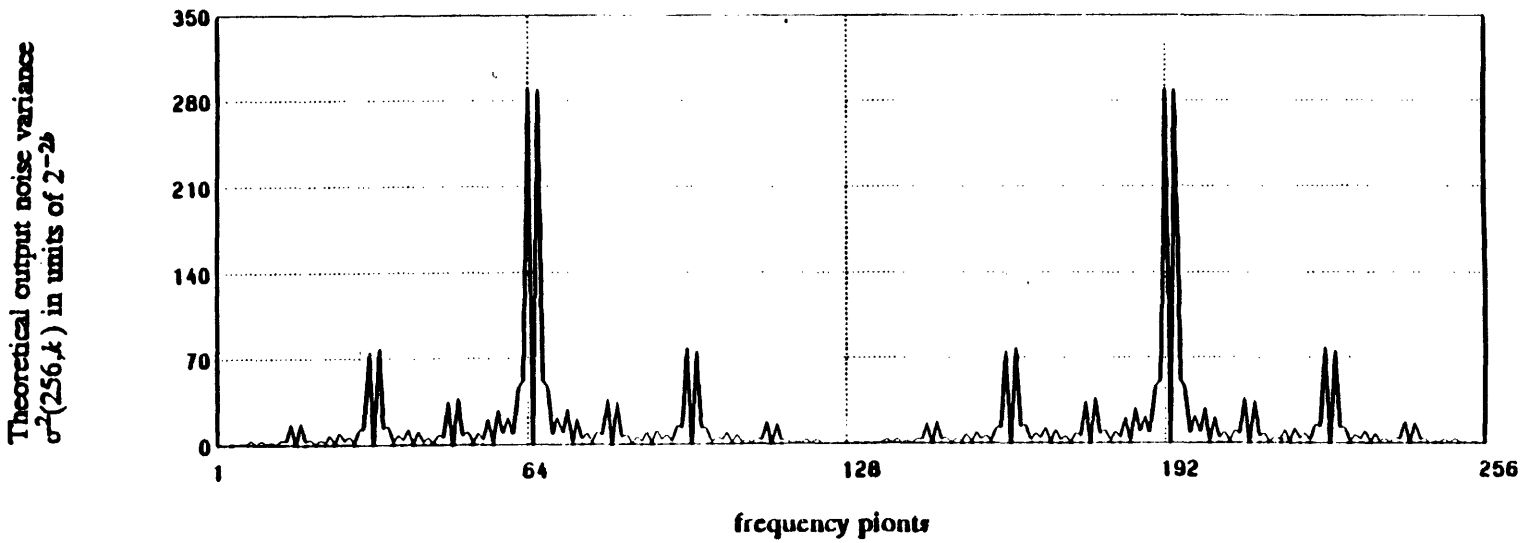


Figure 8.6 Distribution of the output noise variance for 256-point sequences for fixed-point realization of DT2: (a) experimental; (b) theoretical.

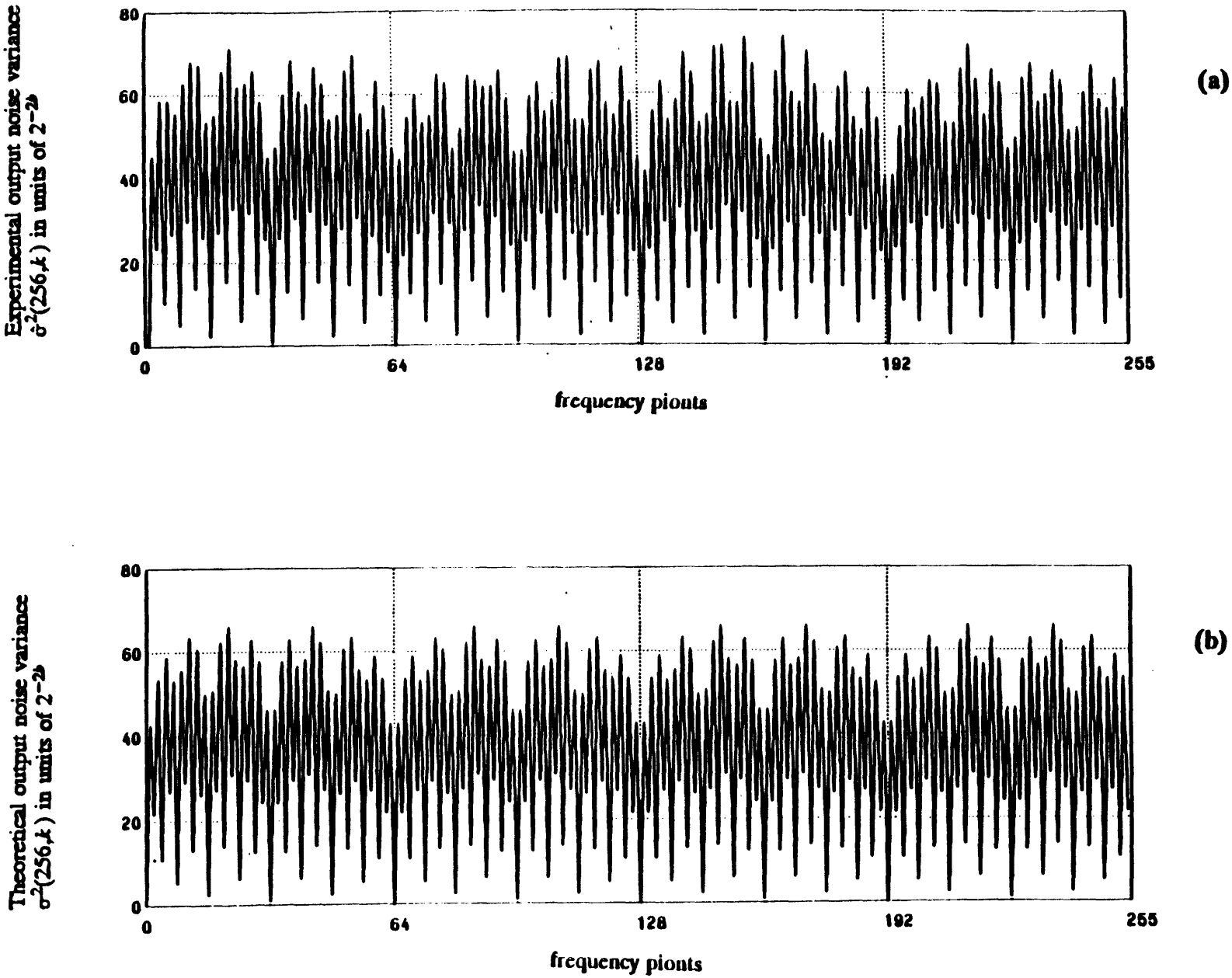


Figure 8.7 Distribution of the output noise variance for 256-point sequences for fixed-point realization of DF2: (a) experimental; (b) theoretical.

On the other hand, for the decimation-in-frequency algorithm DF2, the largest coefficients in the butterflies appear at the very first stage of the algorithm. Therefore their effect is somewhat reduced by the time the signal is passed by the last array of the algorithm. As shown in figure 8.7, there are no peaks similar to figure 8.6 among the frequency points of the decimation-in-frequency algorithm. The wide fluctuations of output noise variance in the neighboring points of figure 8.7 can be associated with the fact that the even and odd frequency points are computed separately.

Recall from section 5.1.1 that a more generalized version of the decomposition used for the DT1 algorithm is given by equation (5.6). The original decomposition shown in equation (5.4) can be considered a special case of the generalized one with $r = 0$. The quantity $\frac{1}{2\cos(\frac{2\pi k(2r+1)}{N})}$ in equation (5.6) attains its largest value for

$$k = \frac{1}{4(2r+1)}[N(2m+1) \pm 4]$$

where m is any integer which makes k of the above equation an integer in the range 0 to $N-1$ (Note that there are only four values of m which result in such values). Therefore, we would expect the distribution of error for the generalized version to have peaks at these frequency points. This has been verified experimentally for $r = 0, 1, 2, 3$ and the resulting output noise variance for 256-point sequences are shown in figure 8.8.

