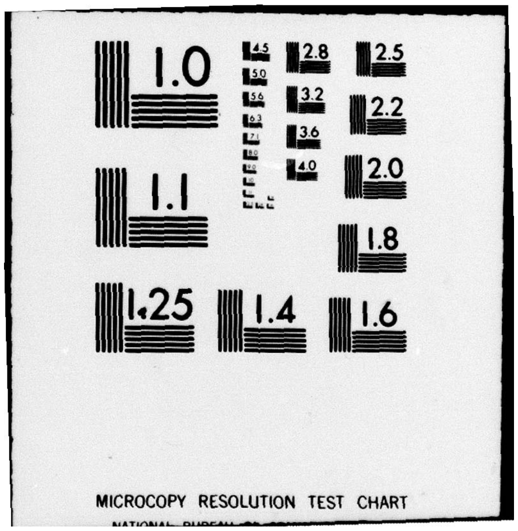

MICROCOPY RESOLUTION TEST CHART

11

MRC Technical Summary Report #1968

FFT AS NESTED MULTIPLICATION, WITH A TWIST

Carl de Boor

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53706

June 1979

Received May 11, 1979

See 1473 on back

Sponsored by

U. S. Army Research Office
P.O. Box 12211
Research Triangle Park
North Carolina 27709

UNIVERSITY OF WISCONSIN - MADISON
MATHEMATICS RESEARCH CENTER

FFT AS NESTED MULTIPLICATION, WITH A TWIST

Carl de Boor

Technical Summary Report #1968
June 1979

ABSTRACT

A simple, yet complete and detailed description of the Fast Fourier Transform for general N is given, with the aim of making the underlying idea quite apparent. To help with this didactic goal, a simple twist, i.e., a shifting of information from rows to columns during the calculations, is introduced which allows to give a simple meaning to intermediate results and assures that the final results need no further reordering.

AMS(MOS) Subject Classification: 65T05

Key Words: FFT = Fast Fourier Transform
nested multiplication

Work Unit No. 8 - Computer Science

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024.

[illegible] -A-

Significance and Explanation

In the last twenty years, various forms of a fast Fourier transform (FFT) have become popular. The various algorithms have in common that they produce the discrete Fourier transform on N points in about $N \ln(N)$ rather than N^2 operations. But, while these ideas, notably through Cooley & Tukey, have found wide application in computations, their didactic treatment has left something to be desired.

This report is an attempt to remedy this situation.

FFT AS NESTED MULTIPLICATION, WITH A TWIST

Carl de Boor

1. Introduction. The discrete Fourier transform (DFT) $\hat{\underline{\underline{z}}} = F_N \underline{\underline{z}}$ of an N-vector $\underline{\underline{z}}$ is given by the rule

$$(1) \qquad \hat{z}_\nu = \sum_{n=1}^{N} z_n \omega_N^{(\nu-1)(n-1)} \quad , \quad \nu = 1,\ldots,N \ ,$$

with

$$(2) \qquad \omega_N = \exp(-\sqrt{-1}\ 2\pi/N)$$

a principal N-th root of unity. Thus, $\hat{z}_\nu$ is given as the value of a polynomial of degree $< N$ at the point $\omega_N^{\nu-1}$ and can therefore be calculated, by nested multiplication, in N operations. Here, I follow Cooley & Tukey [1] in counting a complex multiplication followed by a complex addition as one operation.

In the last twenty years, various forms of a fast Fourier transform (FFT) have become popular. The various algorithms have in common that they produce the DFT on N points in about $N \ln(N)$ rather than N^2 operations. See Winograd [10] for the latest developments. But, while these ideas, notably through Cooley & Tukey [1], have found wide application in computations, their didactic treatment has left something to be desired.

In a recent article [7], H. R. Schwarz attempts, as he says, to remove the mystical aspect which the FFT has for many people. He does this by describing the FFT in terms of a factorization of the transformation matrix, an idea which he ascribes to Theilheimer [9] but which occurs already in Good [5] where a FFT different from that of Cooley & Tukey is given. A factorization of the transformation matrix is also the basic idea on which Glassman [4] builds his FFT, and Drubin [2] has refined this further; see Ferguson [3] for a lucid description and a simple Fortran program.

By contrast, I want to give here what I believe to be a simple description of the FFT for a general N in terms of nested multiplication. Certainly, Cooley & Tukey [1] thought of the FFT in these terms.

