

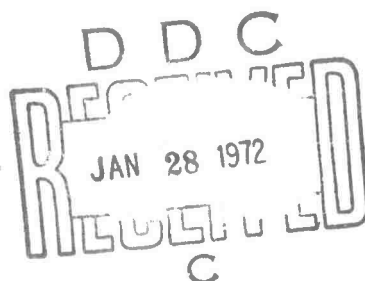
AD7 35920



TM-4857/100/00

A COMPARISON OF FFT ALGORITHMS

5 January 1972



Reproduced by
**NATIONAL TECHNICAL
INFORMATION SERVICE**
Springfield, Va. 22151

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

System Development Corporation
Santa Monica, California

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

A Comparison of FFT Algorithms

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical Report - July, 1971 to December, 1971

5. AUTHOR(S) (First name, middle initial, last name)

H. Barry Ritea

6. REPORT DATE

5 January 1972

7a. TOTAL NO. OF PAGES

32

7b. NO. OF REFS

3

8a. CONTRACT OR GRANT NO.

DAHC15-67-C-0149

8a. ORIGINATOR'S REPORT NUMBER(S)

8. PROJECT NO.

ARPA Order #1327, Amendment No. 4,
Program Code No. 2D30 and 2P10.

8b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

Several different FFT algorithms are presented. Each is compared on the basis of timing, accuracy, storage requirements, and other restrictions.

14

KEY WORDS

LINK A

LINK B

LINK C

ROLE

WT

ROLE

WT

ROLE

WT

Fast Fourier Transform

Digital Signal Processing

Speech Analysis

TECHNICAL MEMORANDUM

(TM Series)

The work reported herein was supported by the Advanced Research Projects Agency of the Department of Defense under Contract DAH015-67-C-0149, ARPA Order No. 1327, Amendment No. 4, Program Code No. 2D30 and 2P10.

A COMPARISON OF FFT ALGORITHMS

by

H. Barry Rites

5 January 1972

SYSTEM

DEVELOPMENT

CORPORATION

2500 COLORADO AVE.

SANTA MONICA

CALIFORNIA

90406

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U. S. Government.



Distribution of this document is unlimited.

5 January 1972

i
(Page ii blank)

System Development Corporation
TM-4857/

ABSTRACT

Several different FFT algorithms are presented.
Each is compared on the basis of timing, accuracy,
storage requirements, and other restrictions.

5 January 1972

iii
(Page iv blank)

System Development Corporation
TM-4857/100/00

TABLE OF CONTENTS

<u>Section</u>		<u>Page</u>
1.	INTRODUCTION	1
2.	THE FAST FOURIER TRANSFORM	1
2.1	Characteristics of the FFT Algorithms	2
2.2	FFT Summary and Comments	25
	REFERENCES	27

5 January 1972

v
(Page vi blank)

System Development Corporation
TM-4857/ 100/00

LIST OF TABLES

<u>Table</u>		<u>Page</u>
1.	Summary of FFT Characteristics	26

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
1.	FFT Algorithm I Program Listing	3
2.	FFT Algorithm II Program Listing	5
3.	FFT Algorithm III Program Listing	7
4.	FFT Algorithm IV Program Listing	21
5.	Subroutine RBITS Program Listing	23
6.	Subroutine RFLBTS Program Listing	24

1. INTRODUCTION

In this document, several different Fast Fourier Transform (FFT) algorithms are presented. Each FFT is tested with respect to timing, accuracy, storage requirements, and other restrictions. The test results for all the FFT codes are then summarized and compared. Complete FORTRAN codes and instructions for their use accompany the discussions.

2. THE FAST FOURIER TRANSFORM

Several different definitions of the discrete Fourier transform appear in the literature. The definition which we shall use is as follows: let $x_0, x_1, \dots, x_{n-1}$ be a sequence of data points. The discrete Fourier transform (DFT) of $x_0, x_1, \dots, x_{n-1}$ is the sequence $X_0, X_1, \dots, X_{n-1}$ given by

$$X_k = \sum_{j=0}^{n-1} x_j e^{-\frac{2\pi i j k}{n}} \quad (k=0, 1, \dots, n-1),$$

where $i=\sqrt{-1}$. The inverse discrete Fourier transform (IDFT) of $X_0, X_1, \dots, X_{n-1}$ is

$$x_j = \frac{1}{n} \sum_{k=0}^{n-1} X_k e^{\frac{2\pi i j k}{n}} \quad (j=0, 1, \dots, n-1)$$

The algorithm for efficiently computing the DFT and the IDFT is called the Fast Fourier Transform and was rediscovered in 1965 by Cooley and Tukey [1]. Several different FORTRAN codes for computing the FFT will be presented. In some cases, the codes allowed the calculation only of the DFT and not of the IDFT (or, in those instances where the definition of the DFT differed from ours, the calculation only of the IDFT). The codes have been modified so that both the DFT and the IDFT can be calculated by the same subroutine.

The following data are presented for the various FFT algorithms:

- (i) Reference to the literature.
- (ii) FORTRAN program listing.
- (iii) Timing: the average computation time for an FFT of n data points on the Raytheon 704 computer where it is assumed that n is a power of 2.
- (iv) Accuracy: if $x_0, x_1, \dots, x_{n-1}$ is the input sequence, we compute

$$e_j = \text{IDFT}\{\text{DFT}[x_j]\} - x_j$$

for $j=0, 1, \dots, n$ and then calculate the root-mean-square (RMS) error

$$\epsilon = \frac{1}{n} \sqrt{\sum_{j=0}^{n-1} |e_j|^2}$$

ϵ will be computed for the sequences

$$x_j = \exp(2\pi i j k / n)$$

for $k=2$ and $n=64, 128, 256, 512$ & 1024 .