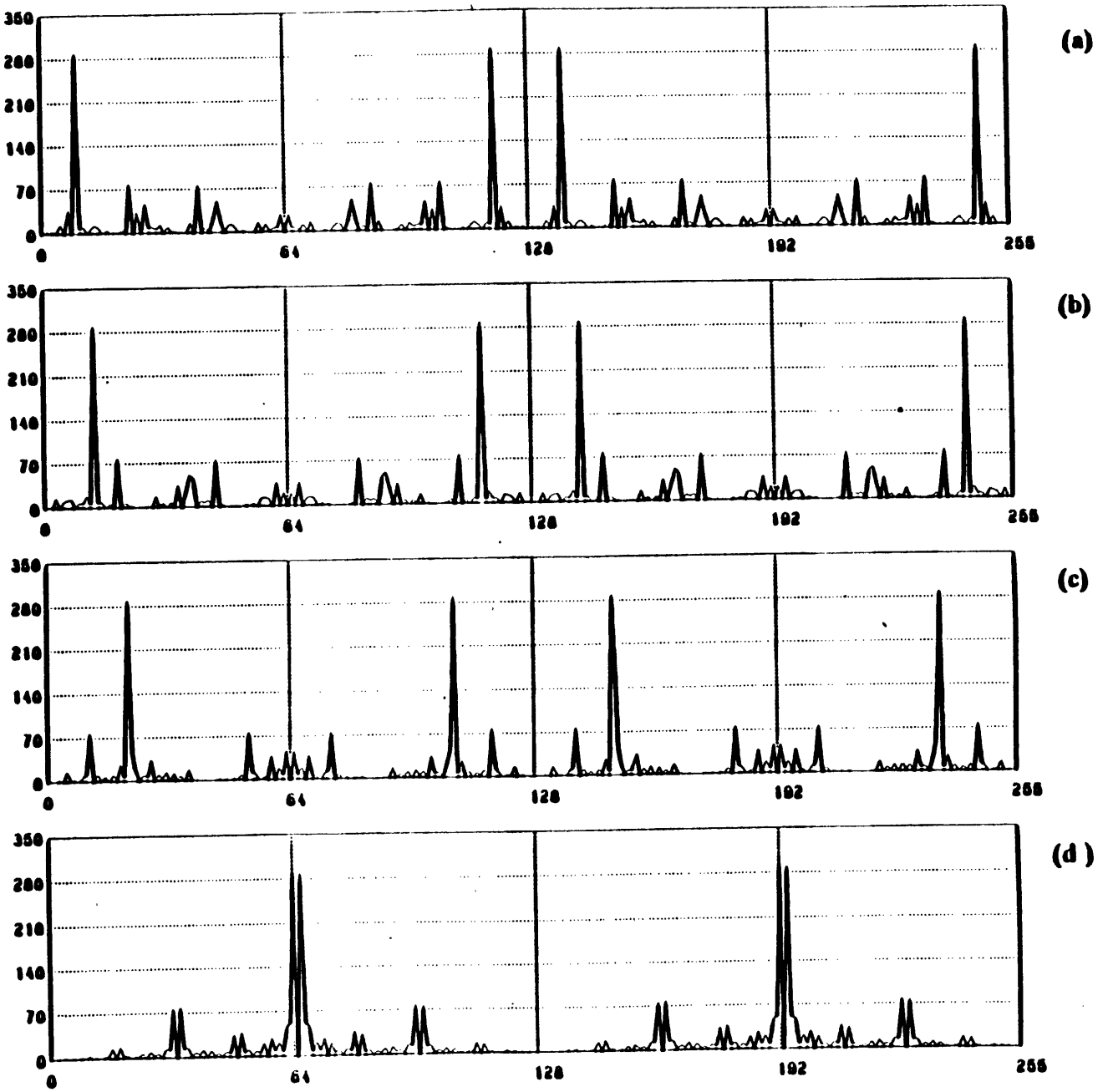


Figure 8.8 Distribution of the output noise variance for 256-point sequences for fixed-point realization of generalized version of the DT1 algorithm: (a) r = 3; (b) r = 2; (c) r = 1; (d) r = 0

8.2.1.2. Floating-point Implementation Results

The corresponding results for the floating-point implementations of the algorithms of the previous section are shown in figures 8.9 and 8.10. As explained in section 8.1 the variance of roundoff error due to multiplications and additions in floating-point implementation with b bits of mantissa is given by

$$\sigma_e^2 = \alpha\, 2^{-2b}$$

The parameter α which is determined by matching the theoretical and experimental noise to signal ratio curves for a specific algorithm, is shown in figures 8.9 and 8.10. The value of α represents essentially an empirical average σ_e^2 for all the multiplications and additions used in computing the DHT of white noise sequences.

Recall from section 6.1.2 and 8.1 that in order to make the signal and error uncorrelated with each other, randomized rounding has to be used when the result of multiplication or addition lies equally between two quantization levels. This situation occurs frequently when we add floating numbers of the same order of magnitude. The experimental results using randomized rounding and non randomized rounding for decimation-in-time and frequency version of Bracewell's algorithm using floating-point arithmetic is shown in figure 8.9. As it is expected, the theoretical curves match the experimental results only if randomized rounding is used.

We can fit the following equations to the data shown in figure 8.9:

$$\frac{\sigma_E^2}{\sigma_{xx}^2\, 2^{-2b}} = .40\,\nu - .53 \qquad DT1 \qquad (8.5a)$$

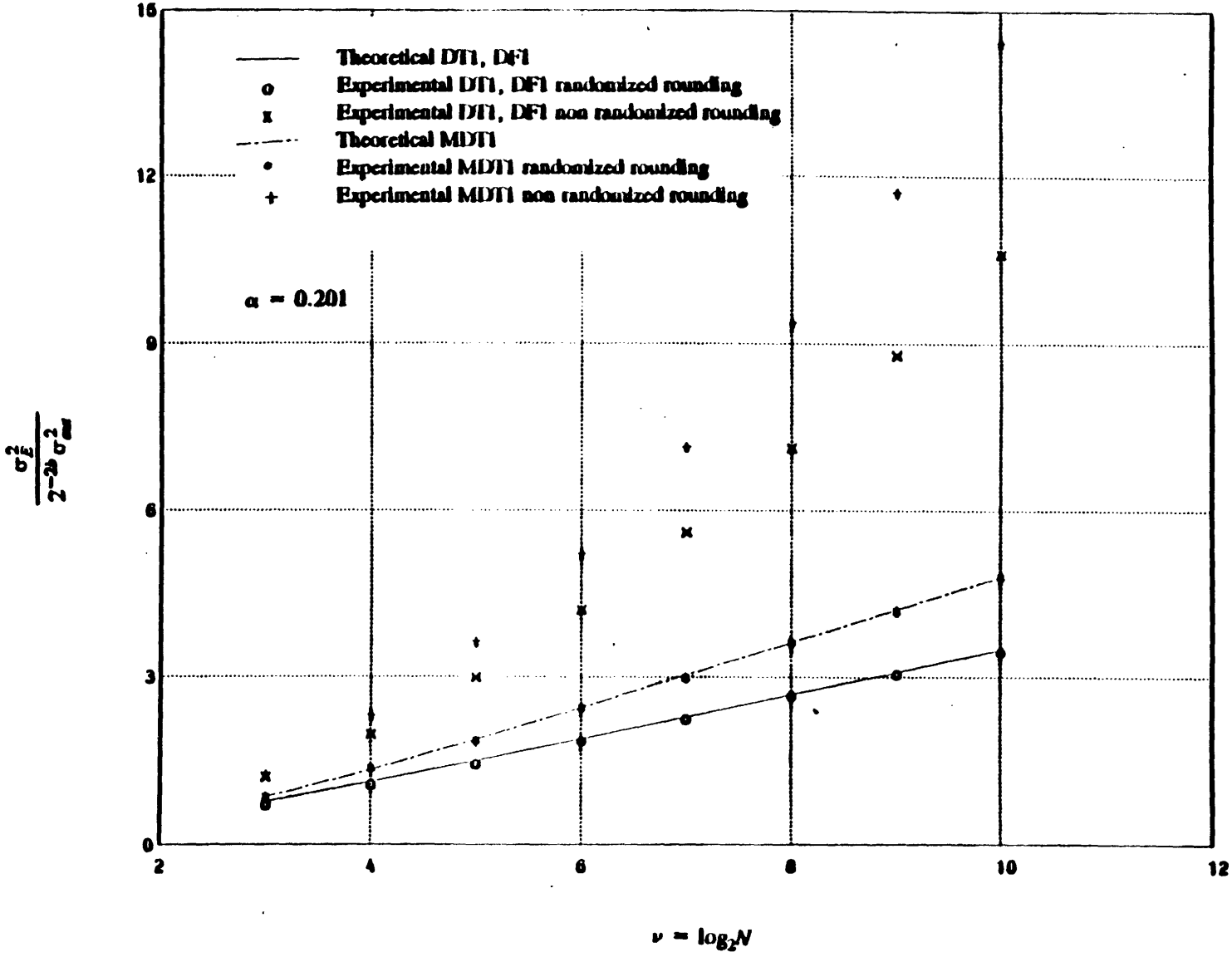


Figure 8.9 Output noise to signal ratio for floating-point realization of DT1 and MDT1 algorithms

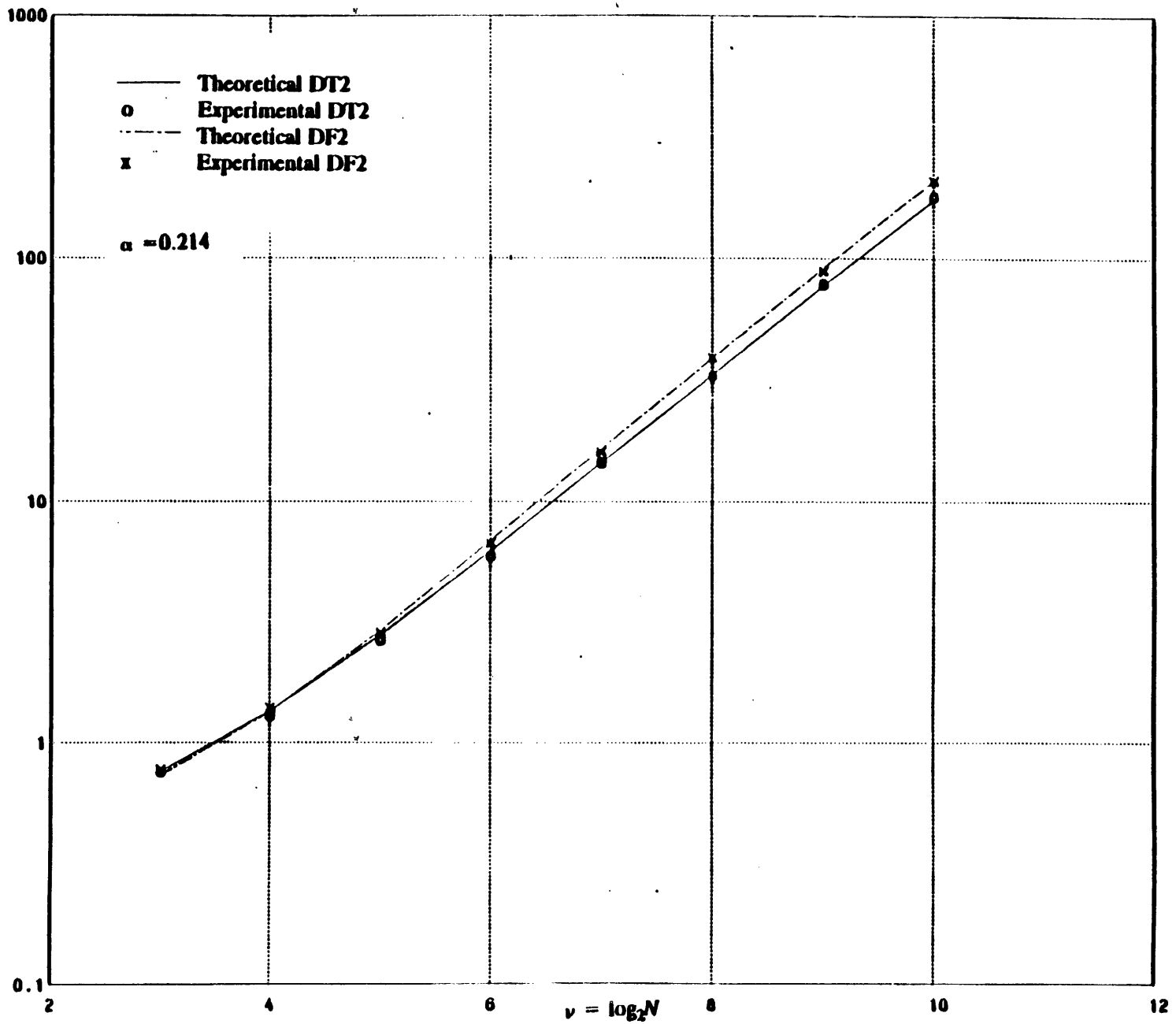


Figure 8.10 Output noise to signal ratio for floating-point realization of DT2 and DF2 algorithms