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024.

__2. The case of two factors.__ Suppose that $N = PQ$ for two integers P and Q greater than 1. Think of the N-vector $\underline{\underline{z}}$ as stored Fortran fashion in a one-dimensional array. Then we can interpret that array also Fortran fashion as a two-dimensional array Z, of dimension (P,Q). This means that

$$Z(p,q) = z_{p+P(q-1)} , \quad p = 1,\dots,P, \quad q = 1,\dots,Q . \tag{3}$$

Correspondingly, factor the sum (1) for $\hat{z}_\nu$ into a double sum,

$$\hat{z}_\nu = \sum_{p=1}^{P} \sum_{q=1}^{Q} Z(p,q)\, \omega_N^{(\nu-1)[p-1 + P(q-1)]}$$

$$= \sum_{p=1}^{P} \left[\sum_{q=1}^{Q} Z(p,q)\, \omega_Q^{(\nu-1)(q-1)} \right] \omega_N^{(\nu-1)(p-1)} .$$

Here, we have made use of the fact that

$$\omega_N^P = \omega_Q .$$

This makes apparent the crucial fact that <u>the inner sum</u> in the last right hand side is Q-periodic in ν, i.e., replacing ν by $\nu+Q$ does not change its value, due to the fact that $\omega_Q^Q = 1$. This means that we need only calculate this sum for $\nu = 1,\dots,Q$ (and for each p). Thus, for each $p = 1,\dots,P$, we calculate from the Q-vector $Z(p,\cdot)$ the Q-vector whose entries are the numbers

$$\sum_{q=1}^{Q} Z(p,q)\, \omega_Q^{(\nu-1)(q-1)} , \qquad \nu = 1,\dots,Q , \tag{4}$$

i.e., <u>we calculate the DFT</u> $F_Q Z(p,\cdot)$, $p = 1,\dots,P$, at a total cost of $P\cdot Q^2 = N\cdot Q$ operations.

Now, we could store the transform of $Z(p,\cdot)$ over $Z(p,\cdot)$. But in anticipation of further developments, we choose to store the transform $F_Q Z(p,\cdot)$ in $Z_1(\cdot,p)$, where Z_1 is a two dimensional array of size (Q,P), rather than (P,Q).

With this, the calculation of $\hat{z}_\nu$ is reduced to the evaluation of the sum

$$\hat{z}_\nu = \sum_{p=1}^{P} Z_1(\nu_Q,p)\, \omega_N^{(\nu-1)(p-1)} , \quad \nu = 1,\dots,N . \tag{5}$$

Here, we have used the notation ν_Q to indicate the integer between 1 and Q for which $\nu - \nu_Q$ is divisible by Q. At this point, it becomes convenient to think of the one-dimensional array which is to contain the N-vector $\hat{\underline{\underline{z}}}$ equivalently as a two-dimensional array Z_0 , of size (Q,P). This means that

(6) $$\hat{z}_{\nu + Q(\mu-1)} = Z_0(\nu,\mu) \ , \quad \text{for} \quad \nu = 1,\ldots,Q, \quad \mu = 1,\ldots,P \ .$$

With this, (5) can be written equivalently as

$$Z_0(\nu,\mu) = \sum_{p=1}^{P} Z_1(\nu,p)\, \omega_N^{[\nu-1 + Q(\mu-1)](p-1)}$$

$$\nu = 1,\ldots,Q, \quad \mu = 1,\ldots,P \ .$$

Here, the right hand side is a polynomial of degree $< P$ in the quantity $\omega_N^{\nu-1 + Q(\mu-1)}$. This quantity can be generated as one goes along, as in the following convenient arrangement of the calculations:

(7)
```
x := 1
for μ = 1,...,P , do:
    for ν = 1,...,Q , do:
        $Z_0(\nu,\mu) := \sum_{p=1}^{P} Z_1(\nu,p)x^{p-1}$
        $x := x \cdot \omega_N$ .
```

The sum in the innermost loop is, of course, to be evaluated by nested multiplication. The total cost of this step is then $Q \cdot P^2 = N \cdot P$ operations (if we neglect the N multiplications needed to generate the various x's). In this way, we have obtained in Z_0 the discrete Fourier transform $\hat{\underline{\underline{z}}}$ of $\underline{\underline{z}}$ at a cost of only $N(P+Q)$ rather than N^2 operations.