- (v) Number of executable FORTRAN statements
- (vi) Internal array storage requirements.
- (vii) External array storage requirements.
- (viii) Type of arithmetic used: real, complex, or integer.
- (ix) Other restrictions
- (x) Program calling sequence

2.1 CHARACTERISTICS OF THE FFT ALGORITHMS

FFT Algorithm I

- (i) Reference: [2]
- (ii) FORTRAN program listing: See Figure 1.

```
      SUBROUTINE FFT(A,M,N,IS)
      DIMENSION A(N)
      COMPLEX A,U,W,T
      N=2**M
      NV2=N/2
      NM1=N-1
      J=1
      DO 30 I=1,NM1
      IF (I.GE.J) GO TO 10
      T=A(J)
      A(J)=A(I)
      A(I)=T
10     K=NV2
20     IF (K.GE.J) GO TO 30
      J=J-K
      K=K/2
      GO TO 20
30     J=J+K
      PI =3.14159265
      DO 80 L=1,M
      LE=2**L
      LE1=LE/2
      U=CMPLX(1.,0.)
      IF (IS) 40,50,50
40     W=CMPLX(COS(PI/LE1),-SIN(PI/LE1))
      GO TO 60
50     W=CMPLX(COS(PI/LE1),SIN(PI/LE1))
60     DO 80 J=1,LE1
      DO 70 I=J,N,LE
      IP=I+LE1
      T=A(IP)*U
      A(IP)=A(I)-T
70     A(I)=A(I)+T
80     U=U*W
      RETURN
      END
```

Figure 1. FFT Algorithm I Program Listing

(iii) Timing: $.0033n \log_2 n$

(iv) Accuracy:

<u>Number of Data Points</u>	<u>RMS error</u>
64	1.9675 E-06
128	3.2880 E-06
256	6.4563 E-06
512	1.1038 E-05
1024	1.8687 E-05

(v) Number of executable FORTRAN statements: 32

(vi) Internal array storage requirements: None.

(vii) External array storage requirements: A complex array of dimension n , where n is the number of data points.

(viii) Type of arithmetic used: Complex

(ix) Other restrictions: Number of data points must be a power of 2.

(x) Calling sequence: CALL FFT (A,M,N,IS), where

A = input array of samples

N = $2**M$ = number of samples

M = $\log_2(N)$

IS = -1, forward transform

= +1, inverse transform (unnormalized).

FFT Algorithm II

(i) Reference: [3]

(ii) FORTRAN program listing: See Figure 2.

(iii) Timing: $.0086n \log_2 n$.

(iv) Accuracy

<u>Number of Data Points</u>	<u>RMS Error</u>
64	9.2160 E-07
128	1.1040 E-06
256	1.2825 E-06
512	1.4254 E-06
1024	1.5463 E-06

```

SUBROUTINE FFT(X,NSTAGE,SIGN)
C INPUT:
C   X(2,1024) : DATA INPUT IN COLUMN 1
C   NSTAGE: POWER OF TWO WHICH N IS:
C   N = 2**NSTAGE
C   SIGN: =-1, FORWARD TRANSFORM
C         =+1, INVERSE TRANSFORM
C OUTPUT:
C   X(2,1024): FORWARD-TRANSFORMED DATA OUTPUT IN COLUMN 1
C               INVERSE-TRANSFORMED DATA OUTPUT IN COLUMN 2
C
COMPLEX X(2,1024),W
INTEGER R, SIGN
N=2**NSTAGE
N2=N/2
FLT N=N
PHI2N=6.2831853/FLT N
DO 3 J=1,NSTAGE
N2J=N/(2**J)
NR=N2J
NI=(2**J)/2
DO 2 I=1,NI
IN2J=(I-1)*N2J
FLIN2J=IN2J
XSIGN=SIGN
TEMP = FLIN2J*PHI2N*XSIGN
W=CMPLX(COS(TEMP),SIN(TEMP))
DO 2 R=1,NR
ISUB=R+IN2J
ISUB1=R+IN2J*2
ISUB2=ISUB1+N2J
ISUB3=ISUB+N2
X(2,ISUB)=X(1,ISUB1) + W*X(1,ISUB2)
X(2,ISUB3)=X(1,ISUB1) - W*X(1,ISUB2)
2 CONTINUE
DO 3 R=1,N
3 X(1,R)=X(2,R)
IF (SIGN.LT.0.) RETURN
DO 4 R=1,N
4 X(2,R)=X(1,R)/FLT N
RETURN
END

```

Figure 2. FFT Algorithm II Program Listing

- (v) Number of executable FORTRAN statements: 28
- (vi) Internal array storage requirements: None
- (vii) External array storage requirements: A $2 \times n$ complex array, where n is the number of data points.
- (viii) Type of arithmetic used: Complex.
- (ix) Calling sequence: CALL FFT (X, NSTAGE, SIGN), where
 - X(2,1024) = data input in column 1
 - NSTAGE = $\log_2(N)$, where N = the number of data points
 - SIGN = -1, forward transform: data output in column 1
 - = +1, inverse transform: normalized data output in column 2

FFT Algorithm III

- (i) Reference: [4]
- (ii) FORTRAN program listing: See Figure 3.
- (iii) Timing: $.0026n \log_2 n$.
- (iv) Accuracy:

