NBSIR 73-303

FAST FOURIER TRANSFORM IMPLEMENTATION FOR THE CALCULATION OF NETWORK FREQUENCY DOMAIN TRANSFER FUNCTIONS FROM TIME DOMAIN WAVEFORMS

William L. Gans and N. S. Nahman

Electromagnetics Division
Institute for Basic Standards
National Bureau of Standards
Boulder, Colorado 80302

December 1972

Final Report

Prepared for
DOD Calibration Coordination Group
72-65

U.S. DEPARTMENT OF COMMERCE, Frederick B. Dent, Secretary
NATIONAL BUREAU OF STANDARDS, Richard W. Roberts, Director

CONTENTS

LIST OF ILLUSTRATIONS

FAST FOURIER TRANSFORM IMPLEMENTATION FOR THE CALCULATION OF NETWORK FREQUENCY DOMAIN TRANSFER FUNCTIONS FROM TIME DOMAIN WAVEFORMS

by

William L. Gans and N. S. Nahman

ABSTRACT

This report is concerned with the software applications of the fast Fourier transform algorithm to the relationship between time domain waveforms and frequency domain spectra. The first chapter is devoted to a description of the discrete Fourier transform and the fast Fourier transform. Chapter 2 contains the text and a brief description of all FORTRAN II programs utilized in connection with this work. All computation was performed on the in-house time share computing system in the NBS facilities, Boulder, Colorado. In Chapter 3, problems encountered using the fast Fourier transform algorithm are discussed, an example of a time domain to frequency domain calculation is presented, and future developmental considerations are mentioned. In addition Appendix A contains a detailed example aimed at disclosing the inner mechanisms of the fast Fourier transform algorithm.

Key Words: Discrete Fourier transform; Fast Fourier transform; Frequency spectra, discrete; Network transfer function; Time domain waveform; Transfer function.

1. DESCRIPTION OF FAST FOURIER TRANSFORM

1.1 Definition of DFT.

For many years, the Fourier integral transform, the Fourier series, and the Laplace transform have been available to engineers for the purposes of time domain-frequency domain correlations. More recently, the sampling theorem and the discrete Fourier Transform (DFT) have evolved as useful tools for handling sampled-data band-limited time waveforms and frequency spectra. Thus, while the Fourier series and the Fourier and Laplace transforms are well suited to operations involving continuous, or piece-wise continuous functions, the DFT allows operations to be performed directly upon a series of discrete data samples representing a continuous time function.

Most recently, (1965) a method has been reported [1] whereby the Fourier coefficients of the DFT may be machine calculated in a fraction of the time previously required. This algorithm is commonly referred to as the Fast Fourier Transform, or FFT.

In order to discuss the FFT it is first necessary to define the DFT. Definitions vary in the literature , however, this paper will follow the definitions and notations used in [2]. The DFT is defined by

$$Ar = \sum_{k=o}^{N-1} X_k e^{-2\pi jrk/N} \qquad r = 0, 1, 2\ldots, \quad N-1 \qquad (1.1)$$

where Ar is the rth coefficient of the DFT and X_k is the k^{th} sample of the time series being transformed. $j = \sqrt{-1}$ and there are N samples in the total time series. Thus, a time waveform consisting of N sequential samples will yield a transform consisting of N frequency domain coefficients. For convenience, let

$$W = e^{-2\pi j/N} \qquad (1.2)$$

then, in simplified form,

$$A_r = \sum_{k=o}^{N-1} X_k W^{rk} \qquad (1.3)$$

As in the case of the Fourier integral pair there exists an inverse transform of the DFT. This transform is written

$$X_\ell = \frac{1}{N} \sum_{r=o}^{N-1} A_r W^{-r\ell} \qquad \ell = 0, 1, 2 \ldots, \quad N-1 \qquad (1.4)$$

and is called the inverse discrete Fourier transform or IDFT.

1.2 Some Properties of the DFT

Both the Ar's and X_ℓ's may be complex numbers; the Ar's, in general, are almost always complex, but for a real time series waveform, the X_ℓ's will be real.

In addition there exists a convolution relationship between the DFT and IDFT. The IDFT of the product of two DFT's is the periodic mean convolution of the two time series waveforms of the DFT's.

Likewise, most other properties of the Fourier integral transform may be shown to apply to the DFT. As a further example, the DFT of the sum of two time series waveforms is the sum of the DFT's of the two waveforms.

Thus, with few exceptions, the DFT lends itself to time-frequency operations on discrete sampled-data functions with the same power and versatility as the Fourier integral transform brings to operations on continuous functions. One exception to be kept in mind, however, is that data obtained by use of the DFT is valid only at the points in time or frequency where samples were taken. Interpolation between the Ar's or X_ℓ's is neither valid nor wise.

1.3 Definition of the Fast Fourier Transform

The fast Fourier transform is an algorithm which allows the machine computation of a set of DFT coefficients to be accomplished much faster and more efficiently. Previous straightforward methods require approximately N^2 arithmetic operations to transform a time series waveform consisting of $N = 2^n$ samples. By use of the FFT, the same results can be obtained with only $2N \log_2 N$ operations. A 1024 point waveform, for example, would require about $(1024)^2$ or 1,048,576 arithmetic operations to obtain its 1024 DFT coefficients by conventional means. Using the FFT, these same coefficients may be obtained after only (2)(1024)(10) or 20,480 operations. Thus, the time and effort required to transform a 1024 point time series using the FFT is about 1/50th of that required using straight-forward techniques. With larger time series, the savings increase.

Although many variations of the FFT algorithm now exist, only Cooley and Tukey's decimation in time algorithm shall be described. [1]. Suppose we wish to calculate the DFT of a time series waveform consisting of N samples. For simplicity, let $N = 2^n$ where n is an integer. The DFT of this waveform is defined as:

$$A_r = \sum_{k=o}^{N-1} X_k e^{-2\pi jrk/N} \qquad r = 0, 1, 2\ldots, N-1 \qquad (1.5)$$

If the original time sequence is now split into two sequences such that the even-numbered X_k's become one sequence and the odd-numbered X_k's the other sequence, then we may define,

$$Y_k = X_{2k}$$

and

$$k = 0, 1, 2, \ldots, \frac{N}{2} - 1$$

$$Z_k = X_{2k+1} \qquad (1.6)$$

Both Y_k and Z_k are now $\frac{N}{2}$-point time series and they each have DFT's as follows,

$$B_r = \sum_{k=o}^{\left(\frac{N}{2}-1\right)} Y_k \, e^{-4\pi jrk/N} \qquad r = 0, 1, 2, \ldots, \frac{N}{2}-1$$

and

$$C_r = \sum_{k=o}^{\left(\frac{N}{2}-1\right)} Z_k \, e^{-4\pi jrk/N} \qquad r = 0, 1, 2, \ldots, \frac{N}{2}-1 \tag{1.7}$$

Now, writing the original DFT in terms of these two new ones:

$$A_r = \sum_{k=o}^{\left(\frac{N}{2}-1\right)} \left\{ Y_k \, e^{-4\pi jrk/N} + Z_k \, e^{-\frac{2\pi jr}{N}[2k+1]} \right\} \qquad r = 0, 1, 2, \ldots, N-1 \tag{1.8}$$

or

$$A_r = \sum_{k=o}^{\left(\frac{N}{2}-1\right)} Y_k \, e^{-4\pi jrk/N} + e^{-2\pi jr/N} \sum_{k=o}^{\left(\frac{N}{2}-1\right)} Z_k \, e^{-4\pi jrk/N} \tag{1.9}$$

using (1.7),

$$A_r = B_r + e^{-2\pi jr/N} C_r \qquad r = 0, 1, 2\ldots, \frac{N}{2}-1 \tag{1.10}$$

For values of r greater than $\frac{N}{2}$, the DFT's of B_r and C_r repeat periodically their values for $r < \frac{N}{2}$. Therefore, substituting $r + \frac{N}{2}$ for r in (1.10) yields

$$A\left(r+\frac{N}{2}\right) = B_r + e^{\frac{-2\pi j}{N}\left[r+\frac{N}{2}\right]} C_r \qquad 0 \le r < \frac{N}{2} \qquad (1.11)$$

or, since $e^{-j\pi} = -1$,

$$A\left(r+\frac{N}{2}\right) = B_r - e^{-2\pi jr/N} C_r \qquad 0 \le r < \frac{N}{2} \qquad (1.12)$$

Therefore (1.2), (1.10) and (1.12) may be combined to yield A_r in terms of B_r and C_r:

$$A_r = B_r + W^r C_r \qquad 0 \le r < \frac{N}{2} \qquad (1.13)$$

$$A\left(r+\frac{N}{2}\right) = B_r - W^r C_r \qquad 0 \le r < \frac{N}{2} \qquad (1.14)$$

Thus, the DFT of X_k may be obtained by computing two smaller DFT's of length $\frac{N}{2}$. It should be noted that

$$W^r = -W^{\left(r-\frac{N}{2}\right)} \qquad (1.15)$$

If this decimation process is continued, B_r and C_r would each reduce to two $\frac{N}{4}$-point transforms. The resulting four transforms would reduce to eight $\frac{N}{8}$-point transforms, and eventually, the problem would be reduced to solving $\frac{N}{2}$ two-point transforms. In particular, for a 2^n-point waveforn, n such reductions will be necessary to reduce the problem to one of solving only two-point transforms.

The result of this reduction process is a problem consisting only of complex multiplications and additions. Furthermore, three-fourths of the multiplications may be eliminated because they either involve multiplication by unity, or they are redundant multiplications. Thus,

in general, an N-point transform will consist of $N \log_2 N$ complex additions and $\frac{1}{2} N \log_2 N$ complex multiplications. See Appendix A for a detailed example of an eight-point transform.

2. OPERATING SYSTEM SOFTWARE

2.1 FFT Subroutines

Four subroutines are used to compute the DFT and IDFT of a time waveform consisting of up to 2^{14}, or 16,384 sequential data points. These subroutines, as well as all other programs in this report, are written in FORTRAN II and are implemented on the in-house time share computing system at NBS, Boulder, Colorado.

In order to compute a "forward" DFT, or to go from the time domain to the frequency domain, the following call sequence of subroutines is used:

```
CALL REORDER (A, B, MM)
CALL CFFTRC (A, B, MM, .5*SC, NX)
CALL REALTRAN (A, B, MM, NX, INV).
```

The notation used in these call statements relates to the DFT as follows:

$$S_r = A_r + jB_r = SC \sum_{k=o}^{N-1} X_k W_N^{(NX)rk}$$

where $NX = \pm 1$ (indicates exponent sign of W_N)

SC = real scaling factor

$INV = \pm 1$ (indicates forward or reverse transform)

$2^{(MM)+1}$ = length of transform

As an example, for the case of a 1,024 point DFT the specific call statements would be:

```
CALL REORDER (A, B, 9)
CALL CFFTRC (A, B, 9, .5*1., 1)
CALL REALTRAN (A, B, 9, 1, 1)
```

The sequence of call statements for the IDFT to return from the frequency domain to the original time domain waveform are:

```
CALL REALTRAN (A, B, MM, NX, INV)
CALL CFFTS (A, B, MM, 1./(N*SC), NX)
CALL REORDER (A, B, MM)
```

For the previous example, the sequence of call statements for the IDFT relating to the 1024 point original time series would be:

```
CALL REALTRAN (A, B, 9, -1, -1)
CALL CFFTS (A, B, 9, 1./1024., -1)
CALL REORDER (A, B, 9)
```

For the 1024 point case, SC is assumed to be unity; NX and INV are +1 for the DFT and -1 for the IDFT.

A source of possible notational confusion exists with the two arrays A and B since they are used to define both input and output data. For the DFT, or time domain to frequency domain case, A and B contain the N values of time domain data points as follows:

$$A_k = X_k \qquad k = 0, 1, 2\ldots, \frac{N}{2}-1 \qquad (2.2)$$

$$B_k = X\left(k + \frac{N}{2}\right) \qquad k = 0, 1, 2\ldots, \frac{N}{2}-1 \qquad (2.3)$$

Thus the first half of the time domain waveform is contained in A and the second half in B.