$$\frac{\sigma_E^2}{\sigma_{out}^2\, 2^{-2b}} = .40\,\nu - .58 \qquad DF1 \qquad (8.5b)$$

$$\frac{\sigma_E^2}{\sigma_{out}^2\, 2^{-2b}} = 0.59\,\nu - 1.09 \qquad MDT1 \qquad (8.5c)$$

$$\frac{\sigma_E^2}{\sigma_{out}^2\, 2^{-2b}} = 0.044\,N^{1.2} \qquad DT2 \qquad (8.6a)$$

$$\frac{\sigma_E^2}{\sigma_{out}^2\, 2^{-2b}} = 0.043\,N^{1.23} \qquad DF2 \qquad (8.6b)$$

Note that unlike the fixed-point case the noise to signal ratio for the MDT1 algorithm shown in figure 8.9 is higher than that of the DT1 algorithm. As explained in section 6.2.2.1 this is due to the correlations between the error at the inputs of the butterflies. As is shown in figure 8.9, the results for the DF1 algorithm are almost identical to that of the DT1 algorithm; Unlike the fixed-point case the dynamic range issues do not exist in floating-point implementations. Therefore as it is seen in figures 8.9 and 8.10 and equations 8.2 and 8.3, the floating-point implementations of the decimation-in-time and frequency algorithms have similar error characteristics.

The theoretical and experimental distribution of variance of error for DT1, MDT1, DF1 and DF2 algorithms shown in figures 8.11 through 8.15 are in good agreement with each other.

8.2.2. Experimental Results for the R4DT1 and R4DT2 algorithms

The error properties of radix 4 versions of all the algorithms discussed in previous sections are very similar to that of radix 2 versions. Figures 8.16 and 8.17 show the experimental noise to signal ratio for DT1, R4DT1, DT2, R4DT2 algorithms. The distribution of roundoff noise due to fixed-point implementation for 256-point

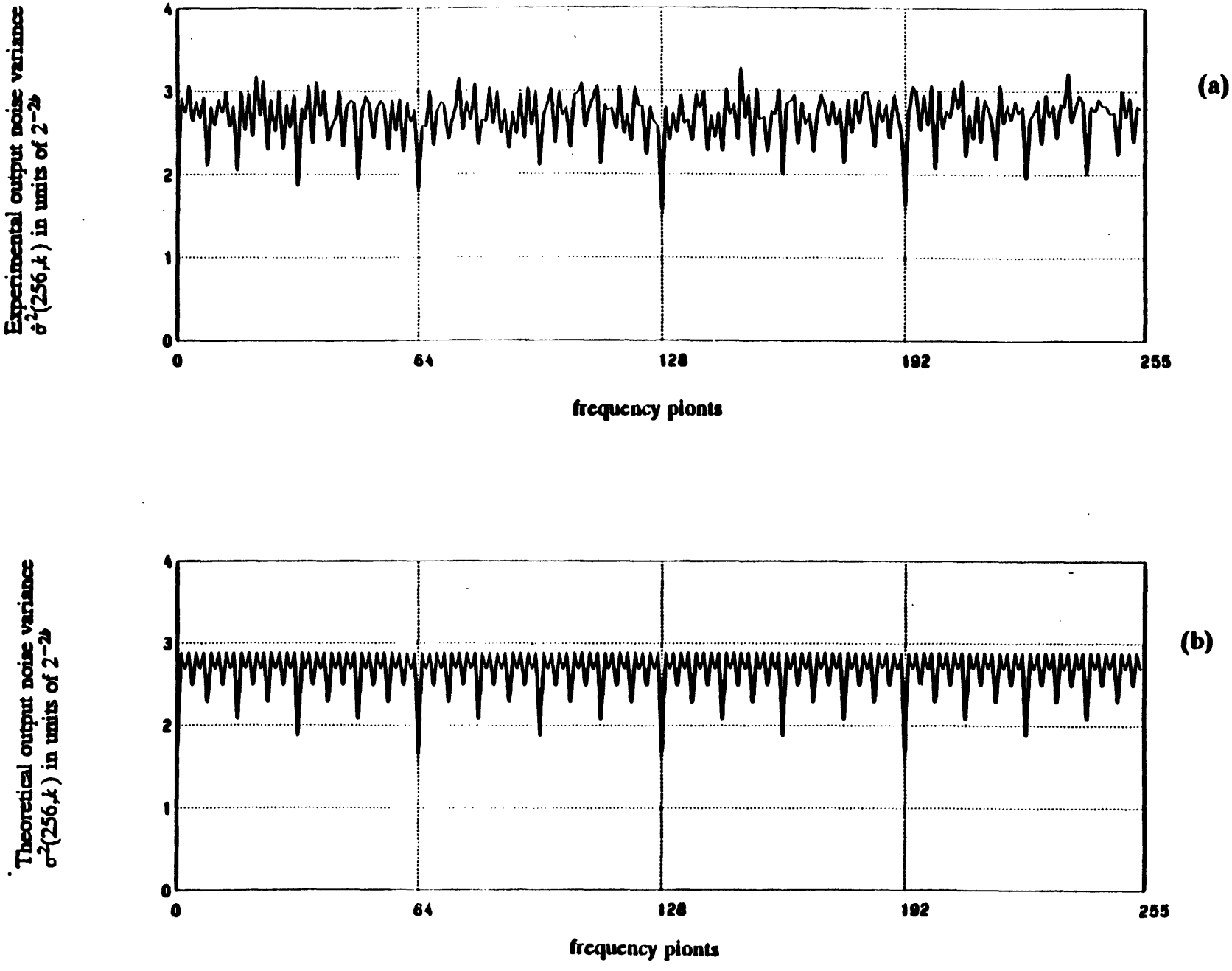


Figure 8.11 Distribution of the output noise variance of 256-point white sequences for floating-point implementation of the DT1 algorithm: (a) experimental; (b) theoretical.