3. The general case. "It is easy to see how successive applications of the above procedure, starting with its application to (4), give an m-step algorithm requiring

$$T = N(P_1 + P_2 + \dots + P_m)$$

operations, where

$$N = P_1 \cdot P_2 \cdot \dots \cdot P_m \text{."} \tag{8}$$

So say Cooley & Tukey [1] (except for a change in symbols and equation numbers). In effect, they point out that the first step of the calculations above consists in forming the DFT of various Q-vectors. Hence if Q itself is the product of two integers greater than 1, this calculation can be carried out in fewer than Q^2 operations by applying the same procedure to it, etc. The actual implementation of this idea may not be immediately obvious, though. For this reason, I now discuss a slightly different (and novel) view, according to which the entire transform can be effected by m applications of a slightly enlarged version of (7).

The basic idea is to interpret the storage arrays for the various N-vectors involved in various ways as multidimensional arrays and to shift information appropriately from "rows to columns" as we did earlier when storing the DFT of the row $Z(p,\cdot)$ of Z in the column $Z_1(\cdot,p)$ of Z_1. For this, I need some notation to indicate that a given one-dimensional array is being considered equivalently as a two- or three-dimensional array.

If Z is a one-dimensional array of length N, then Z^A denotes the equivalent two-dimensional array of dimension $(A,N/A)$, and $Z^{A,B}$ denotes the equivalent three-dimensional array of dimension $(A,B,N/(AB))$. Thus

$$\begin{aligned} Z^{A,B}(a,b,c) &= Z^A(a,b+B(c-1)) = Z^{AB}(a+A(b-1),c) \\ &= Z(a+A(b-1+B(c-1))) . \end{aligned} \tag{9}$$

Let now Z be a one-dimensional array containing $\underline{\underline{z}}$, as before, and, for $k = 0,\dots,m$, let Z_k be a one-dimensional array satisfying

$$Z_k^A(\cdot,c) = F_A Z^{BP}(c,\cdot), \quad c = 1,\dots,BP, \tag{10}$$

with

(11) $$B := B_k := P_1 \cdot \ldots \cdot P_{k-1}, \quad P := P_k, \quad A := A_k := P_m \cdot \ldots \cdot P_{k+1}$$

Then $Z_m = Z$, and Z_0 contains $\hat{\underline{\underline{z}}} = F_N \underline{\underline{z}}$. Further, with A, P, B as given by (11), one obtains Z_{k-1} from Z_k by the following slightly extended version of (7):

(12)
```
x := 1
for p = 1,...,P , do:
    for a = 1,...,A , do:
        for b = 1,...,B, do:
            $Z^{A,P}_{k-1}(a,p,b) := \sum_{\pi=1}^{P} Z^{A,B}_k(a,b,\pi) \cdot x^{\pi-1}$
    x := x · $\omega_{AP}$.
```

Indeed, the algorithm produces

$$Z^{A,P}_{k-1}(a,p,b) = \sum_{\pi=1}^{P} Z^{A,B}_k(a,b,\pi)\ \omega_{AP}^{[a-1+A(p-1)](\pi-1)} .$$

On the other hand, (10) implies that

$$Z^{A,B}_k(\cdot,b,\pi) = F_A Z^{B,P}(b,\pi,\cdot) = \sum_{\alpha=1}^{A} Z^{B,P}(b,\pi,\alpha)\ \omega_A^{(\cdot-1)(\alpha-1)} .$$

Therefore,

$$Z^{A,B}_{k-1}(a,p,b) = \sum_{\pi=1}^{P} \sum_{\alpha=1}^{A} Z^{B,P}(b,\pi,\alpha)\ \omega_{AP}^{P(a-1)(\alpha-1) + [a-1+A(p-1)](\pi-1)} .$$

But now, since $\omega_{AP}^{AP} = 1$, we may add to the exponent on the right hand side any integer multiple of AP, and this allows the conclusion that

$$Z^{A,P}_{k-1}(a,p,b) = \sum_{\pi=1}^{P} \sum_{\alpha=1}^{A} Z^{B,P}(b,\pi,\alpha)\ \omega_{AP}^{[a-1+A(p-1)][\pi-1+P(\alpha-1)]}$$

and so proves that Z_{k-1}, as produced by (12), satisfies (10) (with k replaced by k-1).