<u>Number of Data Points</u>	<u>RMS error</u>
64	1.7072 E-06
128	1.4605 E-06
256	2.0886 E-06
512	2.0916 E-06
1024	2.3507 E-06

- (v) Number of executable FORTRAN statements: 478
- (vi) Internal array storage requirements: 312
- (vii) External array storage requirements: Two real arrays of dimension n , representing the real and imaginary parts of the input sequence of length n .
- (viii) Type of arithmetic used: Real

```

SUBROUTINE FFT(A,B,NTOT,N,NSPAN,ISN)
C  MULTIVARIATE COMPLEX FOURIER TRANSFORM, COMPUTED IN PLACE
C  USING MIXED-RADIX FAST FOURIER TRANSFORM ALGORITHM.
C  BY R. C. SINGLETON, STANFORD RESEARCH INSTITUTE, OCT. 1968
C  ARRAYS A AND B ORIGINALLY HOLD THE REAL AND IMAGINARY
C  COMPONENTS OF THE DATA, AND RETURN THE REAL AND
C  IMAGINARY COMPONENTS OF THE RESULTING FOURIER COEFFICIENTS.
C  MULTIVARIATE DATA IS INDEXED ACCORDING TO THE FORTRAN
C  ARRAY ELEMENT SUCCESSOR FUNCTION, WITHOUT LIMIT
C  ON THE NUMBER OF IMPLIED MULTIPLE SUBSCRIPTS.
C  THE SUBROUTINE IS CALLED ONCE FOR EACH VARIATE.
C  THE CALLS FOR A MULTIVARIATE TRANSFORM MAY BE IN ANY ORDER.
C  NTOT IS THE TOTAL NUMBER OF COMPLEX DATA VALUES.
C  N IS THE DIMENSION OF THE CURRENT VARIABLE.
C  NSPAN/N IS THE SPACING OF CONSECUTIVE DATA VALUES
C  WHILE INDEXING THE CURRENT VARIABLE.
C  THE SIGN OF ISN DETERMINES THE SIGN OF THE COMPLEX
C  EXPONENTIAL, AND THE MAGNITUDE OF ISN IS NORMALLY ONE.
C  A TRI-VARIATE TRANSFORM WITH A(N1,N2,N3), B(N1,N2,N3)
C  IS COMPUTED BY
C      CALL FFT(A,B,N1*N2*N3,N1,N1,1)
C      CALL FFT(A,B,N1*N2*N3,N2,N1*N2,1)
C      CALL FFT(A,B,N1*N2*N3,N3,N1*N2*N3,1)
C  FOR A SINGLE-VARIATE TRANSFORM,
C      NTOT = N = NSPAN = (NUMBER OF COMPLEX DATA VALUES), F.G.
C      CALL FFT(A,B,N,N,N,1)
C  THE DATA MAY ALTERNATIVELY BE STORED IN A SINGLE COMPLEX
C  ARRAY A, THEN THE MAGNITUDE OF ISN CHANGED TO TWO TO
C  GIVE THE CORRECT INDEXING INCREMENT AND A(2) USED TO
C  PASS THE INITIAL ADDRESS FOR THE SEQUENCE OF IMAGINARY
C  VALUES, E.G.
C      CALL FFT(A,A(2),NTOT,N,NSPAN,2)
C  ARRAYS AT(MAXF), CK(MAXF), BT(MAXF), SK(MAXF), AND NP(MAXP)
C  ARE USED FOR TEMPORARY STORAGE. IF THE AVAILABLE STORAGE
C  IS INSUFFICIENT, THE PROGRAM IS TERMINATED BY A STOP.
C  MAXF MUST BE .GE. THE MAXIMUM PRIME FACTOR OF N.
C  MAXP MUST BE .GT. THE NUMBER OF PRIME FACTORS OF N.
C  IN ADDITION, IF THE SQUARE-FREE PORTION K OF N HAS TWO OR
C  MORE PRIME FACTORS, THEN MAXP MUST BE .GE. K-1.
C      DIMENSION A(N), B(N)
C  ARRAY STORAGE IN NFAC FOR A MAXIMUM OF 11 FACTORS OF N.
C  IF N HAS MORE THAN ONE SQUARE-FREE FACTOR, THE PRODUCT OF THE
C  SQUARE-FREE FACTORS MUST BE .LE. 210
C      DIMENSION NFAC(11),NP(209)

```

Figure 3. FFT Algorithm III

```
C  ARRAY STORAGE FOR MAXIMUM PRIME FACTOR OF 23
    DIMENSION AT(23),CK(23),BT(23),SK(23)
    EQUIVALENCE (I,II)
C  THE FOLLOWING TWO CONSTANTS SHOULD AGREE WITH THE ARRAY DIMENSIONS.
    MAXF=23
    MAXP=209
    IF(N .LT. 2) RETURN
    INC=ISN
    RAD=8.0*ATAN(1.0)
    S72=RAD/5.0
    C72 = COS (S72)
    S72=SIN(S72)
    S120=SQRT(0.75)
    IF(ISN .GE. 0) GO TO 10
    S72=-S72
    S120=-S120
    RAD=-RAD
    INC=-INC
10  NT=INC*NTOT
    KS=INC*NSPAN
    KSPAN=KS
    NN=NT-INC
    JC=KS/N
    RADF=RAD*FLOAT(JC)*0.5
    I=0
    JF=0
C  DETERMINE THE FACTORS OF N
    M=0
    K=N
    GO TO 20
15  M=M+1
    NFAC(M)=4
    K=K/16
20  IF (K-(K/16)*16.EQ.0) GO TO 15
    J=3
    JJ=9
    GO TO 30
25  M=M+1
    NFAC(M)=J
    K=K/JJ
```