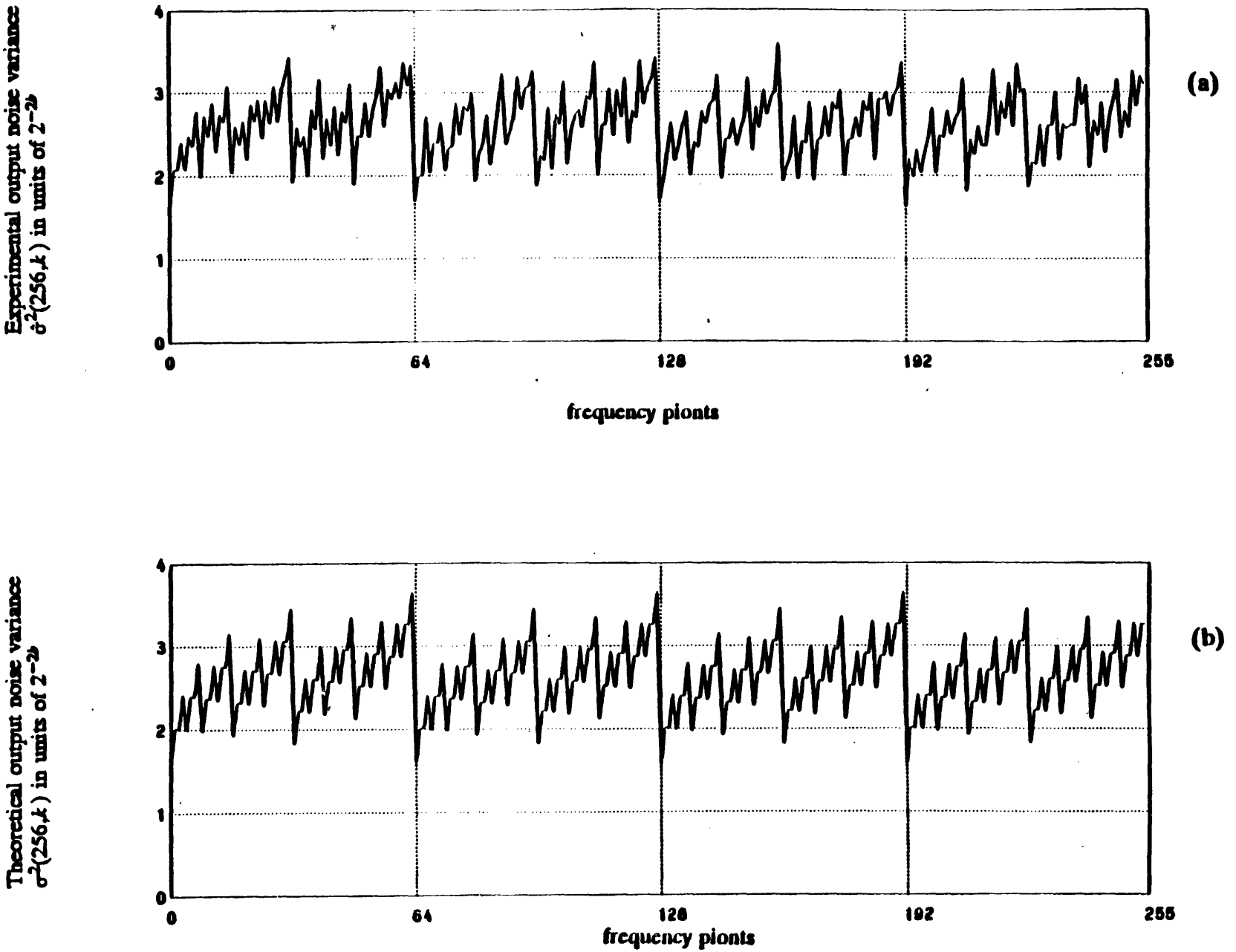
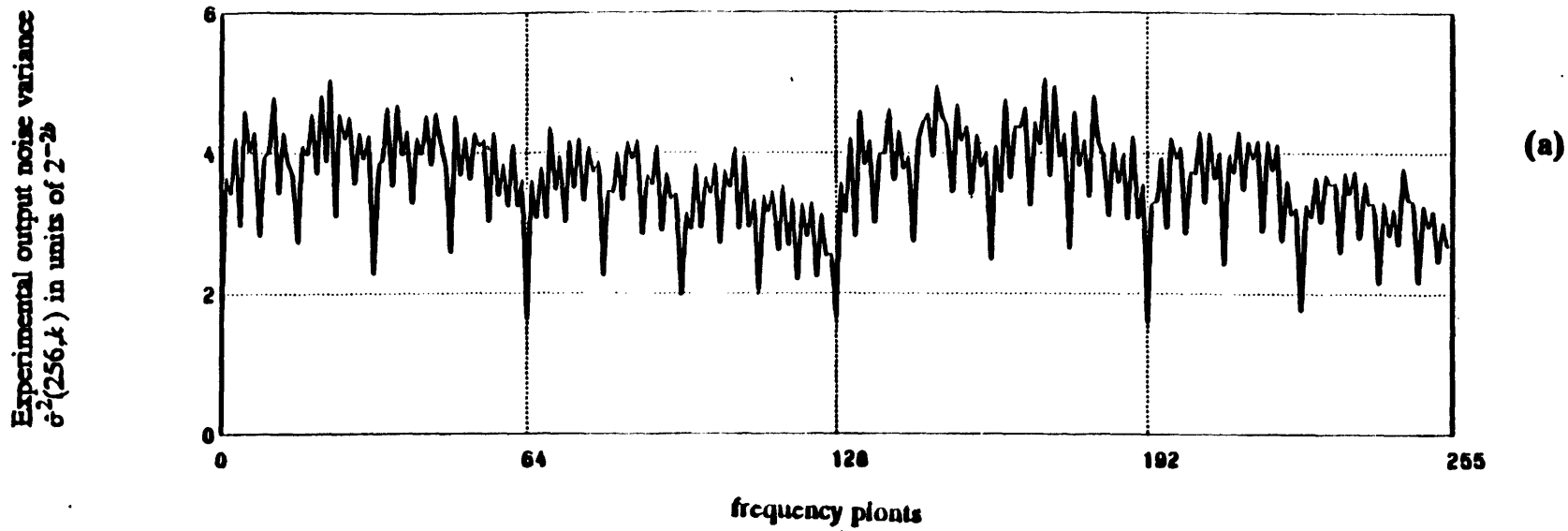


Figure 8.12 Distribution of the output noise variance of 256-point white sequences for floating-point implementation of the MDT1 algorithm: (a) experimental; (b) theoretical.



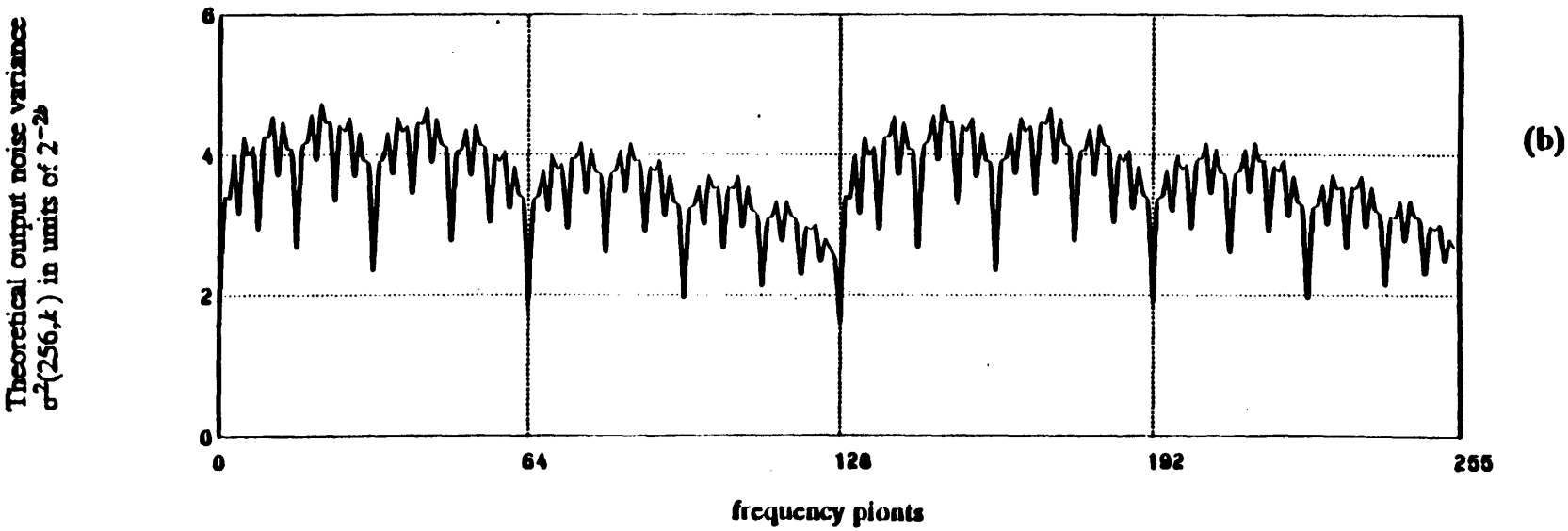


Figure 8.13 Distribution of the output noise variance of 256-point white sequences for floating-point implementation of the DF1 algorithm: (a) experimental; (b) theoretical.

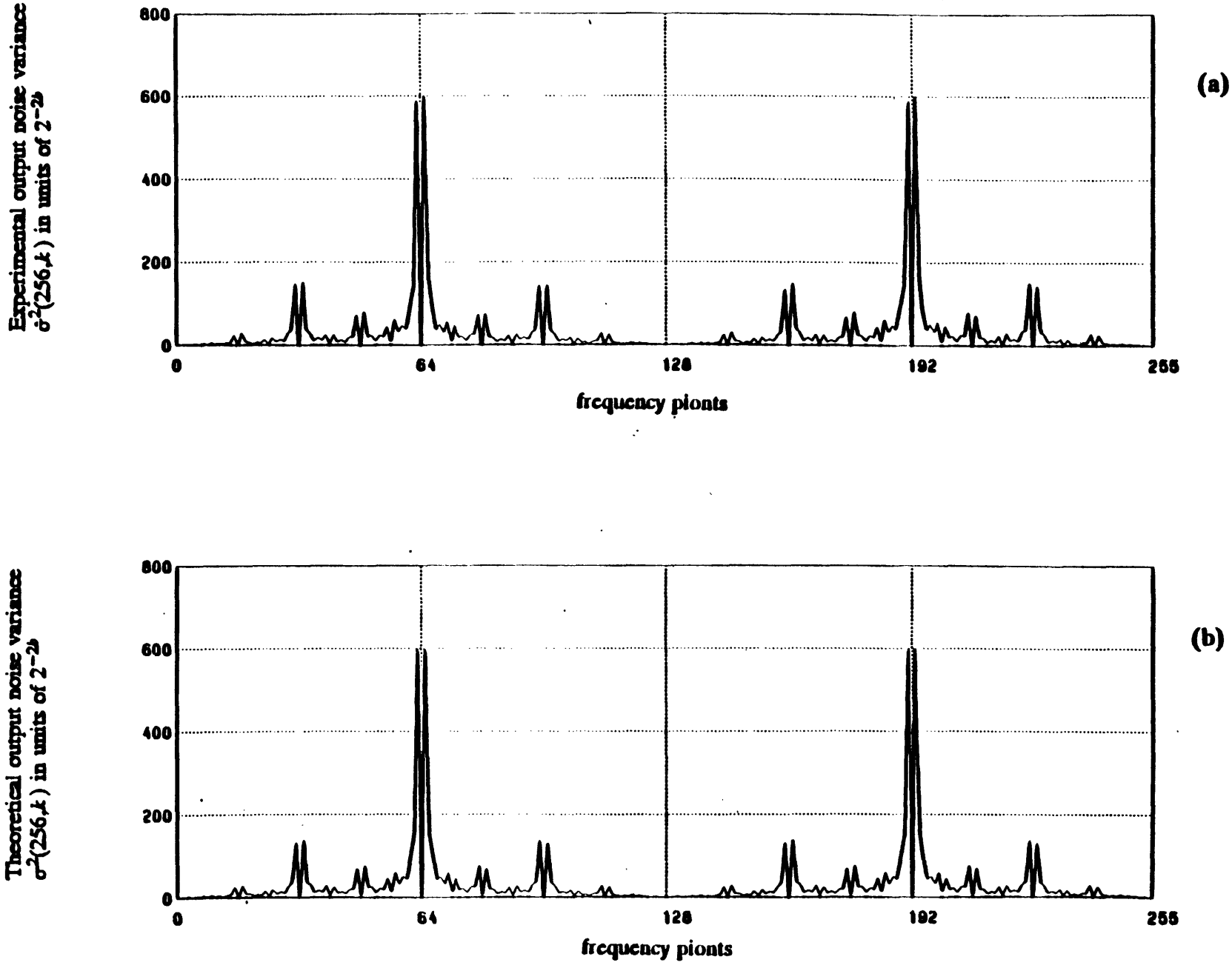


Figure 8.14 Distribution of the output noise variance of 256-point white sequences for floating-point implementation of the DT2 algorithm: (a) experimental; (b) theoretical.