Once the DFT subroutines have run, the contents of A and B are replaced with the output data as follows:

$$S_r = A_r + jB_r \qquad r = 0, 1, 2, \ldots, \frac{N}{2} - 1 \tag{2.4}$$

where A_r and B_r are the real and imaginary parts, respectively, of the complex frequency domain coefficients. For a 1024 point time domain waveform, then, the first 512 points would be sequentially stored in A and the last 512 points in B. The resulting DFT solution would be 512 frequency coefficients each consisting of a real part, A, and an imaginary part, B.

Therefore, for each frequency component,

$$\text{MAGNITUDE}(r) = \sqrt{A_r^2 + B_r^2} \tag{2.5}$$

$$\text{PHASE}(r) = \arctan\left(\frac{B_r}{A_r}\right) \quad . \tag{2.6}$$

Subroutine listings are given for REORDER CFFTS, CFFTRC, and REALTRAN on pages 11-12, pages 13-14, pages 15-16, and page 17, respectively.

These four subroutines are part of an FFT subroutine package contained in the public user's subroutine library of the in-house time share computing system at NBS, Boulder, Colorado. For additional information concerning these subroutines see Ref. [5].

```
      SUBROUTINE REORDER(A,B,MM)
      DIMENSION A(1024),B(1024),LST(15),LC(15),ST(15)
      COMMON M,LC,ST
      M=MM
      CALL FFTC
      IF(M-200)500,170,500
500    CONTINUE
      IF(M)301,170,301
301   CONTINUE
      JA=M+1
      JB=M-1
      KB=0
      I=0
      KU=LC(JA)-1
       DO 20 KA = 1,KU,2
       T = A(KA+1)
       A(KA+1) = B(KA)
       B(KA) = T
 20    CONTINUE
      IF(M-1)302,170,302
302   CONTINUE
      LIM=M/2+1
 30    KS = LC(JB+1)+KB
       KU = KS
      JJ=LC(JA-JB)
       KK = KB + JJ
 40    K = KK + JJ
 50    T = A(KK+1)
       A(KK+1) = A(KS+1)
       A(KS+1) = T
       T = B(KK+1)
       B(KK+1) = B(KS+1)
       B(KS+1) = T
       KK = KK + 1
       KS = KS + 1
      IF(KK-K)50,303,303
303   CONTINUE
      KK=KK+JJ
       KS = KS + JJ
      IF(KK-KU)40,304,304
304   CONTINUE
      IF(JB-LIM)90,90,305
```

```
305    CONTINUE
       JB=JB-1
        I = I+1
        LST(I) = JB
        GO TO 30
90     IF(I)170,170,306
306    CONTINUE
       JB=LST(I)
        I = I - 1
        KB = KS
        GO TO 30
170    RETURN
         END
```

```
      SUBROUTINE CFFTS (A,B,MM,SCALE,NEXP)
      DIMENSION A(16384),B(16384),JC(15),SNT(15)
      COMMON M,JC,SNT
      M=MM
      MA=MM
      CALL FFTC
      IF(M-200)500,170,500
500   CONTINUE
      N=JC(M+1)
      NH=N/2
      NQ=NH/2
      SC = SCALE
      IF(M-2)301,302,302
301   IF(M)120,120,101
302   CONTINUE
      EXPS = ISIGN(1,NEXP)
      NN = N-1
      K = 1
      DO 100 JA = 2,M
      CE=SNT(MA)
      MA = MA - 1
      CD = 2. * CE * CE
      SD = -SNT(MA)
      R = -2. * CD
      CN = 1.
      CM = 0.
      SN = 0.
      JJ = 0
      KK = 1
      SM = +EXPS
 30   JSPAN = NH
      NH = JSPAN/2
 40   KS = KK + JSPAN
      RE = A(KK) - A(KS)
      A(KK) = A(KK) + A(KS)
      FIM = B(KK) - B(KS)
      B(KK) = B(KK) + B(KS)
      A(KS) = CN*RE -SN*FIM
      B(KS) = SN*RE + CN*FIM
      KK = KK + NH
      KS = KS + NH
```

```
      RE = A(KK) - A(KS)
      A(KK) = A(KK) + A(KS)
      FIM = B(KK) -B(KS)
      B(KK) = B(KK) + B(KS)
      A(KS) = CM*RE - SM*FIM
      B(KS) = SM * RE + CM * FIM
      KK = KS + NH
     IF(KK-N)40,303,303
303  CONTINUE
 50    KK = KK - NN
       JJ = JJ + K
     IF(JJ-NQ)304,90,90
304  CONTINUE
 60    CD = R * CN + CD
       CN = CD + CN
       SD = R * CM + SD
       CM = CM + SD
     SN=-CM*EXPS
     SM=CN*EXPS
       GO TO 40
 90    K = K + K
 100   CONTINUE
101    DO 110 KK = 1,N,2
       KS = KK + 1
       RE = A(KK) - A(KS)
       A(KK) =          (A(KK) + A(KS))
       A(KS) = RE
       FIM = B(KK) - B(KS)
       B(KK) =          (B(KK) + B(KS))
       B(KS) = FIM
 110   CONTINUE
120    IF(ABS(SC-1.)-1.E-10)135,305,305
305  CONTINUE
     DO 130 JB=1,N
       A(JB) = SC     * A(JB)
       B(JB) = SC     * B(JB)
 130   CONTINUE
 135   RETURN
170    RETURN
         END
```

```
"FFTS"
      SUBROUTINE CFFTRC (A,B,MM,SCALE,NEXP)
      DIMENSION A(1024),B(1024),JD(15),S(15)
      COMMON M,JD,S
      M=MM
      CALL FFTC
      IF(M-200)500,170,500
500     CONTINUE
      N=JD(M+1)
      K = N/4
      NQ = K
      NN = N-1
      JSPAN = 1
       SC = SCALE
      IF(ABS(SC-1.)-1.E-10)7,301,301
301    CONTINUE
 6     DO 5 JC = 1,N
       A(JC) = SC     * A(JC)
       B(JC) = SC     * B(JC)
 5     CONTINUE
7      IF(M)302,170,302
302   CONTINUE
      DO 10 KK = 1,N,2
      KS = KK+1
            RE = A(KK) - A(KS)
            A(KK) =           A(KK) + A(KS)
            A(KS) =         RE
            FIM   = B(KK) - B(KS)
            B(KK) =           B(KK) + B(KS)
            B(KS) = FIM
  10  CONTINUE
      IF(M-1)303,170,303
303   CONTINUE
       EXPS = ISIGN(1,NEXP)
       NP = 1
      DO 90 JB = 2,M
       SD = -S(JB-1)
       CD = 2.* S(JB) * S(JB)
       R = -2. * CD
            CN = 1.
            CM = 0.
            SN = 0.
            JJ = 0
            KK = 1
             SM = +EXPS
```

```
   12          JSPANH = JSPAN
               JSPAN = JSPAN + JSPAN
   20           KS = KK + JSPAN
               RE = CN * A(KS) - SN * B(KS)
               FIM = SN * A(KS) + CN * B(KS)
               A(KS) = A(KK) - RE
               A(KK) = A(KK) + RE
               B(KS) = B(KK) - FIM
               B(KK) = B(KK) + FIM
               KK = KK + JSPANH
               KS = KS + JSPANH
               FIM = SM * A(KS) + CM * B(KS)
               RE = CM * A(KS) - SM * B(KS)
               A(KS) = A(KK) - RE
               A(KK) = A(KK) + RE
               B(KS) = B(KK) - FIM
               B(KK) = B(KK) + FIM
               KK = KS + JSPANH
         IF(KK-N)20,304,304
304      CONTINUE
   30          KK = KK - NN
               JJ = JJ + K
         IF(JJ-NQ)305,80,80
305      CONTINUE
 40       CD = R * CN + CD
          CN = CD + CN
         SM=CN*EXPS
          SD = R * CM + SD
          CM = SD + CM
         SN=-CM*EXPS
               GO TO 20
   80 K = K/2
         RAD=.5*RAD
   90 CONTINUE
170      RETURN
          END
```

```
      SUBROUTINE REALTRAN(A,B,MM,NE,INV)
       DIMENSION A(1024),B(1024),JC(15),ST(15)
      COMMON M,JC,ST
1     M=MM
      CALL FFTC
      IF(M-200)500,170,500
500    CONTINUE
       K=JC(M+1)
       N=K
      NH=N/2
      NK=NH+1
      CN=ISIGN(1,INV)
      SN=ISIGN(1,NE)
      IF(CN)301,3,3
301   CONTINUE
2     FIM=A(1)-A(N+1)
      A(1)=A(1)+A(N+1)
      B(1)=FIM
      GO TO 4
3     A(N+1)=2.*(A(1)-B(1))
      A(1)=2.*(A(1)+B(1))
      B(N+1)=0
      B(1)=0
4     IF(M)302,170,302
302   CONTINUE
      SD=CN*SN*ST(M)
      R=2.*ST(M+1)
      R=-R*R
      CD=-.5*CN*R
      SN=0.
      DO 5 J=2,NK
            CD = R * CN + CD
            CN = CD + CN
            SD = R * SN + SD
            SN = SD + SN
            AA = A(J) + A(K)
            AB = A(J) - A(K)
            BA = B(J) + B(K)
            BB = B(J) - B(K)
            RE = CN * BA + SN * AB
            FIM = SN * BA - CN * AB
            B(K) = FIM - BB
            B(J) = FIM + BB
            A(K) = AA - RE
            A(J) = AA + RE
5     K=K-1
170   RETURN
        END
```

2.2 Input Data Programs

Two FORTRAN II programs are described in this section. They are used to modify the time domain waveform data stored in local memory, making these data suitable for frequency domain transformation.

The first program, stored in a user's file in the time share system under file name /XX1/, performs three functions. However, before a detailed description of these functions is presented, it is necessary to discuss the format of the data retrieval from local memory.

The time domain waveform data is stored locally in a 1000 point BCD memory. Each Y-axis data point consists of three decimal digits. The X-axis data is not needed since the Y data is stored and transmitted sequentially. This 1000 point waveform is converted to ASCII teletype code and put on paper tape. It is then read into the time share system from a paper tape reader and placed in file name /RAW/. Program /XX1/ is then implemented as follows:

1. The first function of /XX1/ is to shift the array indexing of /RAW/. An idiosyncracy of the hardware is that the first Y data point appears on paper tape three times in succession. Thus it is necessary to delete the first two data points in file /RAW/. This is accomplished by index shifting such that M(1) = M(3), M(2) = M(4), etc.
2. At this point, /RAW/ consists of 1000 three-digit Y data points representing a time domain waveform. The FFT algorithm requires the waveform data array to be of length 2^N, where N is an integer. The closest length is 1024 data points so 24 points must be added to the array. This may be accomplished in any suitable manner, but in order

to introduce a minimum of error to the data, the following method has been chosen. The average value of the last ten data points is computed. This value is then used to fill the array by placing it in positions 1001 thru 1024. In other words, the averaged final value of the waveform is simply repeated 24 times.

3. The third program function is a form of signal averaging. This portion of the program is optional. If an unaveraged waveform array is desired it is stored in file name /DON/. If the averaging process is required, the data output appears in file name /DONER/. Waveform averaging is accomplished by comparing the value of each data point with the average value of its neighbors on either side. If the value of the tested data point exceeds certain limits, that value is changed to equal the neighbors' averaged value. See Figure 2.1.

Program /XX1/ receives values for NR and NL from the TTY keyboard at run time. From Figure 2.1 it is apparent that NR is the total number of neighbors nearest the test point. In this example NR = 12, or six on either side of the test point. LOAV is the computed average value of the six data points to the left of X_k, and LIAV, the average of the six points to the right of X_k. The average value of LOAV and LIAV then is computed and stored as LOT. Therefore, LOT is the average value of the NR nearest neighbors of X_k. The absolute value of X_k-LOT is then compared with the value chosen for NL. If $|X_k - LOT|$ exceeds NL, then X_k is reset equal to LOT. If $|X_k - LOT|$ is less than or equal to NL, then X_k is unchanged. NR must be a positive even integer and NL, a positive integer.