(Cont'd) Figure 3. FFT Algorithm III

```

30 IF(MOD(K,JJ) .EQ. 0) GO TO 25
   J=J+2
   JJ=J**2
   IF(JJ .LE. K) GO TO 30
   IF(K .GT. 4) GO TO 40
   KT=M
   NFAC(M+1)=K
   IF(K .NE. 1) M=M+1
   GO TO 80
40 IF(K-(K/4)*4 .NE. 0) GO TO 50
   M=M+1
   NFAC(M)=2
   K=K/4
50 KT=M
   J=2
60 IF(MOD(K,J) .NE. 0) GO TO 70
   M=M+1
   NFAC(M)=J
   K=K/J
70 J=((J+1)/2)*2+1
   IF(J .LE. K) GO TO 60
80 IF(KT .EQ. 0) GO TO 100
   J=KT
90 M=M+1
   NFAC(M)=NFAC(J)
   J=J-1
   IF(J .NE. 0) GO TO 90
C  COMPUTE FOURIER TRANSFORM
100 SD=RADF/FLOAT(KSPAN)
   CD=2.0*SIN(SD)**2
   SD=SIN(SD+SD)
   KK=1
   I=I+1
   IF(NFAC(I) .NE. 2) GO TO 400
C  TRANSFORM FOR FACTOR OF 2 (INCLUDING ROTATION FACTOR)
   KSPAN=KSPAN/2
   K1=KSPAN+2
210 K2=KK+KSPAN
   AK=A(K2)
   BK=B(K2)
   A(K2)=A(KK)-AK
   B(K2)=B(KK)-BK
   A(KK)=A(KK)+AK
   B(KK)=B(KK)+BK

```

(Cont'd) Figure 3. FFT Algorithm III

```

      KK=K2+KSPAN
      IF(KK .LE. NN) GO TO 210
      KK=KK-NN
      IF(KK .LE. JC) GO TO 210
      IF(KK .GT. KSPAN) GO TO 200
220  C1=1.0-CD
      S1=SD
230  K2=KK+KSPAN
      AK=A(KK)-A(K2)
      BK=B(KK)-B(K2)
      A(KK)=A(KK)+A(K2)
      B(KK)=B(KK)+B(K2)
      A(K2)=C1*AK-S1*BK
      B(K2)=S1*AK+C1*BK
      KK=K2+KSPAN
      IF(KK .LT. NT) GO TO 230
      K2=KK-NT
      C1=-C1
      KK=K1-K2
      IF(KK .GT. K2) GO TO 230
      AK=C1-(C1*C1+SD*S1)
      S1=(SD*C1-CD*S1)+S1
C   THE FOLLOWING THREE STATEMENTS COMPENSATE FOR TRUNCATION
C   ERROR. IF ROUNDED ARITHMETIC IS USED, SUBSTITUTE
C   C1=AK
      C1=0.5/(AK**2+S1**2)+0.5
      S1=C1*S1
      C1=C1*AK
      KK=KK+JC
      IF(KK .LT. K2) GO TO 230
      K1=K1+INC+INC
      KK=(K1-KSPAN)/2+JC
      IF (KK.LE.JC+JC) GO TO 220
      GO TO 100
C   TRANSFORM FOR FACTOR OF 3 (OPTIONAL CODE)
320  K1=KK+KSPAN
      K2=K1+KSPAN
      AK=A(KK)
      BK=B(KK)
      AJ=A(K1)+A(K2)
      BJ=B(K1)+B(K2)
      A(KK)=AK+AJ
      B(KK)=BK+BJ
      AK=-0.5*AJ+AK
      BK=-0.5*BJ+BK

```

(Cont'd) Figure 3. FFT Algorithm III

```
AJ=(A(K1)-A(K2))*S120
BJ=(B(K1)-B(K2))*S120
A(K1)=AK-BJ
B(K1)=BK+AJ
A(K2)=AK+BJ
B(K2)=BK-AJ
KK=K2+KSPAN
IF(KK .LT. NN) GO TO 320
KK=KK-NN
IF(KK .LE. KSPAN) GO TO 320
GO TO 700
C TRANSFORM FOR FACTOR OF 4
400 IF(NFAC(I) .NE. 4) GO TO 600
KSPNN=KSPAN
KSPAN=KSPAN/4
410 C1=1.0
S1=0.0
420 K1=KK+KSPAN
K2=K1+KSPAN
K3=K2+KSPAN
AKP=A(KK)+A(K2)
AKM=A(KK)-A(K2)
AJP=A(K1)+A(K3)
AJM=A(K1)-A(K3)
A(KK)=AKP+AJP
AJP=AKP-AJP
BKP=B(KK)+B(K2)
BKM=B(KK)-B(K2)
BJP=B(K1)+B(K3)
BJM=B(K1)-B(K3)
B(KK)=BKP+BJP
BJP=BKP-BJP
IF(ISN .LT. 0) GO TO 450
AKP=AKM-BJM
AKM=AKM+BJM
BKP=BKM+AJM
BKM=BKM-AJM
IF(S1 .EQ. 0.0) GO TO 460
430 A(K1)=AKP*C1-BKP*S1
B(K1)=AKP*S1+BKP*C1
A(K2)=AJP*C2-BJP*S2
B(K2)=AJP*S2+BJP*C2
A(K3)=AKM*C3-BKM*S3
B(K3)=AKM*S3+BKM*C3
KK=K3+KSPAN
IF(KK .LE. NT) GO TO 420
```