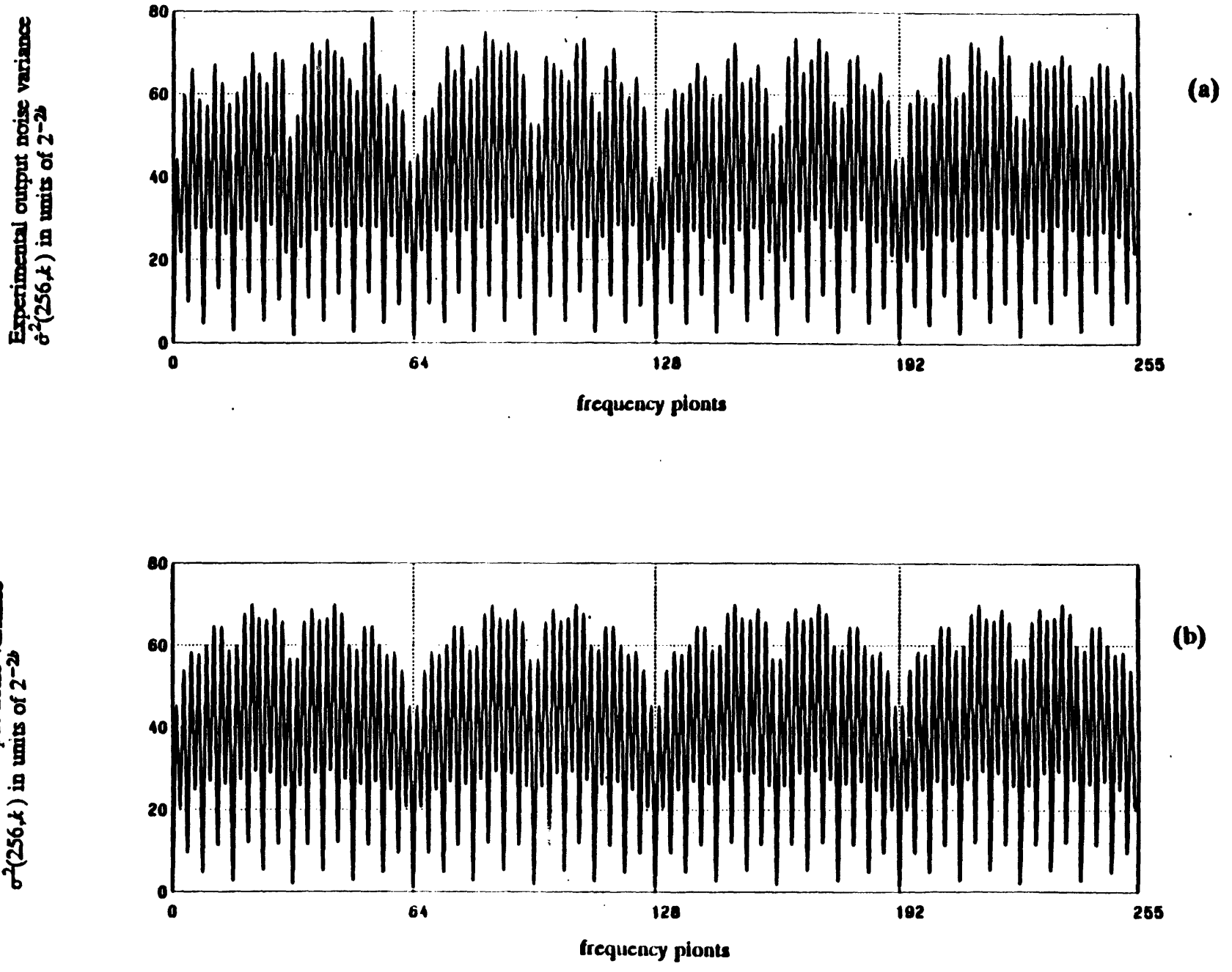


Figure 8.15 Distribution of the output noise variance of 256-point white sequences for floating-point implementation of the DF2 algorithm: (a) experimental; (b) theoretical.

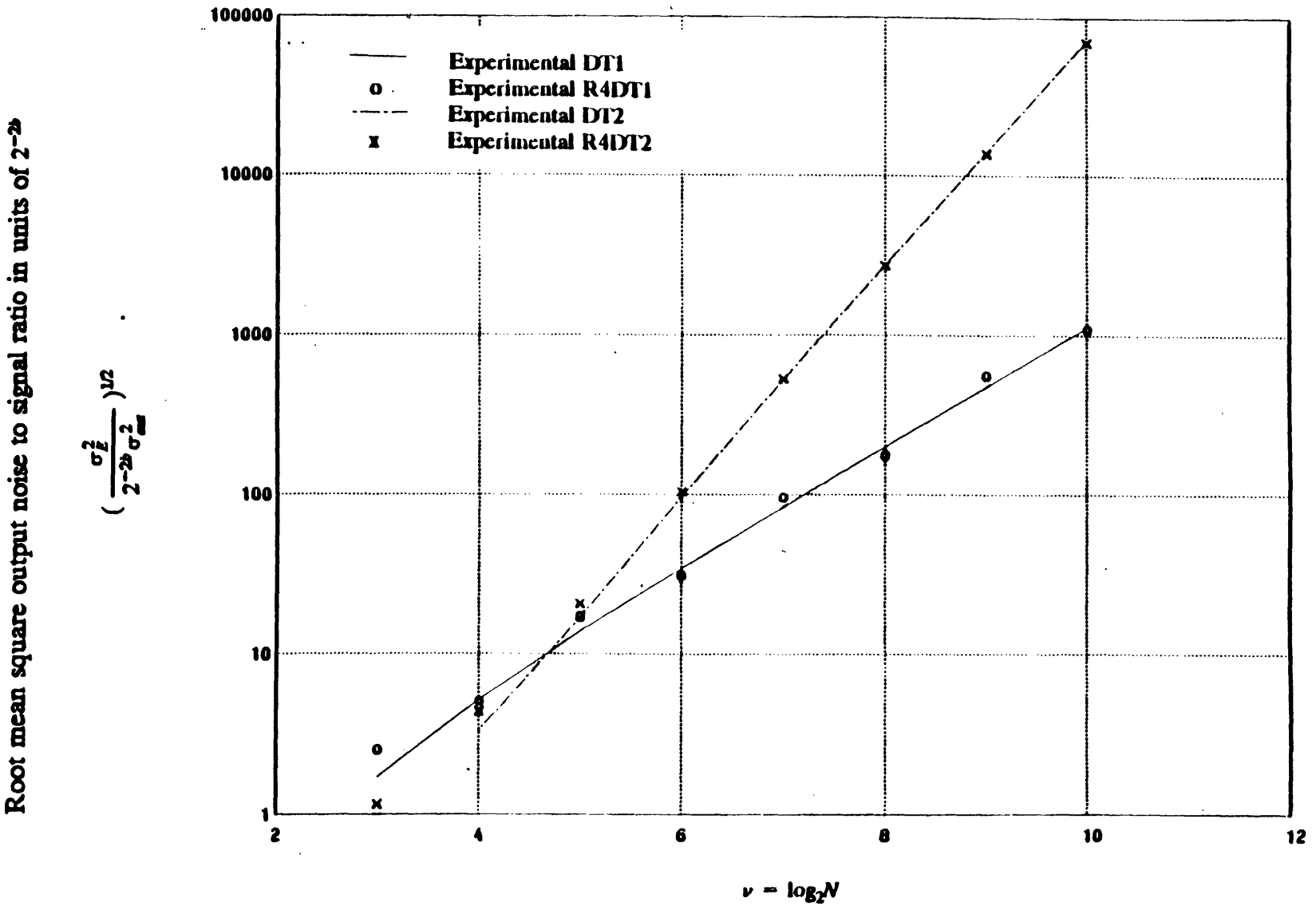


Figure 8.16 Output noise to signal ratio for fixed-point realization of DT1, R4DT1, DT2, R4DT2.