Figure 2-2 shows a typical time domain waveform along with the results of averaging for various values of NR and NL using Program

/XX1/ The program listing is as follows:

NOTE: Because a detailed explanation of both the input data and the output data programs is contained in the text, comment statements are not included in the program listings.

```
 /XX1/
       DIMENSION M(1024)
       DIMENSION L(1024)
       OPEN (3,INPUT,/RAW/)
       OPEN (4,OUTPUT,/DONER/)
       READ 3,10,(M(K),K=1,1003)
       TYPE 25
25     FORMAT (10HNL AND NR?)
       ACCEPT 15,NL,NR
15     FORMAT (2I2)
       DO 5 I=1,1000
5      M(I)=M(I+2)
       SUM=0
       DO 7 I=991,1000
7      SUM=SUM+M(I)
       AVG=SUM/10
       DO 8 I=1001,1024
8      M(I)=AVG
       CLOSE (3)
       OPEN (5,OUTPUT,/DON/)
       DO 17 J=1,1024
17     WRITE 5,20,(M(J))
10     FORMAT (1003(I3/))
20     FORMAT (1024(I3/))
       DO 35 K=1,1024
35     L(K)=M(K)
       DO 40 J=1,(1024-NR)
       LO=0
       LI=0
       DO 60 K=J,(J+(NR/2)-1)
60     LO=LO+L(K)
       DO 70 K=(J+(NR/2)+1),J+NR
70     LI=LI+L(K)
       LOAV=LO/(NR/2)
       LIAV=LI/(NR/2)
       LOT=(LOAV+LIAV)/2
       IF(IABS(L(J+(NR/2))-LOT)-NL)40,40,21
21     L(J+(NR/2))=LOT
40     CONTINUE
       DO 80 J=1,1024
80     WRITE 4,20,(L(J))
       STOP
       END
```

-

USCOMM - ERL

The second data input program is called /MOD1/. The input data to this program is listed in file name /DONER/ and is the output from program /XX1/. Since the FFT algorithm will produce erroneous results for non-periodic waveform inputs, this program insures that the input waveform will appear periodic. This is accomplished by first deleting all the odd-indexed data points in the input waveform. Then the resulting 512 point waveform is inverted, vertically shifted, and placed in indices 513 through 1024. See Figure 2.3 for a sample waveform both before and after being made periodic.

The output of this program is then in proper form to be transformed using the FFT subroutines. The /MOD1/ listing is as follows:

```
/MOD1/
      DIMENSION K(1024)
      DIMENSION L(1024)
      DIMENSION M(1024)
      DIMENSION N(1024)
      OPEN (3,INPUT,/DONER/)
      READ 3,10,(M(J),J=1,1024)
10    FORMAT (I3)
      DO 5 J=1,512
5     N(J)=M(J*2)
      ISUM=0
      DO 20 J=1,10
20    ISUM=ISUM+N(J)
      IAVG=ISUM/10
      ISHIFT=N(511)-IAVG
      DO 30 J=1,512
30    L(J)=N(J)
      DO 40 J=1,512
40    K(J)=2*N(511)-L(J)-ISHIFT
      DO 50 J=1,512
50    N(J+512)=K(J)
      OPEN (4,OUTPUT,/DONE/)
      DO 70 J=1,1024
70    WRITE 4,60,N(J)
60    FORMAT (I3/)
      STOP
      END
```

2.3 Output Data Programs

The first data output program is called /TRI/. This program accepts the output of /MOD1/, a 1024 point averaged, periodic time-domain waveform stored in file name /DONE/ and performs the FFT of this waveform. The data output from this program is available in the following files:

$$/\text{DONEA}/ = A(r)$$

$$/\text{DONEB}/ = B(r)$$

$$/\text{DONEX}/ = \frac{1}{512} \left\{ [A(r)]^2 + [B(r)]^2 \right\}^{1/2}$$

where $S(r) = A(r) + jB(r)$

and $S(r)$ are the 512 unscaled complex frequency coefficients associated with the 1024 point input waveform. /DONEX/ is really a scaled discreet magnitude spectrum of the input waveform. Scaling is accomplished by multiplying each $A(r)$ and $B(r)$ (with the exception of $A(0)$) by $2/N$ where N is the length of the transform. In this case $N = 1024$. $A(0)$, or the D. C. term, must be multiplied by $1/N$ for proper scaling.

The second data output program is called /LSP/ and is used to scale the output magnitude spectrum /DONEX/ so it may be returned to local memory and displayed on a CRT. The scaling equation used to yield an integer spectrum in the range $0 \leq M(r) < 1000$ is:

$$M(r) = B'(r) = 200 \log_{10} [100 \, |S(r)|]$$

where $|S(r)|$ are the scaled magnitudes of the complex frequency coefficients, $B'(r)$ are the rescaled decibel values of $|S(r)|$, and $M(r)$ are the nearest integer values of $B'(r)$. See Figure 2.4 for some sample displays of magnitude spectra in dB versus frequency.

The third data output program, called /UNTRI/, is used to perform the inverse FFT on frequency domain data. The inputs to this program are the unscaled complex frequency coefficients from /TRI/. They are contained in file names /DONEA/ and /DONEB/. The outputs of this program, contained in /Y1/ and /Y2/, are the first and last half, respectively, of the time domain waveform associated with /DONEA/ and /DONEB/. The /UNTRI/ program was used only to check the validity of the FFT program package. The program listings for /TRI/, /UNTRI/ and /LSP/ are as follows:

```
  /TRI/
        DIMENSION M(1024),A(513),B(513),C(1024),D(513)
        OPEN (3,INPUT,/DONE/)
        READ 3,10,(M(K),K=1,1024)
10      FORMAT (I3)
        DO 40 J=1,1024
40      C(J)=M(J)
        DO 50 J=1,512
50      A(J)=C(J)
        DO 60 J=1,512
60      B(J)=C(J+512)
30      FORMAT (F11.2)
        CALL REORDER (A,B,9)
        CALL CFFTRC (A,B,9,.5*1.,1)
        CALL REALTRAN (A,B,9,1,1)
        CLOSE (3)
        OPEN (4,OUTPUT,/DONEA/)
        WRITE 4,30,(A(J),J=1,513)
        OPEN (5,OUTPUT,/DONEB/)
        WRITE 5,30,(B(J),J=1,513)
        DO 100 J=1,513
100     D(J)=(SQRT((A(J)**2)+(B(J)**2)))/512.
        CLOSE (4)
        CLOSE (5)
        OPEN (6,OUTPUT,/DONEX/)
        WRITE 6,120,(D(J),J=1,513)
120     FORMAT (F7.2)
        STOP
        END
```

```
-/UNTRI/
      DIMENSION A(513),B(513)
      OPEN (3,INPUT,/DONEA/)
      OPEN (4,INPUT,/DONEB/)
      READ 3,10,(A(K),K=1,513)
      READ 4,10,(B(K),K=1,513)
10    FORMAT (F11.2)
      CALL REALTRAN (A,B,9,-1,-1)
      CALL CFFTS (A,B,9,1./1024.,-1)
      CALL REORDER (A,B,9)
      CLOSE (3)
      CLOSE (4)
      OPEN (5,OUTPUT,/Y1/)
      OPEN (6,OUTPUT,/Y2/)
      WRITE 5,10,(A(K),K=1,513)
      WRITE 6,10,(B(K),K=1,513)
      STOP
      END
```

-

NOTE: In the listing which follows B(K) represents the term B′(r) as defined on page 22.

```
 /LSP/
      DIMENSION A(513),B(1024),M(1024)
      OPEN (3,INPUT,/DONEX/)
      READ 3,10,(A(K),K=1,513)
      A(1)=0.
      DO 15 K=1,256
15    A(2*K)=200.*(ALOG(100.*A(2*K)))/(ALOG(10.0))
      DO 25 K=1,512
      B(2*K-1)=0.
25    B(2*K)=A(K)
      DO 35 K=1,1024
35    M(K)=B(K)
      OPEN (4,OUTPUT,/SPEC/)
      WRITE 4,20,(M(K),K=1,1024)
      OPEN (5,OUTPUT,/SHORT/)
      WRITE 5,20,(M(4*K),K=1,256)
10    FORMAT (F7.2)
20    FORMAT (I3)
      STOP
      END
```

3. EXPERIMENTAL CONCLUSIONS

3.1 Problems Encountered Using the FFT

The primary goal of the work upon which this report is based is to develop both hardware and software for a particular system. At this point in time, the main desired function of the system is to produce the gain versus frequency characteristics of certain electrical networks using time domain techniques. As a result, only specific FFT application problems, pertinent to this particular system, will be discussed. Problems of a more theoretical nature are discussed in detail in the literature, e.g., [3] or [4].

Basically, the system works in the following manner. A small transition-time voltage step-waveform is generated, observed with a sampling oscilloscope, and the oscilloscope analog output recorded. The FFT of the recorded waveform is then computed, and a magnitude versus frequency graph is recorded. The voltage step-waveform is then applied to the input terminals of the network under test (e.g., a coaxial attenuator); the network output waveform is observed with the sampling oscilloscope, recorded and transformed. The logarithmic difference of these two transforms, then, is computed to yield the gain versus frequency characteristics of the network.

Actually, four major problems or limitations were encountered which had a detrimental effect on network measurements. Each problem shall be discussed individually.

The first problem was due to the choice of a voltage step as an input waveform. The FFT algorithm assumes the input waveform to be periodic. Thus an unwanted discontinuity will appear between the beginning and end points of the step waveform, implying that the voltage step was instantaneously turned off. The FFT will transform the instantaneous transition as well as the valid voltage step waveform data, yielding a grossly invalid frequency spectrum.

The objective, then, was to make the step waveform appear periodic while introducing neither discontinuities nor other non-causal voltage changes. This was accomplished through the use of FORTRAN II program /MOD 1/. This program modifies the input step waveform in such a manner that it appears to the FFT as if the voltage step were turned off in exactly the same manner in which it was turned on. (See Figure 2.3). Consequently, the modified waveform appears periodic with no discontinuities and no new non-causal data added to it.

The second problem was also solved using program /MOD 1/. Figure 3.1 consists of two photographs of the magnitude spectra in dB of two ideal rectangular pulses of arbitrary width. These spectra are both difficult to interpret since they are made up of two curves superimposed upon one another. In fact, one curve on each photograph represents the magnitude spectrum of the even harmonics while the other curve represents the magnitude spectrum of the odd harmonics.

It was found that the manner in which program /MOD 1/ produces a periodic waveform from a voltage step waveform also ensures that the even harmonic magnitude spectra will be forced to zero and the odd harmonic magnitude spectra will decrease monotonically as in Figure 3.2. In terms of Fourier series analysis, program /MOD 1/ generates a single period of a periodic waveform in which

$$f(t) = -f\left(t + \frac{T}{2}\right) \qquad (3.1)$$

where T is the period of the program-generated waveform. From Fourier series analysis, any periodic waveform that fulfills equation (3.1) will contain only odd harmonics.

The third problem is really a limitation inherent in the use of a step waveform as a driving function. The step waveform must be made periodic in order to satisfy the requirements of the FFT. Thus, the

modified waveform will always resemble a rectangular pulse. But the Fourier integral transform of a rectangular pulse is of the form six x/x where x is a function of both frequency and the width of the time window. Consequently, there is an inherent 1/x decay of signal input level as a function of frequency. This signal level decay may be overcome somewhat by careful selection of the time window used, and possibly by pulsed RF techniques.

The fourth major problem is that of noise. This problem is not caused by use of the FFT, but rather, it is caused by the data acquisition system. It is worth mentioning because several software problems result in processing the resultant noisy signals. The magnitude spectra shown in Figure 2.4 are noisy spectra typically encountered. As may be seen, only the very beginning of each spectrum appears to show any continuity. The rest is much too noisy to be interpreted.