(Cont'd) Figure 3. FFT Algorithm III

```
440 C2=C1-(CD*C1+SD*S1)
    S1=(SD*C1-CD*S1)+S1
C   THE FOLLOWING THREE STATEMENTS COMPENSATE FOR TRUNCATION
C   ERROR. IF ROUNDED ARITHMETIC IS USED, SUBSTITUTE
C   C1=C2
    C1=0.5/(C2**2+S1**2)+0.5
    S1=C1*S1
    C1=C1*C2
    C2=C1**2-S1**2
    S2=2.0*C1*S1
    C3=C2*C1-S2*S1
    S3=C2*S1+S2*C1
    KK=KK-NT+JC
    IF(KK .LE. KSPAN) GO TO 420
    KK=KK-KSPAN+INC
    IF(KK .LE. JC) GO TO 410
    IF(KSPAN .EQ. JC) GO TO 800
    GO TO 100
450 AKP=AKM+BJM
    AKM=AKM-BJM
    BKP=BKM-AJM
    BKM=BKM+AJM
    IF(S1 .NE. 0.0) GO TO 430
460 A(K1)=AKP
    B(K1)=BKP
    A(K2)=AJP
    B(K2)=BJP
    A(K3)=AKM
    B(K3)=BKM
    KK=K3+KSPAN
    IF(KK .LE. NT) GO TO 420
    GO TO 440
C   TRANSFORM FOR FACTOR OF 5 (OPTIONAL CODE)
510 C2=C72**2-S72**2
    S2=2.0*C72*S72
520 K1=KK+KSPAN
    K2=K1+KSPAN
    K3=K2+KSPAN
    K4=K3+KSPAN
    AKP=A(K1)+A(K4)
    AKM=A(K1)-A(K4)
    BKP=B(K1)+B(K4)
    BKM = B(K1)-B(K4)
```

(Cont'd) Figure 3. FFT Algorithm III

5 January 1972

13

System Development Corporation
TM-4857/100/00

```
AJP=A(K2)+A(K3)
AJM=A(K2)-A(K3)
BJP=B(K2)+B(K3)
BJM=B(K2)-B(K3)
AA=A(KK)
BB=B(KK)
A(KK)=AA+AKP+AJP
B(KK)=BB+BKP+BJP
AK=AKP+C72+AJP+C2+AA
BK=BKP+C72+BJP+C2+BB
AJ=AKM+S72+AJM+S2
BJ=BKM+S72+BJM+S2
A(K1)=AK-BJ
A(K4)=AK+BJ
B(K1)=BK-AJ
B(K4)=BK+AJ
AK=AKP+C2+AJP+C72+AA
BK=BKP+C2+BJP+C72+BB
AJ=AKM+S2-AJM+S72
BJ=BKM+S2-BJM+S72
A(K2)=AK-BJ
A(K3)=AK+BJ
B(K2)=BK-AJ
B(K3)=BK+AJ
KK=K4+KSPAN
IF(KK .LT. NN) GO TO 520
KK=KK-NN
IF(KK .LE. KSPAN) GO TO 520
GO TO 700
C TRANSFORM FOR ODD FACTORS
600 K=NFAC(1)
KSPNN=KSPAN
KSPAN=KSPAN/K
IF(K .EQ. 3) GO TO 320
IF(K .EQ. 5) GO TO 510
IF(K .EQ. JF) GO TO 640
JF=K
S1=RAD/FLOAT(K)
C1=COS(S1)
S1=SIN(S1)
IF(JF .GT. MAXF) GO TO 998
CK(JF)=1.0
SK(JF)=0.0
J=1
```

(Cont'd) Figure 3. FFT Algorithm III

```
630 CK(J)=CK(K)*C1+SK(K)*S1
    SK(J)=CK(K)*S1-SK(K)*C1
    K=K-1
    CK(K)=CK(J)
    SK(K)=-SK(J)
    J=J+1
    IF(J .LT. K) GO TO 630
640 K1=KK
    K2=KK+KSPNN
    AA=A(KK)
    BB=B(KK)
    AK=AA
    BK=BB
    J=1
    K1=K1+KSPAN
650 K2=K2-KSPAN
    J=J+1
    AT(J)=A(K1)+A(K2)
    AK=AT(J)+AK
    BT(J)=B(K1)+B(K2)
    BK=BT(J)+BK
    J=J+1
    AT(J)=A(K1)-A(K2)
    BT(J)=B(K1)-B(K2)
    K1=K1+KSPAN
    IF(K1 .LT. K2) GO TO 650
    A(KK)=AK
    B(KK)=BK
    K1=KK
    K2=KK+KSPNN
    J=1
660 K1=K1+KSPAN
    K2=K2-KSPAN
    JJ=J
    AK=AA
    BK=BB
    AJ=0.0
    BJ=0.0
    K=1
```