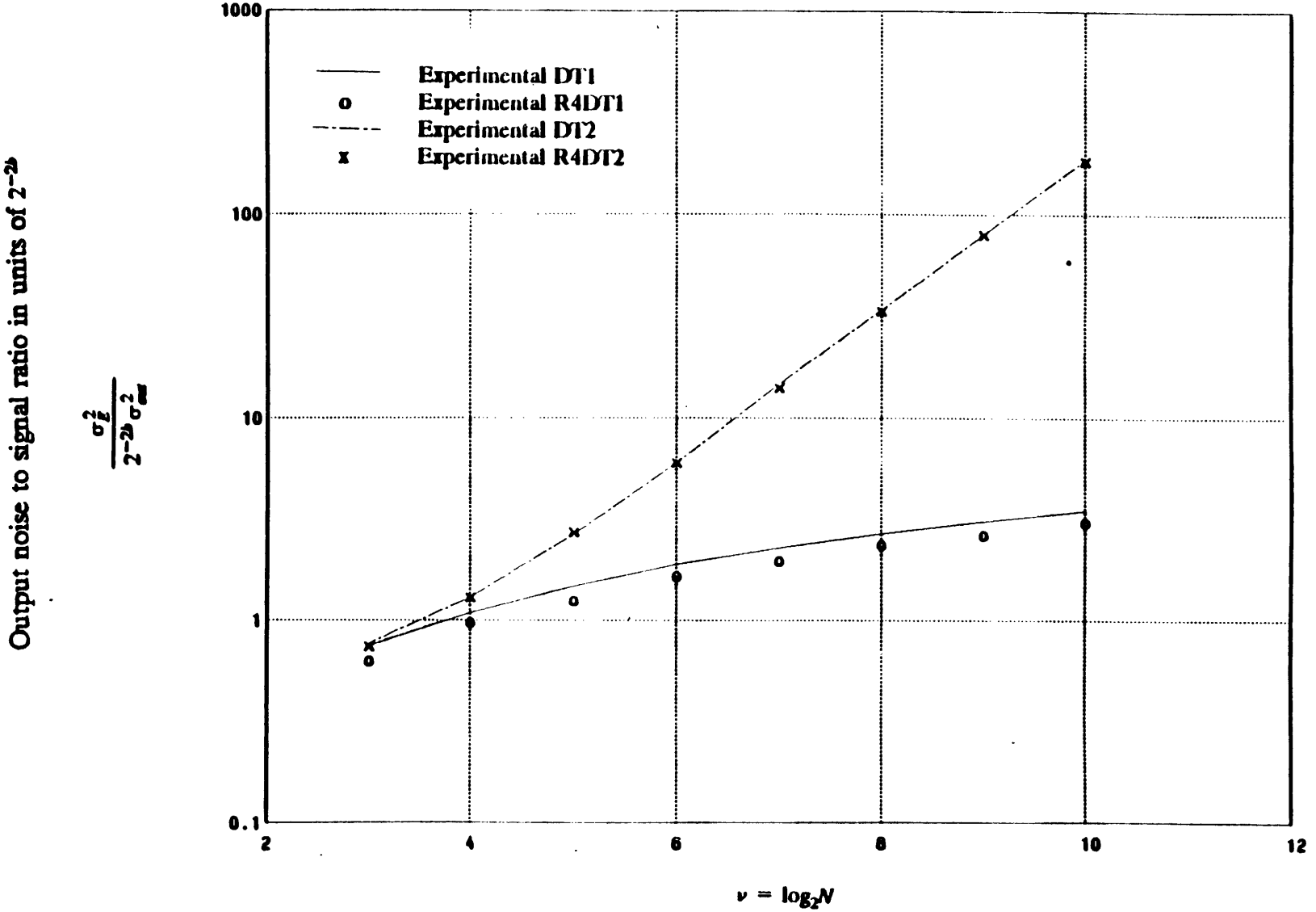


Figure 8.17 Output noise to signal ratio for floating-point realization of DT1, R4DT1, DT2, R4DT2.

sequences using the R4DT2 algorithm is shown in figure 8.18. As is expected, the peaks in the radix 2 version also appear in the radix 4 implementation.

8.3. Comparison of Error Properties of DHT Algorithms

Figures 8.19 and 8.20 show the noise to signal ratio curves for DT1, MDT1, DF1, DT2, DF2, DT3 algorithms using fixed and floating-point arithmetic. The main conclusion which can be drawn from figures 8.19 and 8.20 is that the algorithms of chapter 5 (i.e. DT2 and DF2) and chapter 3 (DT1 and DF1) have the highest and lowest noise to signal ratios respectively. The noise characteristics of Wang's algorithm (DT3) lies somewhere in between these two class of algorithms.

The total operation count for all the algorithms discussed in this thesis is shown in table 8.1. Notice that the total number of arithmetic operations is identical for all the algorithms. However, DT2 and DF2 algorithms use the least number of multiplies and DT1 and DF1 need the most number of multiplies. Therefore, in applications where the multiplication cost (e.g. time) is roughly the same as the addition cost, DT1 and DF1 algorithms will obviously outperform DT2 and DF2. However, in applications where the multiplication is significantly more costly than additions, there is a tradeoff between cost and accuracy depending upon which algorithm is used.

Of course, speed and accuracy are not the only issues involved in comparing algorithms. Complexity, the amount of overhead, storage and other factors are often equally as important. For instance, the DT3 algorithm is by far the most com-

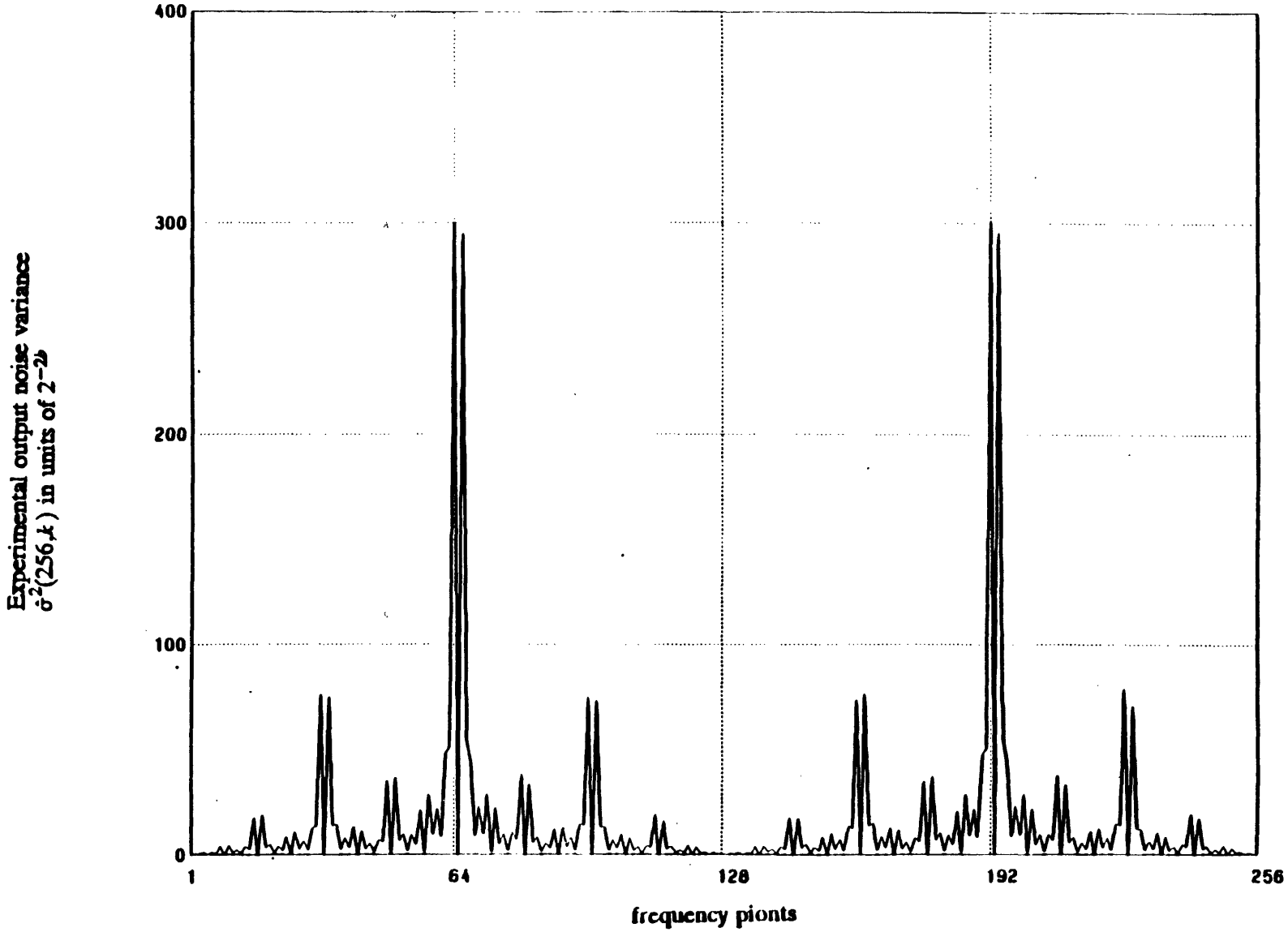


Figure 8.18 Distribution of the output noise variance for 256-point sequences for fixed point realization of R4DF2

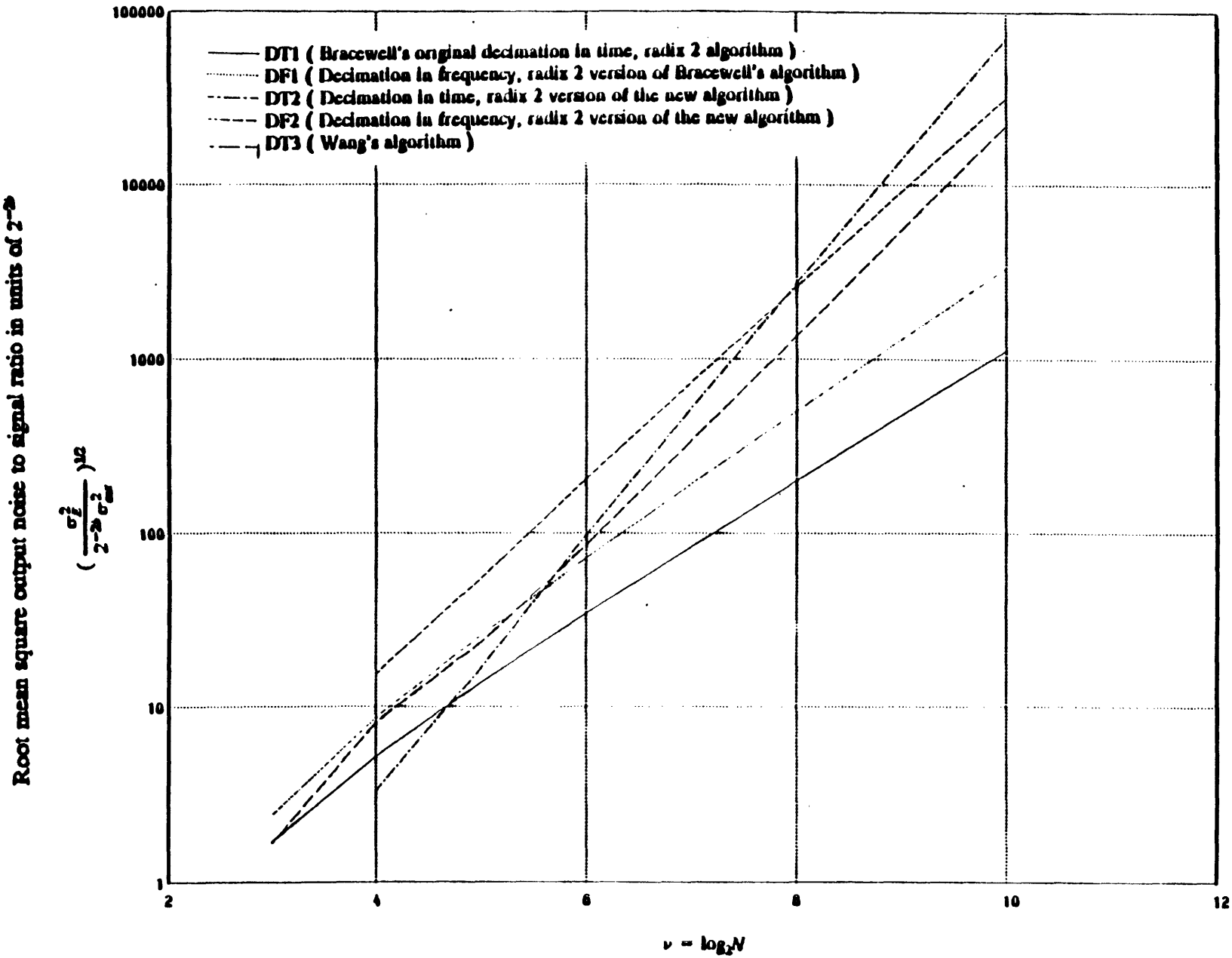


Figure 8.19 Output noise to signal ratio for fixed-point realization of DT1, DF1, DT2, DF2, DT3 algorithms. The error characteristics fo the MDT1 algorithm are almost identical to that of the DT1 algorithm.