This shows that the DFT $\hat{\underline{\underline{z}}}$ is obtainable, in Z_0, by m applications of algorithm (21), starting from Z_m containing $\underline{\underline{z}}$. Since the k-th such application costs $P_k A_k B_k P_k = N \cdot P_k$ operations, the total number of required operations is indeed given by (8).

In a Fortran implementation of the algorithm, one would, of course, need only two arrays to play the role, in alternation, of the m+1 arrays $Z_m, \ldots, Z_0$.

```
      SUBROUTINE FFT ( Z1, Z2, N, INZEE )
CONSTRUCTS THE DISCRETE FOURIER TRANSFORM OF  Z1 (OR Z2) IN THE COOLEY-
C  TUKEY WAY, BUT WITH A TWIST.
      INTEGER INZEE,N,    AFTER,BEFORE,NEXT,NEXTMX,NOW,PRIME(12)
      COMPLEX Z1(N),Z2(N)
C******  I N P U T  ******
C  Z1, Z2  COMPLEX N-VECTORS
C  N  LENGTH OF  Z1  AND  Z2
C  INZEE  INTEGER INDICATING WHETHER  Z1  OR  Z2  IS TO BE TRANSFORMED
C      = 1 , TRANSFORM  Z1
C      = 2 , TRANSFORM  Z2
C******  W O R K  A R E A S  ******
C  Z1, Z2  ARE BOTH USED AS WORKARRAYS
C******  O U T P U T  ******
C  Z1  OR  Z2  CONTAINS THE DESIRED TRANSFORM (IN THE CORRECT ORDER)
C  INZEE  INTEGER INDICATING WHETHER  Z1 OR Z2  CONTAINS THE TRANSFORM,
C      = 1 , TRANSFORM IS IN  Z1
C      = 2 , TRANSFORM IS IN  Z2
C******  M E T H O D  ******
C      THE INTEGER  N  IS DIVIDED INTO ITS PRIME FACTORS (UP TO A POINT).
C  FOR EACH SUCH FACTOR  P , THE P-TRANSFORM OF APPROPRIATE P-SUBVECTORS
C  OF  Z1 (OR Z2) IS CALCULATED IN  F F T S T P  AND STORED IN A SUIT-
C  ABLE WAY  IN  Z2 (OR Z1).  SEE TEXT FOR DETAILS.
C
      DATA NEXTMX,PRIME / 12, 2,3,5,7,11,13,17,19,23,29,31,37 /
      AFTER = 1
      BEFORE = N
      NEXT = 1
C
   10 IF ((BEFORE/PRIME(NEXT))*PRIME(NEXT) .LT. BEFORE) THEN
         NEXT = NEXT + 1
         IF (NEXT .LE. NEXTMX) THEN
                                                  GO TO 10
         ELSE
            NOW = BEFORE
            BEFORE = 1
         END IF
      ELSE
         NOW = PRIME(NEXT)
         BEFORE = BEFORE/PRIME(NEXT)
      END IF
C
      IF (INZEE .EQ. 1)  THEN
         CALL FFTSTP( Z1, AFTER, NOW, BEFORE, Z2 )
      ELSE
         CALL FFTSTP( Z2, AFTER, NOW, BEFORE, Z1 )
      END IF
      INZEE = 3 - INZEE
      IF (BEFORE .EQ. 1)                          RETURN
      AFTER = AFTER*NOW
                                                  GO TO 10
      END
      SUBROUTINE FFTSTP ( ZIN, AFTER, NOW, BEFORE, ZOUT )
C  CALLED IN  F F T .
C  CARRIES OUT ONE STEP OF THE DISCRETE FAST FOURIER TRANSFORM.
      INTEGER AFTER,BEFORE,NOW,    IA,IB,IN,J
      REAL ANGLE,RATIO,TWOPI
      COMPLEX ZIN(AFTER,BEFORE,NOW),ZOUT(AFTER,NOW,BEFORE),    ARG,OMEGA,
     *                                                         VALUE
      DATA TWOPI / 6.2831 85307 17958 64769 /
      ANGLE = TWOPI/FLOAT(NOW*AFTER)
      OMEGA = CMPLX(COS(ANGLE),-SIN(ANGLE))
      ARG = CMPLX(1.,0.)
      DO 100 J=1,NOW
         DO 90 IA=1,AFTER
            DO 80 IB=1,BEFORE
               VALUE = ZIN(IA,IB,NOW)
               DO 70 IN=NOW-1,1,-1
   70             VALUE = VALUE*ARG + ZIN(IA,IB,IN)
   80          ZOUT(IA,J,IB) = VALUE
   90       ARG = ARG*OMEGA
  100 CONTINUE
                                        RETURN
      END
```