(Cont'd) Figure 3. FFT Algorithm III

```
670 K=K+1
    AK=AT(K)*CK(JJ)+AK
    BK=BT(K)*CK(JJ)+BK
    K=K+1
    AJ=AT(K)*SK(JJ)+AJ
    BJ=BT(K)*SK(JJ)+BJ
    JJ=JJ+J
    IF(JJ .GT. JF) JJ=JJ-JF
    IF(K .LT. JF) GO TO 670
    K=JF-J
    A(K1)=AK-BJ
    B(K1)=BK+AJ
    A(K2)=AK+BJ
    B(K2)=BK-AJ
    J=J+1
    IF(J .LT. K) GO TO 660
    KK=KK+KSPNN
    IF(KK .LE. NN) GO TO 640
    KK=KK-NN
    IF(KK .LE. KSPAN) GO TO 640
C  MULTIPLY BY ROTATION FACTOR (EXCEPT FOR FACTORS OF 2 AND 4)
700 IF(I .EQ. M) GO TO 800
    KK=JC+1
710 C2=1.0-CD
    S1=SD
720 C1=C2
    S2=S1
    KK=KK+KSPAN
730 AK=A(KK)
    A(KK)=C2*AK-S2*B(KK)
    B(KK)=S2*AK+C2*B(KK)
    KK=KK+KSPNN
    IF(KK .LE. NT) GO TO 730
    AK=S1*S2
    S2=S1*C2+C1*S2
    C2=C1*C2-AK
    KK=KK-NT+KSPAN
    IF(KK .LE. KSPNN) GO TO 730
    C2=C1-(CD*C1+SD*S1)
    S1=S1+(SD*C1-CD*S1)
C  THE FOLLOWING THREE STATEMENTS COMPENSATE FOR TRUNCATION
C  ERROR. IF ROUNDED ARITHMETIC IS USED, THEY MAY
C  BE DELETED.
```

(Cont'd) Figure 3. FFT Algorithm III

```
C1=0.5/(C2**2+S1**2)+0.5
S1=C1*S1
C2=C1*C2
KK=KK-KSPNN+JC
IF(KK .LE. KSPAN) GO TO 720
KK=KK-KSPAN+JC+INC
IF(KK .LE. JC+JC) GO TO 710
GO TO 100
C PERMUTE THE RESULTS TO NORMAL ORDER---DONE IN TWO STAGES
C PERMUTATION FOR SQUARE FACTORS OF N
800 NP(1)=KS
   IF(KT .EQ. 0) GO TO 890
   K=KT+KT+1
   IF(M .LT. K) K=K-1
   J=1
   NP(K+1)=JC
810 NP(J+1)=NP(J)/NFAC(J)
   NP(K)=NP(K+1)*NFAC(J)
   J=J+1
   K=K-1
   IF(J .LT. K) GO TO 810
   K3=NP(K+1)
   KSPAN=NP(2)
   KK=JC+1
   K2=KSPAN+1
   J=1
   IF(N .NE. NTOT) GO TO 850
C PERMUTATION FOR SINGLE-VARIATE TRANSFORM (OPTIONAL CODE)
820 AK=A(KK)
   A(KK)=A(K2)
   A(K2)=AK
   BK=B(KK)
   B(KK)=B(K2)
   B(K2)=BK
   KK=KK+INC
   K2=KSPAN+K2
   IF(K2 .LT. KS) GO TO 820
830 K2=K2-NP(J)
   J=J+1
   K2=NP(J+1)+K2
   IF(K2 .GT. NP(J)) GO TO 830
   J=1
```

(Cont'd) Figure 3. FFT Algorithm III

```
840 IF(KK .LT. K2) GO TO 820
    KK=KK+INC
    K2=KSPAN+K2
    IF(K2 .LT. KS) GO TO 840
    IF(KK .LT. KS) GO TO 830
    JC=K3
    GO TO 890
C PERMUTATION FOR MULTIVARIATE TRANSFORM
850 K=KK+JC
860 AK=A(KK)
    A(KK)=A(K2)
    A(K2)=AK
    BK=B(KK)
    B(KK)=B(K2)
    B(K2)=BK
    KK=KK+INC
    K2=K2+INC
    IF(KK .LT. K) GO TO 860
    KK=KK+KS-JC
    K2=K2+KS-JC
    IF(KK .LT. NT) GO TO 850
    K2=K2-NT+KSPAN
    KK=KK-NT+JC
    IF(K2 .LT. KS) GO TO 850
870 K2=K2-NP(J)
    J=J+1
    K2=NP(J+1)+K2
    IF(K2 .GT. NP(J)) GO TO 870
    J=1
880 IF(KK .LT. K2) GO TO 850
    KK=KK+JC
    K2=KSPAN+K2
    IF(K2 .LT. KS) GO TO 880
    IF(KK .LT. KS) GO TO 870
    JC=K3
890 IF(2*KT+1 .GE. M) RETURN
    KSPNN=NP(KT+1)
C PERMUTATION FOR SQUARE-FREE FACTORS OF N
    J=M-KT
    NFAC(J+1)=1
```