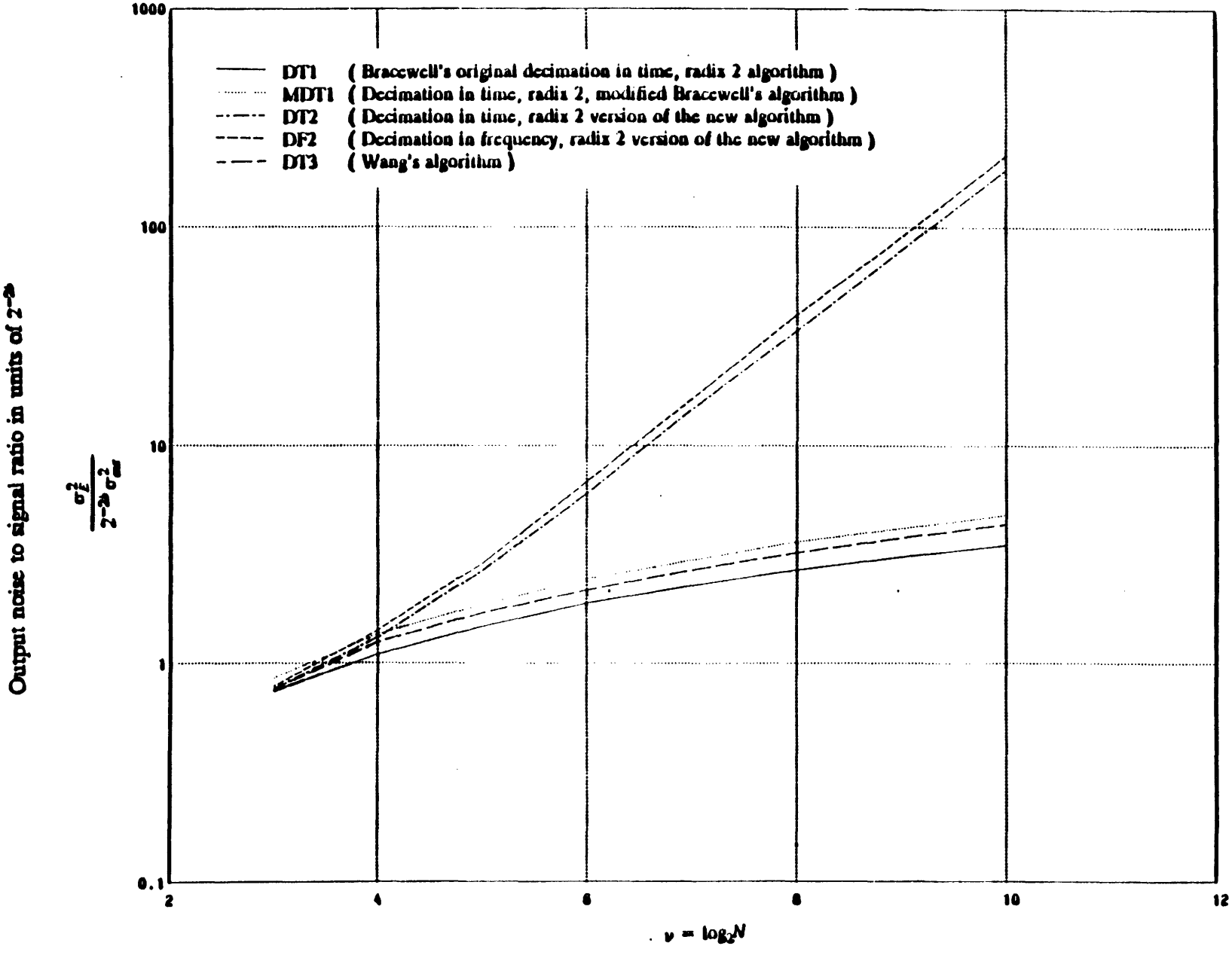


Figure 8.20 Output noise to signal ratio for floating-point realization of DT1, MDT1, DT2, DF2, DT3 algorithms. The error characteristics of the DF1 algorithm are almost identical to that of the DT1 algorithm.

Algorithm	# of Multiplies	# of Adds	Total Operation Count
DT1	$N\log_2 N$	$\frac{3N}{2}\log_2 N$	$\frac{5N}{2}\log_2 N$
DF1	$N\log_2 N$	$\frac{3N}{2}\log_2 N$	$\frac{5N}{2}\log_2 N$
MDT1	$\frac{3N}{4}\log_2 N$	$\frac{7N}{4}\log_2 N$	$\frac{5N}{2}\log_2 N$
DT3	$\frac{3N}{4}\log_2 N$	$\frac{7N}{4}\log_2 N$	$\frac{5N}{2}\log_2 N$
DT2	$\frac{N}{2}\log_2 N$	$2N\log_2 N$	$\frac{5N}{2}\log_2 N$
DF2	$\frac{N}{2}\log_2 N$	$2N\log_2 N$	$\frac{5N}{2}\log_2 N$

Table 8.1 Operation count for various radix 2 DHT algorithms.

plex one discussed in this thesis. The complexity of DT2 (or DF2) and DT1 (or DT1) algorithms are roughly the same. However, because of the special cases shown in equation (5.4) the overhead is slightly higher for DT2 than it is for the DT1 algorithm.

CHAPTER 9: Conclusion and Suggestions For Future Research

9.1. Conclusions

In this thesis, the error properties of various discrete Hartley transform algorithms were investigated theoretically and experimentally. More specifically, we analyzed the arithmetic roundoff error characteristics of DHT algorithms proposed by Bracewell and Wang in addition to a new DHT algorithm.

Statistical models for roundoff errors and linear system noise theory were employed to estimate the output noise variance in various DHT algorithms for fixed and floating-point arithmetic. By considering the overflow constraint in conjunction with these noise analyses, output noise to signal ratios were derived. We used experimental noise measurements to support the predictions obtained via the models. Empirical results shown in chapter 7 were found to be in excellent agreement with the theoretical predictions based on the statistical models.

Having verified our analytical results, we then compared the DHT algorithms of chapters 3, 4 and 5 in terms of their error properties as well as their computational efficiencies. For both fixed and floating-point implementations, Bracewell's algorithm and the new algorithm exhibited the lowest and highest noise to signal ratios respectively.

For a given radix, the total operation count for N-point real sequences was found to be the same using any of the DHT algorithms described in this thesis. However the new algorithm (DT2 and DF2)and Bracewell algorithm (DT1 and

DF1) required the least and most number of multiplications respectively. Speed and efficiencies of these algorithms are also influenced by other factors such as complexity of implementation, storage requirements and the amount of overhead. In fact, in some applications, the data management costs exceed that of the arithmetic operations. Of the three algorithms mentioned above, Wang's algorithm was found to be the most complex one to implement. On the other hand, Bracewell's algorithm which in many ways resembles the FFT, was found to be the most straightforward one to implement. Realization of the new algorithm of chapter 5, was somewhat complicated by having to take care of the special cases.

9.2. Suggestions for Future Research

The finite register length in DHT computations affects the output noise via two different mechanisms: coefficient quantization and arithmetic roundoff error. In this thesis, we have only been concerned with the error due to fixed or floating-point computations. Our approach has been to assume that the DHT coefficients are known with enough precision so that the primary source of error at the output is due to the roundoff arithmetic noise. A possible area of further research would be to find out the way in which coefficient quantization affects the output for different DHT algorithms.

Another possible extension involves handling of overflows. Recall that in fixed-point implementations of various algorithms, we had to scale down the input signal so that we are guaranteed of no overflows. Another way to prevent overflow would be to attenuate the input signal by some factor at every stage of an

algorithm. In the case of the FFT, this results in a considerable improvement in the output noise ratio. Another approach to avoid overflow in the FFT, which could potentially be applied to the DHT algorithms, is the use of block floating-point. In this procedure, the original array is normalized to the far left of the computer word with the restriction that the maximum magnitude of the input signal is less than one. The computation proceeds in a fixed-point manner, except that after every addition, there is an overflow test. If overflow is detected, the entire array is divided by 2 and the computation continues. The number of necessary shifts are counted to determine a scale factor or exponent for the entire final array. The output noise to signal ratio depends strongly on how many overflows occur and at what stage of the computations they occur. The position and timing of the overflows are determined by the signal being transformed and thus in order to analyze the noise to signal ratio in a block floating-point implementation, one needs to know the properties of the input signal. For the case of the FFT, this problem of finding the output noise to signal ratio for white inputs has been analyzed theoretically [13]. It would be interesting to apply this idea to the DHT algorithms and find out the amount of improvement that can be achieved in the noise to signal ratio.