Aside from attempts to diminish the noise sources within the hardware, two software techniques were used to bring the noise level down. One was signal averaging as in FORTRAN II program /XX1/. The other was taking transforms over several decreasingly smaller time windows to obtain usable frequency domain data over an increasingly wider frequency band.

3.2 Time Domain Measurement Example

In this section the measurement of the gain versus frequency characteristics of a coaxial 10 MHz low-pass filter is presented. Figure 3.3 is a block diagram of the waveform data acquisition system used to perform this measurement. For purposes of this report, this data system simply acquires and stores in local memory a 1000 point representation of a time domain waveform. The input step waveform and the low-pass filter output waveform are shown in Figure 3.4. In the example, the input waveform is unsmoothed while the output waveform has been smoothed using FORTRAN II program /XX1/.

The FFT software programs were then applied to both waveforms and the logarithmic magnitude versus frequency spectrum of each waveform was recorded in local memory. An X-Y recording of these two spectra is shown in Figure 3.5. This figure clearly shows the 1/x decay characteristics of the transformed step waveform. Also, it exhibits a relatively noiseless gain window of about 60 dB below the fundamental frequency amplitude.

Figure 3.6 is an X-Y recording of the algebraic difference of the input and output spectra shown in Figure 3.5. This figure shows a fairly flat response of the low-pass filter out to approximately 9 MHz. The gain then drops at a rate of about 20 dB/MHz into the noise.

The results of this measurement appear to be promising. The purpose of this measurement was merely a qualitative check of the basic system hardware and software, but the resulting curve of Figure 3.6 shows a noise level and a frequency resolution that indicates the feasibility of the time-domain calibration system.

3.3 Future Developmental Considerations

With the goal in mind of developing a time domain network calibration system competitive with current capabilities of frequency domain calibration systems, much more work is necessary. Outlined below is a schedule of tasks deemed necessary to meet this goal.

1. The first task, that of implementing a dedicated minicomputer into the system, is already underway. The minicomputer will allow much higher speeds of data handling and processing as well as increased system automation capabilities.
2. After implementation of the minicomputer, a more detailed and exhaustive set of experiments aimed at error analysis may be performed. At present, use of the time share computing system is much too time consuming to permit an in-depth study of systematic or random errors.

3. With sources of errors pinpointed, software may then be developed as far as possible both to determine the magnitude of and to reduce the effects of those errors.

4. Concurrent with software development, the fourth task would be hardware development. This would involve development of fast-rise noiseless step generators, improved sampling systems, increased data acquisition resolution, and increased system automation.

The general concept of using the FFT to compute the gain versus frequency characteristics of electrical networks appears quite feasible. Most probably, the real limitations on measurement precision and accuracy lie in system hardware, and not in FFT programming or other software.

This report has been concerned with three main topics: a definition of the fast Fourier transform; a description of software programs used to compute the FFT on the in-house time share computing system; and a brief description of problems encountered in the data reduction process used.

The next report on this continuing work will be addressed to the actual use of the system for quantitative measurements of amplitude and phase characteristics for microwave attenuators, filters, and mismatches. These measurements will also be compared to those made at other laboratories using time domain techniques (e. g., see Ref. [6] and [7]).

REFERENCES

1. Cooley, J. W. and Tukey, J. W., "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computers, Vol. 19, pp. 297-301, April, 1965.
2. Cochran, W. T., et al., "What is the Fast Fourier Transform?", IEEE Trans. on Audio and Electroacoustics, Vol. AU-15, No. 2, June, 1967.
3. Cooley, J. W., et al., "Application of the Fast Fourier Transform to Computation of Fourier Integrals, Fourier Series, and Convolution Integrals", IEEE Trans. on Audio and Electroacoustics, Vol. AU-15, No. 2, pp. 79-84, June 1967.
4. Bergland, G. D., "A Guided Tour of the Fast Fourier Transform", IEEE Spectrum, pp. 41-51, July, 1969.
5. FFT subroutine package written by David L. Lewis, National Oceanic and Atmospheric Administration, Space Environment Laboratory, Boulder, Colorado 80302.
6. Nicholson, M. A., et al., "Applications of Time-Domain Metrology to the Automation of Broad-Band Microwave Measurements", IEEE Trans. on Microwave Theory and Techniques, Vol. MTT-20, No. 1, January 1972.
7. Cronson, H. M., et al., "Time Domain Measurement Study", Sperry Rand Research Center, Sudbury, Massachusetts, August, 1972.

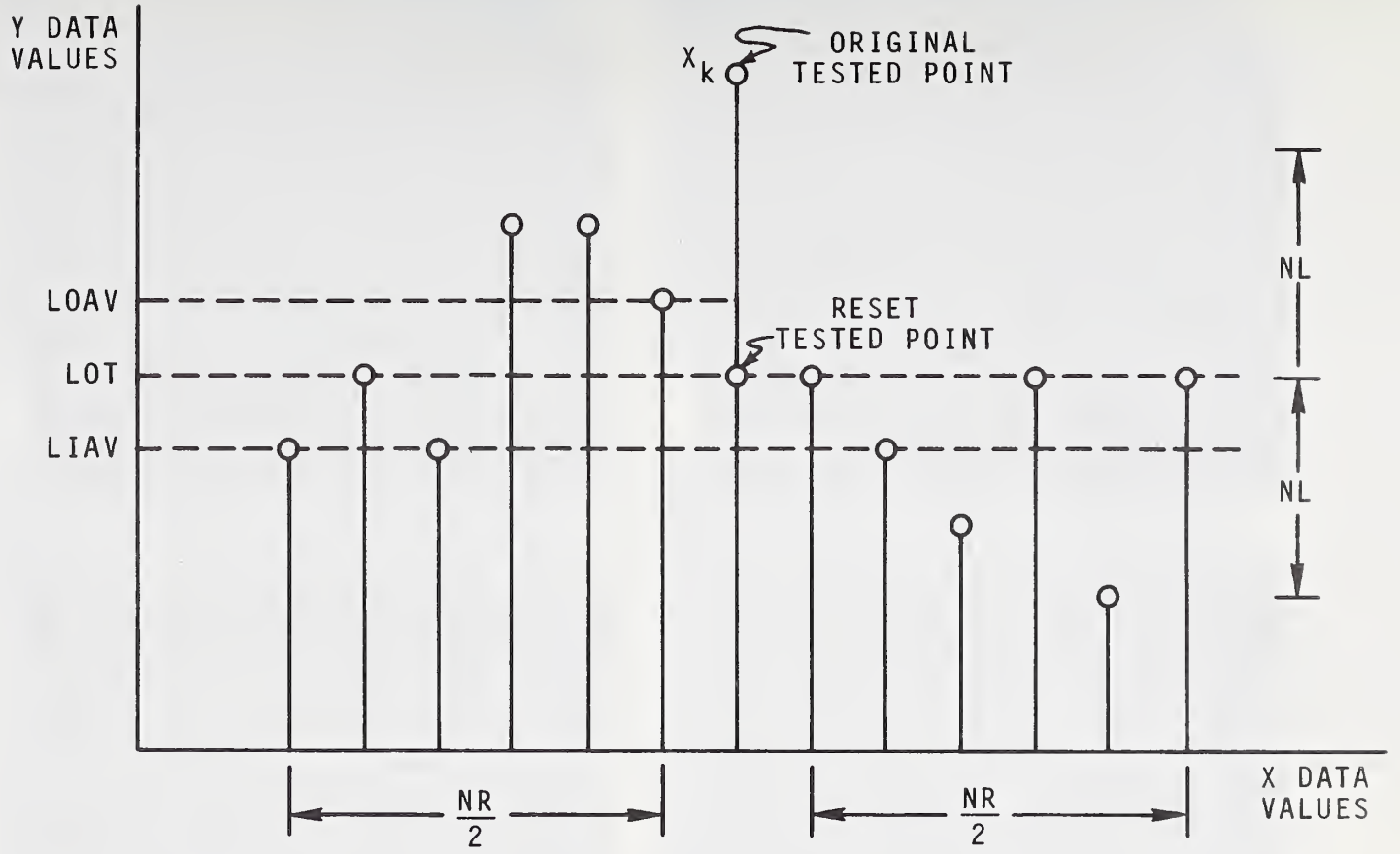
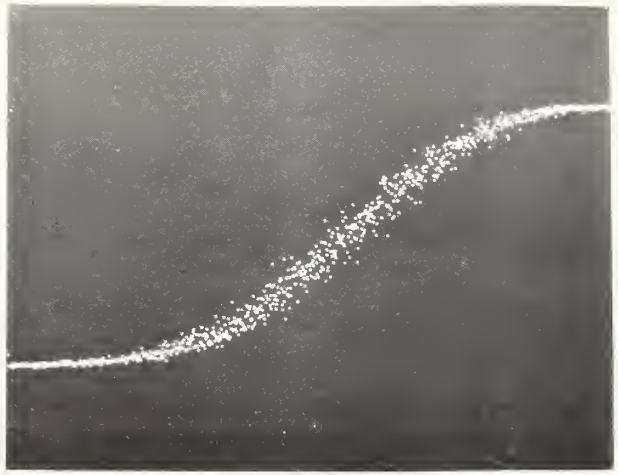
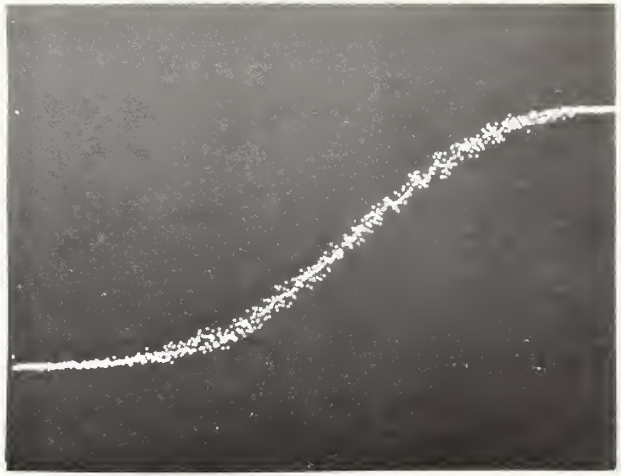


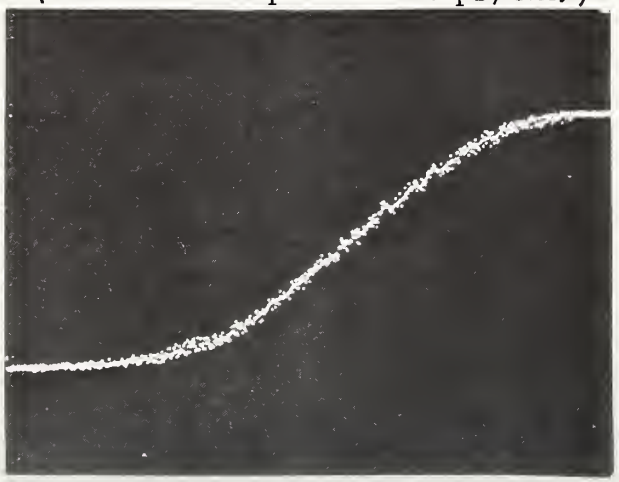
Fig. 2.1 Graphical interpretation of waveform averaging process.



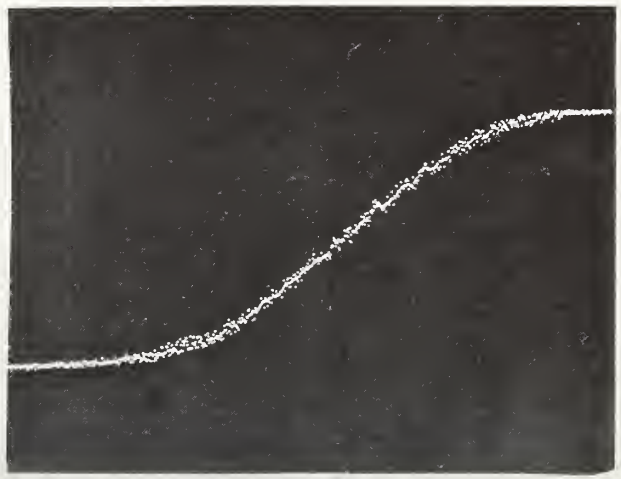
A. Original noisy time domain waveform (Tunnel diode pulse at 10 ps/cm.)