(Cont'd) Figure 3. FFT Algorithm III

```
900 NFAC(J)=NFAC(J)*NFAC(J+1)
    J=J-1
    IF(J .NE. KT) GO TO 900
    KT=KT+1
    NN=NFAC(KT)-1
    IF(NN .GT. MAXP) GO TO 998
    JJ=0
    J=0
    GO TO 906
902 JJ=JJ-K2
    K2=KK
    K=K+1
    KK=NFAC(K)
904 JJ=KK+JJ
    IF(JJ .GE. K2) GO TO 902
    NP(J)=JJ
906 K2=NFAC(KT)
    K=KT+1
    KK=NFAC(K)
    J=J+1
    IF(J .LE. NN) GO TO 904
C  DETERMINE THE PERMUTATION CYCLES OF LENGTH GREATER THAN 1
    J=0
    GO TO 914
910 K=KK
    KK=NP(K)
    NP(K)=-KK
    IF(KK .NE. J) GO TO 910
    K3=KK
914 J=J+1
    KK=NP(J)
    IF(KK .LT. 0) GO TO 914
    IF(KK .NE. J) GO TO 910
    NP(J)=-J
    IF(J .NE. NN) GO TO 914
    MAXF=INC*MAXF
C  REORDER A AND B, FOLLOWING THE PERMUTATION CYCLES
    GO TO 950
924 J=J-1
    IF(NP(J) .LT. 0) GO TO 924
    JJ=JC
```

(Cont'd) Figure 3. FFT Algorithm III

```
926 KSPAN=JJ
    IF(JJ .GT. MAXF) KSPAN=MAXF
    JJ=JJ-KSPAN
    K=NP(J)
    KK=JC*K+II+JJ
    K1=KK+KSPAN
    K2=0
928 K2=K2+1
    AT(K2)=A(K1)
    BT(K2)=B(K1)
    K1=K1-INC
    IF(K1 .NE. KK) GO TO 928
932 K1=KK+KSPAN
    K2=K1-JC*(K+NP(K))
    K=-NP(K)
936 A(K1)=A(K2)
    B(K1)=B(K2)
    K1=K1-INC
    K2=K2-INC
    IF(K1 .NE. KK) GO TO 936
    KK=K2
    IF(K .NE. J) GO TO 932
    K1=KK+KSPAN
    K2=0
940 K2=K2+1
    A(K1)=AT(K2)
    B(K1)=BT(K2)
    K1=K1-INC
    IF(K1 .NE. KK) GO TO 940
    IF(JJ .NE. 0) GO TO 926
    IF(J .NE. 1) GO TO 924
950 J=K3+1
    NT=NT-KSPNN
    II=NT-INC+1
    IF(NT .GE. 0) GO TO 924
    RETURN
C  ERROR FINISH, INSUFFICIENT ARRAY STORAGE
998 ISN=0
C  PRINT 999
    STOP
999 FORMAT(44H0ARRAY BOUNDS EXCEEDED WITHIN SUBROUTINE FFT)
    END
```

(Cont d) Figure 3. FFT Algorithm III

Other restrictions: This algorithm does not require that the number n of sample points be a power of 2; indeed, the prime factorization of $n = p_1^{k_1} \dots p_j^{k_j}$ need only have the property that $p_i \leq 23$ for $i=1, \dots, j$.

(x) Calling sequence: CALL FFT (A, B, N, N, N, ISN), where

A = array containing real parts of input data,

B = array containing imaginary parts of input data,

N = number of data points

ISN = -1, forward transform

= +1, inverse transform (unnormalized).

For the use of this algorithm for other than radix 2, see the program listing in Figure 3.

FFT Algorithm IV

- (i) Reference: [5]
- (ii) FORTRAN program listing: see Figure 4.
- (iii) Timing: $.0042n \log_2 n$
- (iv) Accuracy:

<u>Number of Data Points</u>	<u>RMS error</u>
64	9.5800 E-07
128	1.2362 E-06
256	1.3819 E-06
512	1.4222 E-06
1024	1.4869 E-06

- (v) Number of executable FORTRAN statements: 18
- (vi) Internal array storage requirements: None
- (vii) External array storage requirements: Two real arrays of dimension n , representing the real and imaginary parts of the input sequence of length n .
- (viii) Type of arithmetic: Real

```
C      SUBROUTINE FFT(X,Y,M,N,IS)
      X,Y = REAL & IMAGINARY PARTS OF THE INPUT SEQUENCE
      DIMENSION X(N),Y(N)
      DO 10 L0=1,M
      LMX=2**(M-L0)
      LIX=2*LMX
      SCL=6.283185/LIX
      DO 10 LM=1,LMX
      ARG=(LM-1)*SCL
      C=COS(ARG)
      S=-FLOAT(IS)*SIN(ARG)
      DO 10 LI=LIX,N,LIX
      J1=LI-LIX+LM
      J2=J1+LMX
      T1=X(J1)-X(J2)
      T2=Y(J1)-Y(J2)
      X(J1)=X(J1)+X(J2)
      Y(J1)=Y(J1)+Y(J2)
      X(J2)=C*T1+S*T2
      Y(J2)=C*T2-S*T1
10     RETURN
      END
```

Figure 4. FFT Algorithm IV Program Listing

(ix) Other restrictions: Number of data points must be a power of 2. Also, program must be used in conjunction with subroutine RBITS to appropriately post-order the computed DFT sequence (see Figure 5 for a listing of RBITS).*

(x) Calling sequence: CALL FFT (X, Y, M, N, IS), where

X = array containing real parts of input data,

Y = array containing imaginary parts of input data,

N = 2**M = number of data points

IS = -1, forward transform

= +1, inverse transform (unnormalized)

The above call to the FFT must be followed by either

CALL RBITS (X, Y, M, N)

or the two calls

CALL RFLBTS (X, N)

CALL RFLBTS (Y, N).