There is no claim that the above program is competitive with the carefully constructed codes such as that of Singleton [8]. Its virtue lies chiefly in its simplicity and transparency. On the other hand, Eric Grosse [6] found that the above code, modified to give special treatment in FFTSTP for the case NOW = 2, and to avoid subroutine calls for the complex arithmetic operations, and compiled by an optimizing compiler, needed only 1.5 to 2 times as much computing time as did Singleton's program for a variety of choices of N .

Finally, the above discussion is based on the Fortran convention whereby multidimensional arrays are stored "column by column", i.e., with the first index running fastest. It is easy to base the discussion instead on the Algol convention whereby arrays are stored "row by row", i.e., with the last index running fastest.

Acknowledgement. I am grateful to Warren Ferguson for several discussions concerning fast Fourier transforms and for comments on an earlier draft. I am indebted to Eric Grosse for carrying out the comparisons mentioned above and for suggesting that the more leisurely discussion in an earlier draft be replaced by showing directly that the z_k as generated by (12) satisfy (10).

REFERENCES

1. J. W. Cooley & J. W. Tukey, An algorithm for the machine calculation of complex Fourier series, Math. Comp. 19 (1965) 297-301.

2. M. Drubin, Kronecker product factorization of the FFT matrix, IEEE Trans. on Computers, C-20 (1971) 590-593.

3. W. Ferguson, A simple derivation of Glassman's general N fast Fourier transform, Mathematics Research Center Technical Summary Report # (1979).

4. J. A. Glassman, A generalization of the fast Fourier transform, IEEE Trans. on Computers, C-19 (1970) 105-116.

5. I. J. Good, The interaction algorithm and practical Fourier series, J. Roy. Statist. Soc. Ser. B 20 (1958) 361-372; Addendum, 22 (1960) 372-375.

6. Eric Grosse, letter dated April 6, 1979.

7. H. R. Schwarz, Elementare Darstellung der schnellen Fouriertransformation, Computing 19 (1977) 107-116.

8. R. C. Singleton, Algorithm 339. An Algol procedure for the Fast Fourier Transform with arbitrary factors, Comm. Assoc. Comp. Mach. 11 (1968) 776ff.

9. F. Theilheimer, A matrix version of the Fast Fourier Transform, IEEE Trans. on Audio and Electroacoustics 17 (1969) 158-161.

10. S. Winograd, On computing the discrete Fourier transform, Math. Comp. 32 (1978) 175-199.

CdB:db

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 1968	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (*and Subtitle*) 6 FFT AS NESTED MULTIPLICATION, WITH A TWIST.		5. TYPE OF REPORT & PERIOD COVERED Summary Report - no specific reporting period
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(*s*) 10 Carl de Boor		8. CONTRACT OR GRANT NUMBER(*s*) 15 DAAG29-75-C-0024
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mathematics Research Center, University of Wisconsin 610 Walnut Street Madison, Wisconsin 53706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 8 - Computer Science
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office P.O. Box 12211 Research Triangle Park, North Carolina 27709		12. REPORT DATE 11 June 1979
		13. NUMBER OF PAGES 8
14. MONITORING AGENCY NAME & ADDRESS(*if different from Controlling Office*) 14 MRC-TSR-1968		15. SECURITY CLASS. (*of this report*) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (*of this Report*)

Approved for public release; distribution unlimited.

9 Technical summary rept., 12 12

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20, if different from Report*)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (*Continue on reverse side if necessary and identify by block number*)

FFT = Fast Fourier Transform
nested multiplication

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

A simple, yet complete and detailed description of the Fast Fourier Transform for general N is given, with the aim of making the underlying idea quite apparent. To help with this didactic goal, a simple twist, i.e., a shifting of information from rows to columns during the calculations, is introduced which allows to give a simple meaning to intermediate results and assures that the final results need no further reordering.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

221 200

[illegible]