B. NR= 20, NL = 25

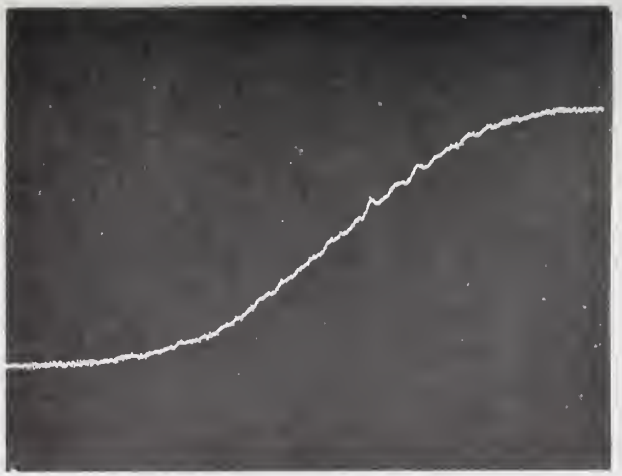


C. NR = 20, NL = 15

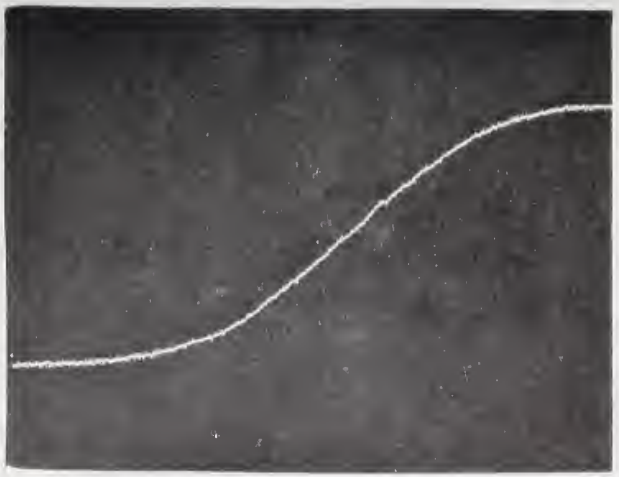


D. NR = 20, NL = 5

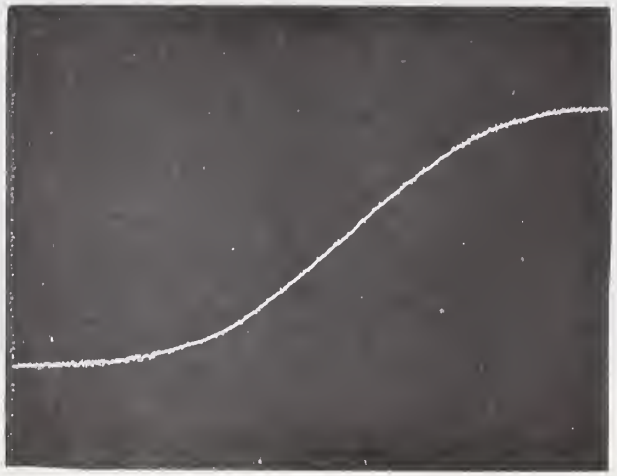
Fig. 2.2 Typical noisy time domain waveform and various results of waveform averaging.



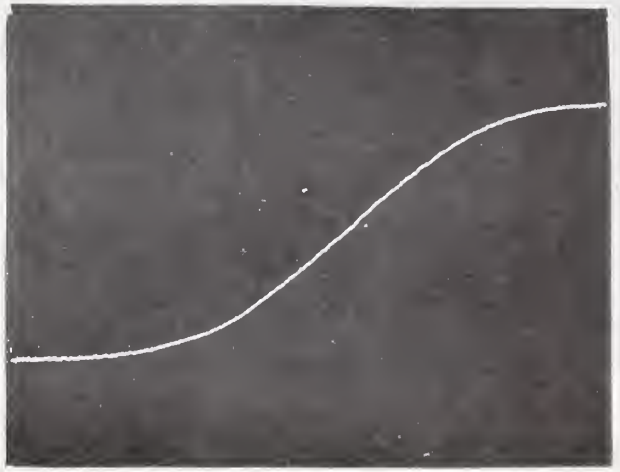
E. NR = 20, NL = 2



F. NR = 50, NL = 5

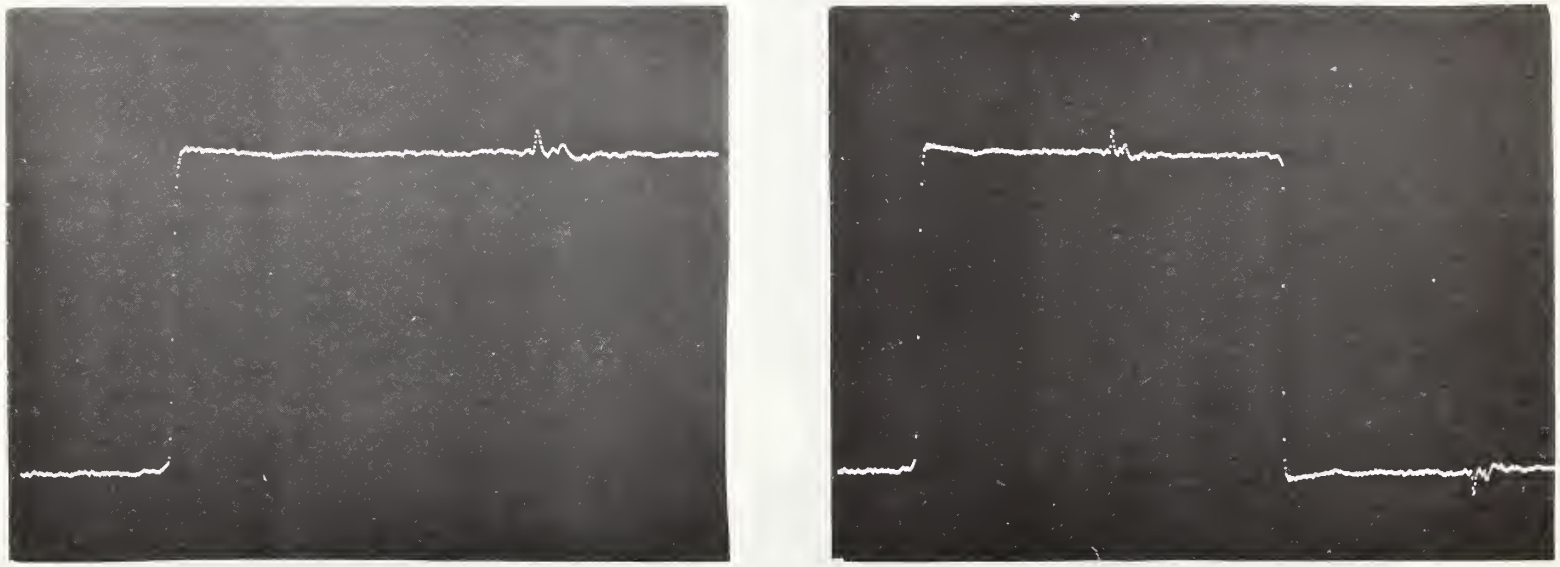


G. NR = 80, NL = 5



H. NR = 80, NL = 2

Fig. 2.2 Typical noisy time domain waveform and various results of waveform averaging (continued).



A. Non-periodic waveform

B. Waveform made periodic

Fig. 2.3 Time domain waveform, both before and after being made periodic.

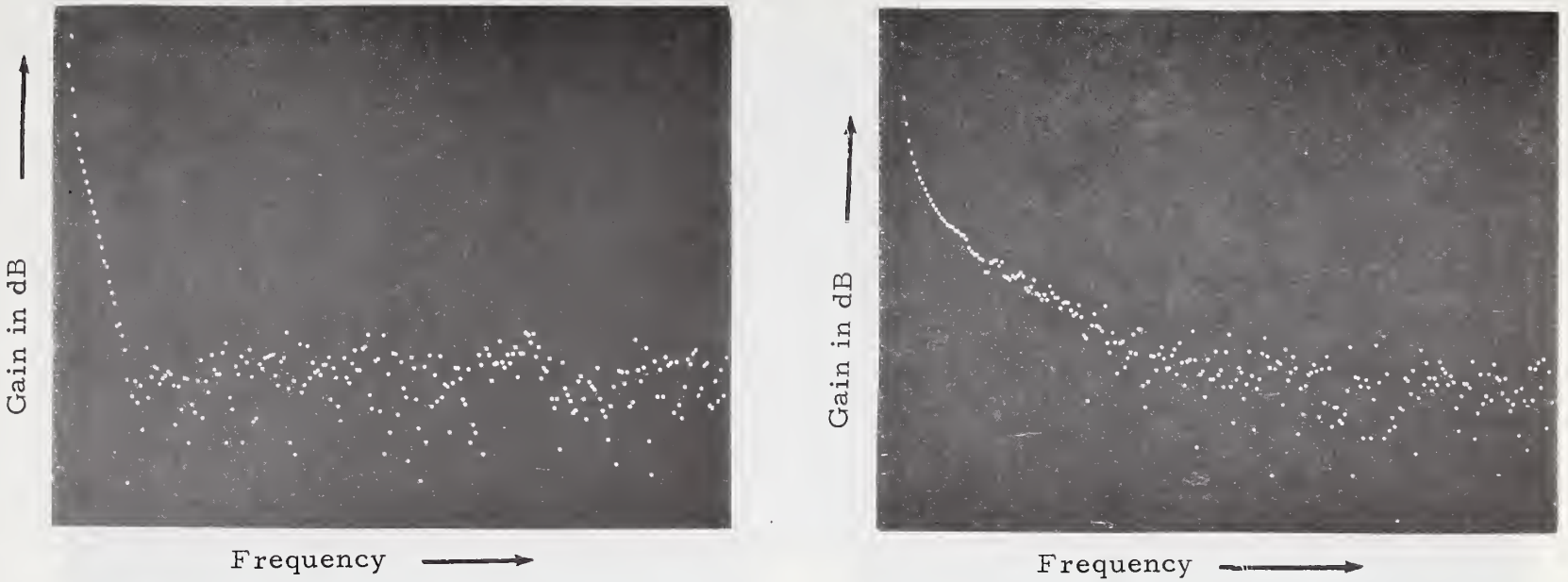


Fig. 2.4 Typical displays of magnitude versus frequency spectra for time domain waveforms.