*RBITS is designed to perform a "bit-reflection", and not a bit-reversal as indicated by Markel. A simpler and more flexible bit-reflection code is shown in Figure 6; this program will be recognized as the first section of FFT algorithm 1.

```
      SUBROUTINE RBITS(X,Y,M,N)
C      PERFORMS IN-PLACE BIT REVERSAL FOR N=2**M VALUES X(I),
C      WHERE M IS LESS THAN OR EQUAL TO 10
C      OUTPUT SEQUENCE IS Y(I)
      DIMENSION X(N),Y(N),L(10)
      EQUIVALENCE (L10,L(1)),(L9,L(2)),(L8,L(3)),(L7,L(4)),(L6,L(5))
      EQUIVALENCE (L5,L(6)),(L4,L(7)),(L3,L(8)),(L2,L(9)),(L1,L(10))
      DO 20 J=1,10
      L(J)=1
      IF (J-M) 10,10,20
10     L(J)=2**(M+1-J)
20     CONTINUE
      JN=1
      DO 50 J1=1,L1
      DO 50 J2=J1,L2,L1
      DO 50 J3=J2,L3,L2
      DO 50 J4=J3,L4,L3
      DO 50 J5=J4,L5,L4
      DO 50 J6=J5,L6,L5
      DO 50 J7=J6,L7,L6
      DO 50 J8=J7,L8,L7
      DO 50 J9=J8,L9,L8
      DO 50 JR=J9,L10,L9
      IF (JN-JR) 30,30,40
30     R=X(JN)
      X(JN)=X(JR)
      X(JR)=R
      F1=Y(JN)
      Y(JN)=Y(JR)
      Y(JR)=F1
40     JN=JN+1
50     CONTINUE
      RETURN
      END
```

Figure 5. Subroutine RBITS Program Listing

```
      SUBROUTINE RFLBTS(A,N)
C      PERFORMS IN-PLACE 'BIT-REFLECTION' FOR N=2**M VALUES A(I)
C      E.G., FOR N=8, THE FOLLOWING MAPPING WOULD TAKE PLACE:
C      0=000 IS MAPPED INTO 000
C      1=001 IS MAPPED INTO 100
C      2=010 IS MAPPED INTO 010
C      3=011 IS MAPPED INTO 110
C      4=100 IS MAPPED INTO 001
C      5=101 IS MAPPED INTO 101,ETC.
C      ORIGINAL ORDERING OF SEQUENCE IS DESTROYED
      DIMENSION A(N)
      NV2=N/2
      NM1=N-1
      J=1
      DO 30 I=1,NM1
      IF (I.GE.J) GO TO 10
      T=A(J)
      A(J)=A(I)
      A(I)=T
10     K=NV2
20     IF (K.GE.J) GO TO 30
      J=J-K
      K=K/2
      GO TO 20
30     J=J+K
      RETURN
      END
```

Figure 6. Subroutine RFLBTS Program Listing

2.2 FFT SUMMARY AND COMMENTS

A summary of the characteristics of the FFT algorithms is given in Table 1. We see that algorithm III is the fastest and most flexible with respect to the number of allowable data points but requires more program storage than the others. The most accurate algorithm is II, but it is about two or three times slower than the others. Algorithms I and IV are the best of the four with respect to timing, accuracy, and storage requirements, and since timing is usually the chief requirement in speech processing, algorithm I appears to be optimal.

Table 1. Summary of FFT Characteristics

Algorithm	Reference	Timing	Accuracy		#FORTRAN Statements	Storage Requirements		Arithmetic	Comments
			n	RMS		Internal Array	External Array		
I	[2]	$.0033 \ n \ \log_2 n$	64	1.9675E-06	32	None	n	Complex	Number of data points must be a power of 2
			128	3.2880E-06					
			256	6.4563E-06					
			512	1.1038E-05					
			1024	1.8687E-05					
II	[3]	$.0086 \ n \ \log_2 n$	64	9.2160E-07	28	None	2n	Complex	Number of data points must be a power of 2
			128	1.1040E-06					
			256	1.2825E-06					
			512	1.4254E-06					
			1024	1.5463E-06					
III	[4]	$.0026 \ n \ \log_2 n$	64	1.7072E-06	478	312	2n	Real	Mixed-Radix Transform
			128	1.4605E-06					
			256	2.0886E-06					
			512	2.0916E-06					
			1024	2.3507E-06					
IV	[5]	$.0042 \ n \ \log_2 n$	64	9.5800E-07	18	None	2n	Real	Number of data points must be a power of 2; Must be used in conjunction with a bit-reflection subroutine
			128	1.2362E-06					
			256	1.3819E-06					
			512	1.4222E-06					
			1024	1.4869E-06					

5 January 1972

27
(last page)

System Development Corporation
TM-4857/100/00

REFERENCES

- [1] COOLEY, J. W. and TUKEY, J. W., "An Algorithm for the Machine Calculation of Complex Fourier Series," Math. Comp., Vol. 19 (1965), pp. 297-301.
- [2] COOLEY, J. W. LEWIS, P. A. W. and WELCH, P. D., "The Fast Fourier Transform and its Applications," IEEE Transactions on Education, Vol. 12 (1969), pp. 27-34.
- [3] UHRICH, M. L., "Fast Fourier Transforms Without Sorting," IEEE Transactions on Audio and Electroacoustics, Vol. AU-17 (1969), pp. 170-172
- [4] SINGLETON, R. C., "An Algorithm for Computing the Mixed Radix Fast Fourier Transform," IEEE Transactions on Audio and Electroacoustics, Vol. AU-17 (1969), pp. 93-103.
- [5] MARKEL, J. D., "FFT Pruning," IEEE Transactions on Audio and Electroacoustics, Vol. AU-19 (1971), pp. 305-311.