Finally, throughout this thesis we assumed the input signals to be white. This simplified our analysis to a great extent. In the case of the FFT, the output noise to signal ratios for sinusoid type signals have been found to be within 15 percent of that of white signal. It would be worthwhile, to examine the error properties of the DHT algorithms using other types of input signals such as the sinusoids.

References

[1] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing , Englewood Cliffs, NJ, Prentice-Hall Inc. ,1975*

[2] R.V. Hartley, "A More Symmetrical Fourier Analysis Applied to Transmission Problems," *Proc. IRE,* Vol. 30, March 1942, pp. 144-150.

[3] R.N. Bracewell, "The Discrete Hartley Transform," *Journal of Optical Society of America, Vol. 73, December 1983, pp. 1832-1835.*

[4] R.N. Bracewell, "The Fast Hartley Transform, " *Proceedings of the IEEE,* Vol. 72, No. 8, August 1984, pp. 1010-1018.

[5] Z.W. Wang, "Fast Algorithms for the Discrete *W* Transform and for the Discrete Fourier Transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* Vol. ASSP-32, No. 4, August 1984, pp. 803-816.

[6] C.S. Burrus, H.V. Sorenson, D.L. Jones, M.T. Heideman, "On Computing the Discrete Hartley Transform," *to be submitted for publication.*

[7] P. Duhamel and H. Hollmann, "Split Radix FFT Algorithm," *Electronic Letters,* Vol. 32, No. 4, August 1984, pp. 750-762.

[8] C.S. Burrus, H.V. Sorensen, M.T. Heidemen," On Computing the Split-Radix FFT," *submitted for publication.*

[9] C.S. Burrus, "Index Mapping for Multidimensional Formulation of the DFT and Convolution," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* Vol. ASSP-25, No. 3, June 1977, pp. 239-242.

[10]J. Makhoul, "A Fast Cosine Transform in One and Two Dimensions," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* Vol. ASSP-28, No. 1, February 1980, pp. 27-34.

[11]C.J. Weinstein, "Roundoff Noise in Floating Point Fast Fourier Transform Computation," *IEEE Trans. on Audio an Electroacoustics,* Vol. AU-17, No. 3, September 1969, pp. 209-215. pp. 90-93.

[12]A.V. Oppenheim and C.J. Weinstein, "Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform," *Proceedings of the IEEE,* Vol. 60, No. 8, August 1972, pp. 957-978.

[13]C.J. Weinstien, "Quantization Effects in Digital Filters," Ph.D. Thesis, Department of Electrical Engineering, MIT, 1969.

[14]W.H. Chen, C.H. Smith, and S.C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE Trans. Commun., Vol. COM-25, 1977, pp. 1004-1009.*

[15]Z. Wang, "Reconsideration of 'A Fast Computational Algorithm for the Discrete Cosine Transform, ' " *IEEE Trans. Commun., Vol. COM-31, 1983, pp. 121-123.*

[16]L.R. Rabiner, R.W. Schafer, and C.M. Rader, "The Chirp z-Transform Algorithm," *IEEE Trans. Audio Electroacoust.,* Vol. AU-17, June 1969, pp. 86-92.

[17]R.N. Bracewell, *The Fourier Transform and Its Applications,* McGraw-Hill, 1975.

Appendix A : A Numerical Technique to Determine the Overflow Constraint

As we mentioned in chapter 6, dynamic range issues become important in fixed-point realization of DHT algorithms. In particular, in order to obtain output noise to signal ratio, we have to consider the overflow constraint in conjunction with our noise analysis. For a given algorithm, we can ensure against overflow by keeping the input $x(n)$ sufficiently small so that no element of the intermediate or the output array exceeds unity. In this appendix we will describe a way of obtaining upper bounds on the maximum magnitude of input signal in order to prevent overflows.

Suppose we would like to find the overflow constraint for N-point input sequences to a given algorithm A. Consider another algorithm B which is exactly identical to A except that at every stage signals and the coefficients of butterflies are replaced with their absolute values and minus signs are replaced with plus signs and subtractions with additions. Now let us pass an N-point sequence of all ones through algorithm B (using floating-point arithmetic so that we are guaranteed of no overflows) and find the maximum magnitude of the output array $|X_{out}|_{max}$ numerically.

Because of the way algorithm B is constructed, as we move from one stage to the next the maximum magnitude of the elements of arrays increases. That is

$$|X_n|_{max} < |X_{n+1}|_{max} \qquad \text{(A.1)}$$

where $|X_i|_{max}$ denotes the maximum magnitude of the elements of the *ith* array of the algorithm. This implies that if no overflows occur in the final array (output), we are guaranteed of no overflows in the intermediate stages. Therefore if we chose the maximum magnitude of the input signal $|X_{in}|_{max}$ to be

$$|X_{in}|_{max} < \frac{1}{|X_{out}|_{max}} \tag{A.2}$$

we can ensure against overflows in the entire algorithm B.

So far, we have only been concerned with the overflow constraint for algorithm B. Examining the way we constructed algorithm B from A, we realize that the maximum magnitude of all the intermediate quantities in B are larger than the corresponding quantities in A. Therefore, the constraint shown in equation (A.2) also applies to algorithm A. It is important to note that for algorithm A, the above bound is a sufficient but not a necessary condition on the input to prevent overflows.

To summarize, we constructed a new algorithm B which was identical to algorithm A except that at every stage signals and coefficients of butterflies were replaced with their absolute values and subtractions were replaced by additions. Then we found the maximum magnitude of the output of algorithm B, $|X_{out}|_{max}$, with a sequence of all ones as its input. The overflow constraint for algorithm A is then given by equation (A.2).

Appendix B : Estimating the Output Noise Variance From the Experimental Data

Recall from our experimental procedure described in chapter 7 that in order to find a stable estimate of the output noise variance for each frequency point of a given transform size, we used 1000 experimental values for error. In this appendix we will characterize the estimator used in predicting the output noise variance, and state the reason behind choosing 1000 as the number of experimental values of error.

The estimator used is basically of the form

$$\hat{\sigma}^2(N,k) = \frac{1}{n}\sum_{i=1}^{n}\left(\epsilon_i(N,k) - \hat{m}_\epsilon(N,k)\right)^2 \tag{B.1a}$$

$$\hat{m}_\epsilon(N,k) = \frac{1}{n}\sum_{i=0}^{n}\epsilon_i(N,k) \tag{B.1b}$$

where $\epsilon_i(N,k)$ denotes the *ith* experimental value of the error at the *kth* frequency point of an N-point transform, n is the number of experiments and $\hat{m}_\epsilon(N,k)$ and $\hat{\sigma}^2(N,k)$ denote the estimates for the mean and variance of error at the *kth* frequency point of an N-point transform. The mean and variance of $\hat{m}_\epsilon(N,k)$ are given by

$$E\left[\hat{m}_\epsilon(N,k)\right] = m_\epsilon(N,k) \tag{B.2a}$$

$$Var\left[\hat{m}_\epsilon\right] = \frac{\sigma^2(N,k)}{n} \tag{B.2b}$$

Thus $\hat{m}_\epsilon(N,k)$ is an unbiased and consistent estimate of $m_\epsilon(N,k)$. This implies that no matter what the probability distribution for $\epsilon(N,k)$ is, $\hat{m}_\epsilon(N,k)$ is a good esti-

mate of $m_\epsilon(N,k)$. Moreover, if we assume that $\epsilon(N,k)$ is a normal random variable with mean $m_\epsilon(N,k)$ and variance $\sigma^2(N,k)$, then $\hat{m}_\epsilon(N,k)$ is the maximum likelihood estimator and is efficient and sufficient.

On the other hand, the expected value of $\hat{\sigma}^2(N,k)$ of equation (B.1a) is given by

$$E\ [\ \hat{\sigma}^2(N,k)\ ] = \frac{n-1}{n}\sigma^2(N,k) \tag{B.3}$$

Thus $\hat{\sigma}^2(N,k)$ is biased. Furthermore, if we assume that $\epsilon_i(N,k)$ is a normal random variable with mean $m_\epsilon(N,k)$ and variance $\sigma^2(N,k)$, $\hat{\sigma}^2(N,k)$ of equation (B.1a) is the maximum likelihood estimator and is consistent. In this case the probability distribution for $\hat{\sigma}^2(N,k)$ is given by

$$\frac{n\hat{\sigma}^2(N,k)}{\sigma^2(N,k)} \approx \chi_n^2 \tag{B.4}$$

where χ_n^2 denotes the chi square distribution with n degrees of freedom defined by

$$f_Y(y) = \frac{1}{2^{n/2}\Gamma(n/2)} y^{n/2-1}\, e^{y/2} \tag{B.5}$$

Since the variance of a random variable with chi square distribution of degree n is $2n$, using equation (B.4) we get

$$Var\,[\hat{\sigma}^2(N,k)] = \frac{2\sigma^4(N,k)}{n} \tag{B.6}$$

In order to obtain a stable estimate of $\hat{\sigma}^2(N,k)$ we would like its standard deviation to be much smaller that its mean. That is

$$\sqrt{\frac{2}{n}}\sigma^2(N,k) \ll \frac{n-1}{n}\sigma^2(N,k) \tag{B.7}$$

Thus by choosing n to be 1000, the standard deviation of $\hat{\sigma}^2(N,k)$ becomes about 3 percent of its mean.

DISTRIBUTION LIST

	DODAAD Code	
Director Defense Advanced Research Project Agency 1400 Wilson Boulevard Arlington, Virginia 22209 Attn: Program Management	HX1241	(1)
Group Leader Information Sciences Associate Director for Engineering Sciences Office of Naval Research 800 North Quincy Street Arlington, Virginia 22217	N00014	(1)
Administrative Contracting Officer E19-628 Massachusetts Institute of Technology Cambridge, Massachusetts 02139	N66017	(1)
Director Naval Research Laboratory Attn: Code 2627 Washington, D.C. 20375	N00173	(6)
Defense Technical Information Center Bldg. 5, Cameron Station Alexandria, Virginia 22314	S47031	(12)
Dr. Judith Daly DARPA/TTO 1400 Wilson Boulevard Arlington, Virginia 22209		(1)