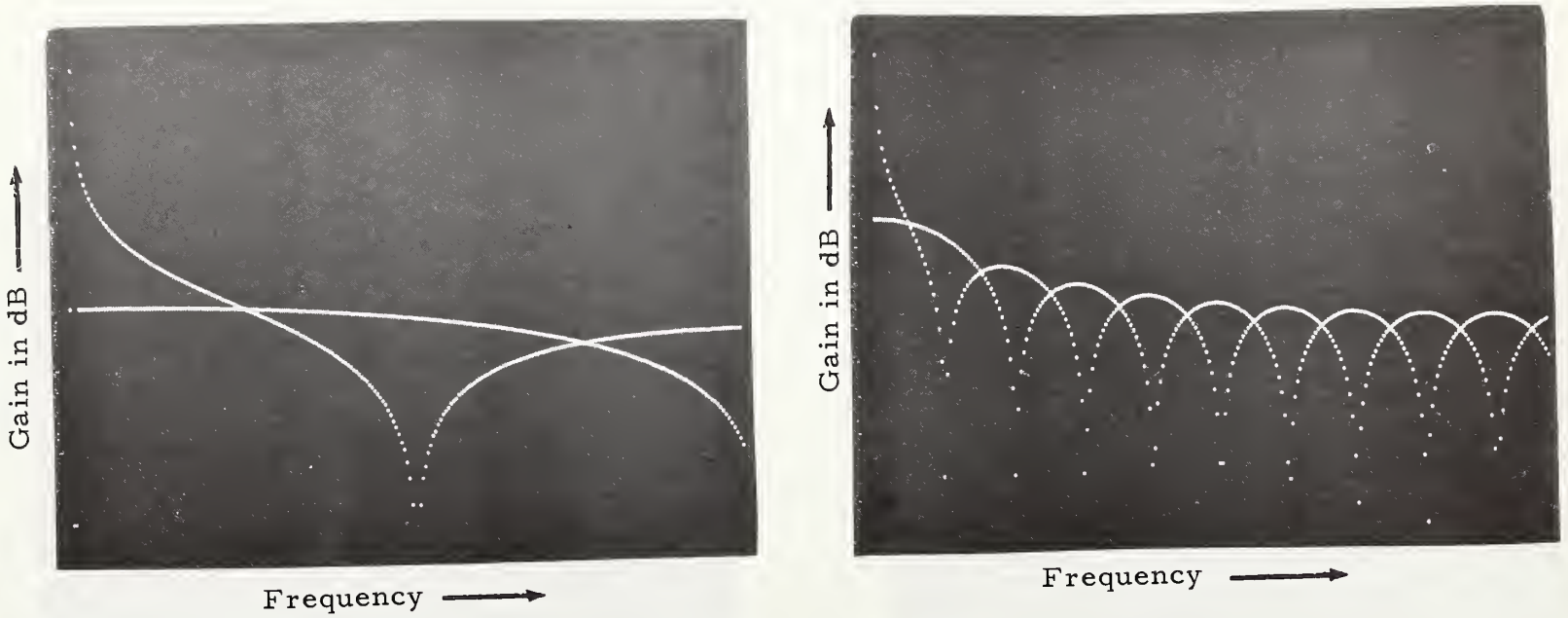


Fig. 3.1 Two possible magnitude spectra of an ideal rectangular pulse.

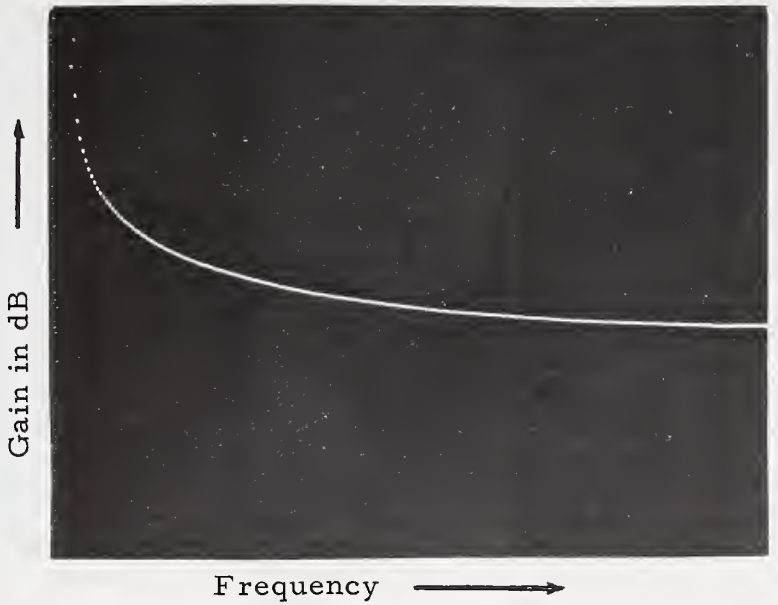


Fig. 3.2 Magnitude versus frequency spectrum of an ideal rectangular pulse with sin x variation removed.

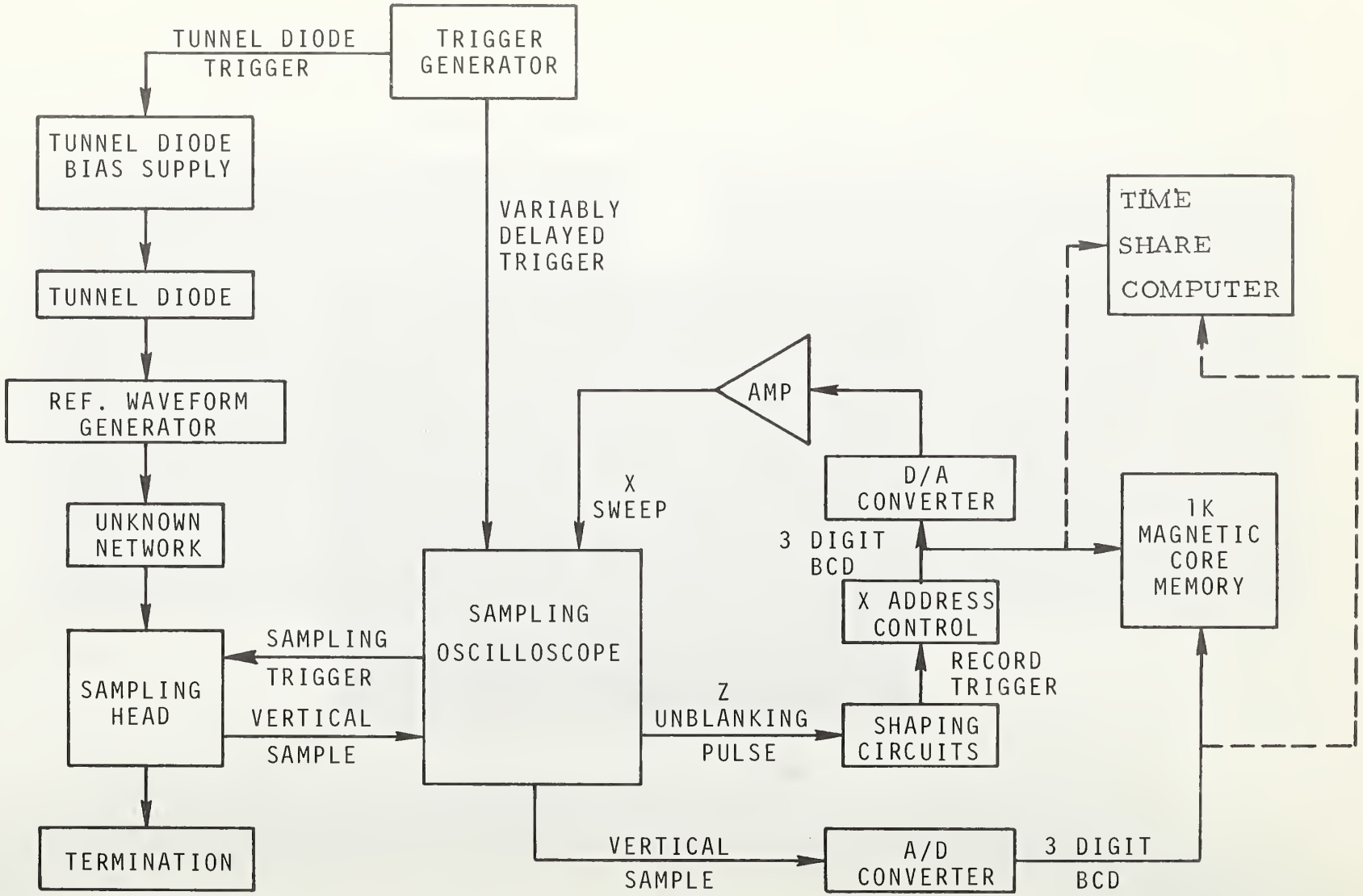
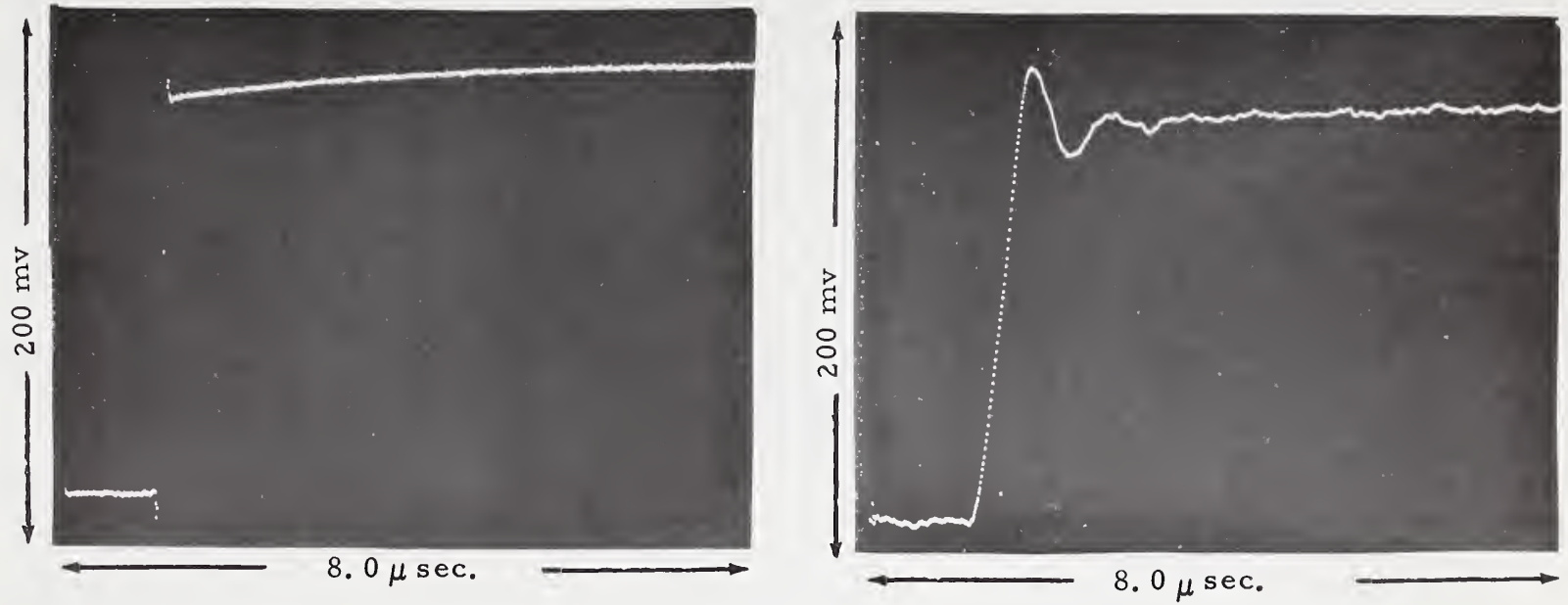


Fig. 3.3 Block diagram of time domain measurement system.



A. Input waveform to filter

B. Filter output waveform

Fig. 3.4 Input and output waveforms from 10 MHz low pass filter experiment.

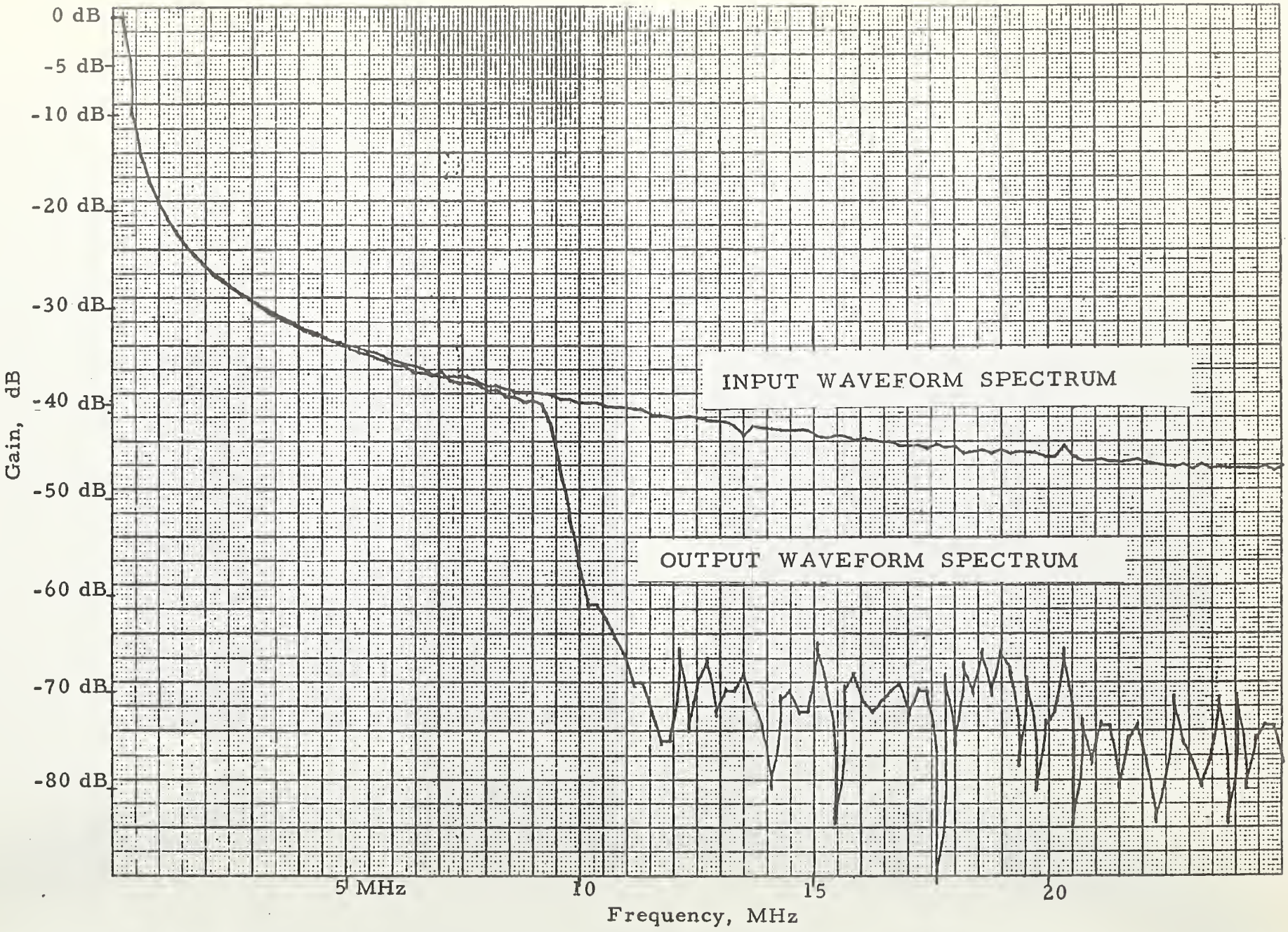


Fig. 3.5 Input and output waveform spectra for 10 MHz low pass filter.

B-104279

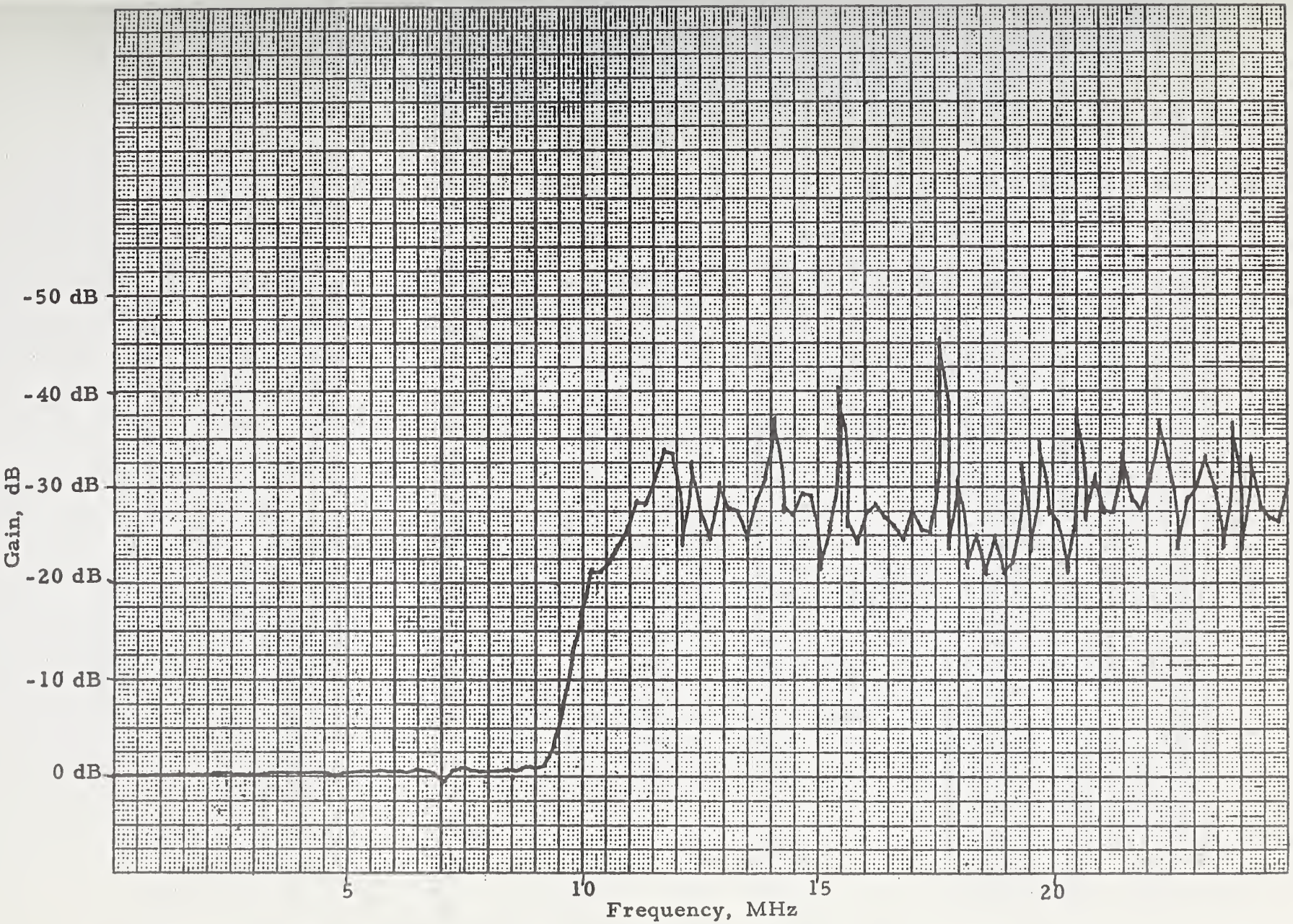


Fig. 3.6 Plot of algebraic difference between input and output spectra for 10 MHz low pass filter.

APPENDIX A

To yield some insight into the mechanisms of the FFT, an eight-point DFT will be solved below using the FFT algorithm.

Given a time domain waveform represented by eight equally time-spaced data points, as illustrated in Figure A-1, we wish to compute the DFT for this waveform. In particular, we wish to calculate the eight A_r's associated with the eight assumed X_k's. The DFT for the set of data points is,

$$A_r = \sum_{k=o}^{7} X_k e^{\frac{-2\pi jrk}{8}} \qquad r = 0, 1, 2, \ldots, 7 \qquad \text{(A-1)}$$

For the first reduction of transform length we let

$$B_r = \sum_{k=o}^{3} Y_k e^{\frac{-4\pi jrk}{8}} \qquad r = 0, 1, 2, 3 \qquad \text{(A-2)}$$

and

$$C_r = \sum_{k=o}^{3} Z_k e^{\frac{-4\pi jrk}{8}} \qquad r = 0, 1, 2, 3 \qquad \text{(A-3)}$$

where $$Y_k = X_{2k} \qquad \text{(A-4)}$$

and $$Z_k = X_{2k+1}. \qquad \text{(A-5)}$$

Using the simplifying notation that

$$W = e^{\frac{-2\pi j}{N}} \qquad \text{(A-6)}$$

we have

$$A_r = \sum_{k=o}^{7} X_k W^{rk} \qquad r = 0, 1, 2, \ldots, 7 \qquad \text{(A-7)}$$

and $$B_r = \sum_{k=o}^{3} Y_k W^{2rk} \qquad r = 0, 1, 2, 3. \qquad \text{(A-8)}$$

$$C_r = \sum_{k=o}^{3} Z_k W^{2rk} \qquad r = 0, 1, 2, 3. \qquad \text{(A-9)}$$

Thus, we have now reduced the problem to that of finding two four-point DFT's rather than one eight-point DFT. (See Figure A-2).

For the second reduction, B_r and C_r may each be reduced to two two-point transforms as follows:

Let

$$D_r = \sum_{k=o}^{1} R_k W^{4rk} \qquad r = 0, 1 \qquad \text{(A-10)}$$

and $$E_r = \sum_{k=o}^{1} S_k W^{4rk} \qquad r = 0, 1 \qquad \text{(A-11)}$$

where $$R_k = Y_{2k} = X_{4k} \qquad \text{(A-12)}$$

and $$S_k = Y_{2k+1} = X_{4k+2} \qquad \text{(A-13)}$$

and let

$$F_r = \sum_{k=o}^{1} T_k W^{4rk} \qquad r = 0, 1 \qquad \text{(A-14)}$$

$$G_r = \sum_{k=o}^{1} V_k W^{4rk} \qquad r = 0, 1 \qquad \text{(A-15)}$$

where $$T_k = Z_{2k} = X_{4k+1} \qquad \text{(A-16)}$$

and $$V_k = Z_{2k+1} = X_{4k+3}. \qquad \text{(A-17)}$$

Then D_r is the two point transform of data points X_o and X_4, E_r, the transform of X_2 and X_6, F_r, that of X_1 and X_5, and G_r, that of X_3 and X_7.

The third reduction, which would yield eight one-point transforms, is really unnecessary for the DFT of a single point function is the sample itself. Therefore, substitution of the data points into (A-10), (A-11), (A-14), and (A-15) yields the following set of equations.

$$\begin{aligned} D_o &= X_o + W^o X_4 & F_o &= X_1 + W^o X_5 \\ D_1 &= X_o - W^o X_4 & F_1 &= X_1 - W^o X_5 \end{aligned} \qquad \text{(A-18)}$$

$$\begin{aligned} E_o &= X_2 + W^o X_6 & G_o &= X_3 + W^o X_7 \\ E_1 &= X_2 - W^o X_6 & G_1 &= X_3 - W^o X_7 \end{aligned}$$

As was shown in the text,

$$A_r = B_r + W^r C_r \qquad r = 0, 1, 2, 3 \qquad \text{(A-19)}$$

and

$$A_{\left(r+\frac{N}{2}\right)} = B_r - W^r C_r \qquad r = 0, 1, 2, 3 \qquad \text{(A-20)}$$

Likewise, it may be shown that

$$B_r = D_r + W^{2r} E_r \qquad r = 0, 1 \qquad \text{(A-21)}$$

$$B_{\left(r+\frac{N}{4}\right)} = D_r - W^{2r} E_r \qquad r = 0, 1 \qquad \text{(A-22)}$$

and

$$C_r = F_r + W^{2r} G_r \qquad r = 0, 1 \qquad \text{(A-23)}$$

$$C_{\left(r+\frac{N}{4}\right)} = F_r - W^{2r} G_r \qquad r = 0, 1 \qquad \text{(A-24)}$$

Recalling that

$$W^r = -W^{\left(r-\frac{N}{2}\right)} \qquad \text{(A-25)}$$

we may substitute (A-18) into (A-21) and (A-22) and then into (A-19) and (A-20) to yield the following matrix equation:

$$\begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 & -1 & -W^1 & -W^2 & -W^3 \\ 1 & W^2 & -1 & -W^2 & 1 & W^2 & -1 & -W^2 \\ 1 & W^3 & -W^2 & +W^1 & -1 & -W^3 & W^2 & -W^1 \\ 1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\ 1 & -W^1 & W^2 & -W^3 & -1 & W^1 & -W^2 & W^3 \\ 1 & -W^2 & -1 & W^2 & 1 & -W^2 & -1 & W^2 \\ 1 & -W^3 & -W^2 & -W^1 & -1 & W^3 & W^2 & +W^1 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} \qquad \text{(A-26)}$$

Examining (A-26) we see that half of the multiplying matrix terms are unity. In addition, the matrix is symmetrical about the main diagonal. Consequently, about three-fourths of the multiplications are unnecessary. Figure A-3 is a signal flow graph illustrating this computation.

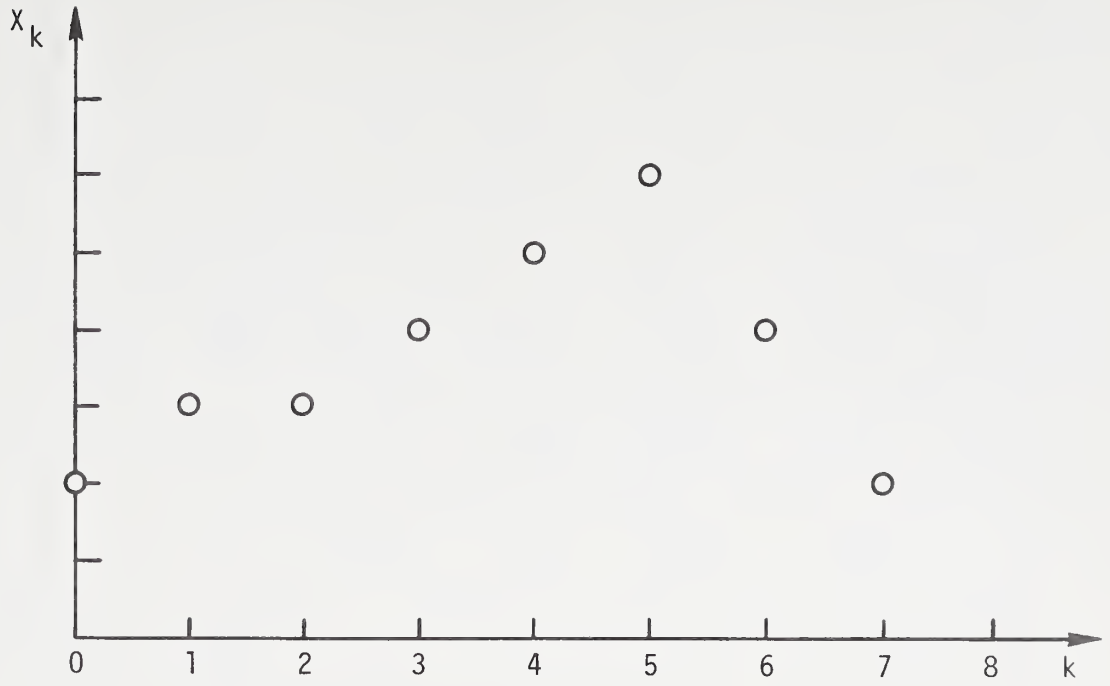


Fig. A-1 Eight-point representation of a time domain waveform.

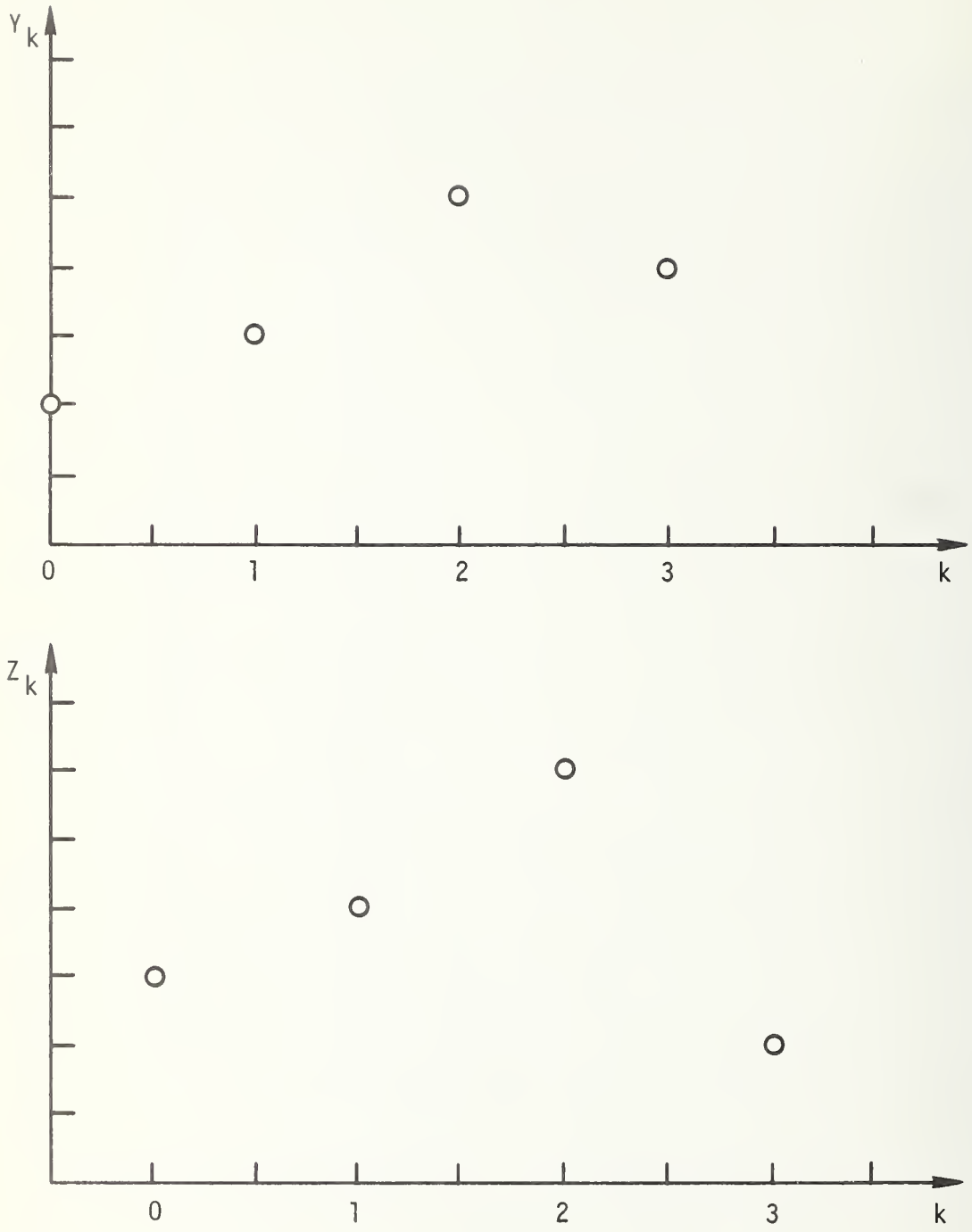


Fig. A-2 Two four-point representations of the same time domain waveform.

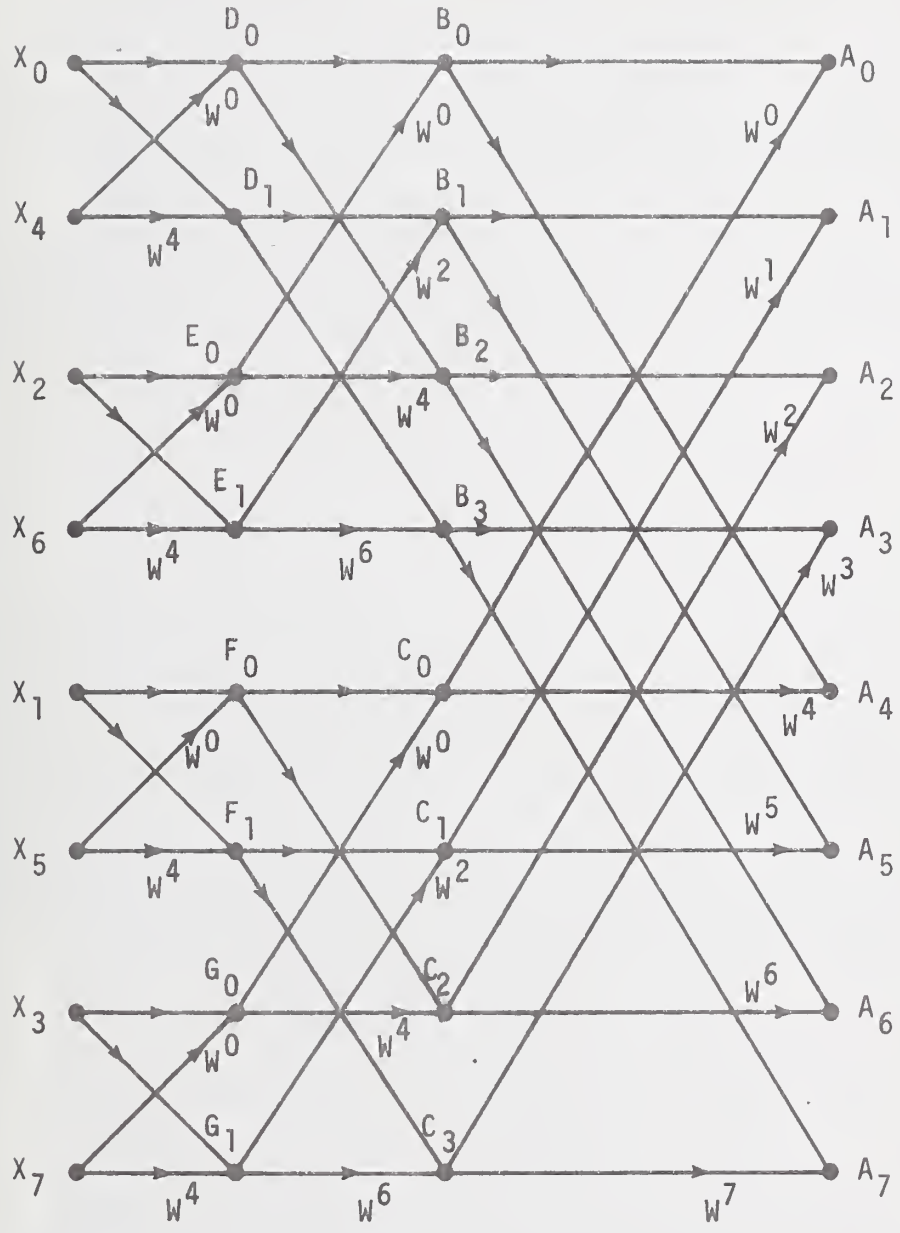


Fig. A-3 Signal flow graph of the FFT computation of an eight-point time domain waveform.

FORM NBS-114A (1-71)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBSIR 73-303	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE
Fast Fourier Transform Implementation for the Calculation of Network Frequency Domain Transfer Functions from Time Domain Waveforms

5. Publication Date
December 1972

6. Performing Organization Code

7. AUTHOR(S) William L. Gans and N. S. Nahman

8. Performing Organization
NBSIR 73-303

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS, Boulder Labs
DEPARTMENT OF COMMERCE
Boulder, Colo. 80302

10. Project/Task/Work Unit No.
2722390

11. Contract/Grant No.

12. Sponsoring Organization Name and Address
Department of Defense
Calibration Coordination Group
CCG 72-65

13. Type of Report & Period Covered
Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

16. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)

This report is concerned with the software applications of the fast Fourier transform algorithm to the relationship between time domain waveforms and frequency domain spectra. The first chapter is devoted to a description of the discrete Fourier transform and the fast Fourier transform. Chapter 2 contains the text and a brief description of all FORTRAN II programs utilized in connection with this work. All computation was performed on the in-house time share computing system in the NBS facilities, Boulder, Colorado. In Chapter 3, problems encountered using the fast Fourier transform algorithm are discussed, an example of a time domain to frequency domain calculation is presented, and future developmental considerations are mentioned. In addition Appendix A contains a detailed example aimed at disclosing the inner mechanisms of the fast Fourier transform algorithm.

17. KEY WORDS (Alphabetical order, separated by semicolons) Discrete Fourier transform; fast Fourier transform; frequency spectra, discrete; network transfer function; time domain waveform; transfer function.

18. AVAILABILITY STATEMENT

[X] UNLIMITED.

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NTIS.

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PAGES
UNCLASSIFIED	
20. SECURITY CLASS (THIS PAGE)	22. Price
UNCLASSIFIED	

USCOMM-DC 66244-P71

USCOMM - ERL