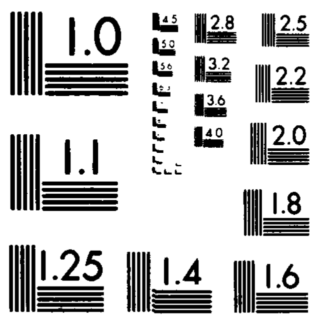
AD-A083 811 WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER F/G 12/1
A SIMPLE DERIVATION OF GLASSMAN'S GENERAL N FAST FOURIER TRANSF--ETC(U)
DEC 79 W E FERGUSON DAAG29-75-C-0024
UNCLASSIFIED MRC-TSR-2029 NL

1 OF 1
AD
A083811

END
DATE
FILMED
6-80
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(2)

AD A083811

(9) MRC Technical Summary Report #2029

(6) A SIMPLE DERIVATION OF GLASSMAN'S GENERAL N FAST FOURIER TRANSFORM.

(10) Warren E. Ferguson, Jr

LEVEL II

(14) MRC-TSR-2029

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

(11) Dec [redacted] 979 (12) 21

(Received July 27, 1979) (15) DAAG29-75-C-0024, NSF-MCS7[illegible]

Approved for public release
Distribution unlimited

Sponsored by

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

National Science Foundation
Washington, D. C. 20550

UNIVERSITY OF WISCONSIN-MADISON
MATHEMATICS RESEARCH CENTER

A SIMPLE DERIVATION OF GLASSMAN'S GENERAL N FAST FOURIER TRANSFORM

Warren E. Ferguson, Jr.

Technical Summary Report #2029
December 1979

ABSTRACT

A simple derivation of Glassman's general N fast Fourier transform, and corresponding FORTRAN program, is presented. This fast Fourier transform is based upon a representation of the discrete Fourier transform matrix as a product of sparse matrices.

AMS (MOS) Subject Classification: 65T05

Key Words: FFT, Fast Fourier transform factorization, Discrete Fourier transform

Work Unit Number 8 (Computer Science)

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024. This material is based upon work supported by the National Science Foundation under Grant No. MCS78-09525.

SIGNIFICANCE AND EXPLANATION

The discrete Fourier transform is the basis for several accurate techniques for the numerical solution of partial differential equations. The fast Fourier transform, an algorithm which allows one to compute rapidly the discrete Fourier transform, makes these techniques computationally efficient. This paper attempts to present a lucid description of one fast Fourier transform, the fast Fourier transform presented by Glassman.

In the past people have frequently been content to compute rapidly the discrete Fourier transform of vectors whose length is a power of two. Glassman's fast Fourier transform allows rapid computation of the discrete Fourier transform of vectors of arbitrary length.

The responsibility for the wording and views expressed in this descriptive summary lies with MRC, and not with the author of this report.

A SIMPLE DERIVATION OF GLASSMAN'S GENERAL N FAST FOURIER TRANSFORM

Warren E. Ferguson, Jr.

1. Introduction

Let the N-vector v be the discrete Fourier transform (DFT) of the N-vector u, i.e., the components v_k of v are computed from the components u_ℓ of u by the rule

$$v_k = \sum_{\ell=1}^{N} u_\ell \omega_N^{(k-1)(\ell-1)} \quad \text{for} \quad k = 1,2,\ldots,N$$

where

$$\omega_N \equiv \exp\{-2\pi\sqrt{-1}/N\}$$

is a principle N-th root of unity. It is easily demonstrated that the components of u can be recovered from the components of v by the rule

$$u_\ell = \frac{1}{N} \sum_{k=1}^{N} v_k \omega_N^{-(k-1)(\ell-1)} \quad \text{for} \quad \ell = 1,2,\ldots,N .$$

The N-point DFT matrix W_N is defined to be the matrix of order N whose entry in row i, column j is

$$\omega_N^{(i-1)(j-1)} .$$

Therefore the relations between u and v presented above can be written as

$$v = W_N u \quad \text{and} \quad u = \frac{1}{N} \bar{W}_N v$$

where $\bar{W}_N$ denotes the matrix obtained by replacing each entry of W_N by its complex conjugate.

A fast Fourier transform (FFT) is generally considered to be any algorithm which rapidly computes the DFT of a given vector. One of the most popular FFTs was presented by

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024. This material is based upon work supported by the National Science Foundation under Grant No. MCS78-09525.

Cooley and Tukey [3] in 1965. Their algorithm computes the DFT of an N-vector using

$$N \cdot (R_1 + R_2 + \cdots + R_K)$$

complex operations, where one operation denotes one multiplication followed by one addition, whenever N admits the representation

$$N = R_1 R_2 \cdots R_K$$

as a product of K positive integers $R_1, R_2, \ldots, R_K$. Since the publication of their article numerous authors have presented other FFTs, each requiring approximately the same number of complex operations. One notable exception is the FFT of Winograd [7].

In this paper I will present a description of Glassman's [5] FFT. This description of Glassman's FFT differs from one presented by Drubin [4] only in the definition of the tensor product. (However, neither Glassman nor Drubin presented a FORTRAN program which computes the DFT of a given N-vector.) I define the tensor product $A \quad B$ of two matrices A, B to be the matrix which, when partitioned into blocks the size of A, has $Ab_{i,j}$ as the entry in block row i and block column j. In the appendix of this paper I have presented proofs of three well known properties possessed by this tensor product.

Glassman's FFT computes the DFT of an N-vector using the same number of complex operations as the Cooley-Tukey FFT. The main advantage of Glassman's FFT is that it is easily coded, a fact which should be compared with Singleton's [6] FFT. The main disadvantage of Glassman's FFT is that it requires an N-vector of working storage to compute the DFT of an N-vector. I will show how one can, to some extent, eliminate this disadvantage.

I would also like to mention that de Boor [1] has recently presented an FFT that is also easily described and coded.

2. Factorization of the Discrete Fourier Transform Matrix

Consider the DFT matrix W_{PQ} where P,Q are two positive integers.

Partition the i-th row of W_{PQ} into Q groups of P successive entries. The entries in the q-th group are

$$\left[\omega_{PQ}^{(i-1)(0+(q-1)P)};\ \omega_{PQ}^{(i-1)(1+(q-1)P)};\ \ldots;\ \omega_{PQ}^{(i-1)(P-1+(q-1)P)}\right] .$$

Each member of this group contains the common factor

$$\omega_{PQ}^{(i-1)(q-1)P} = \omega_{Q}^{(i-1)(q-1)} ,$$

therefore the q-th group admits the representation

$$\omega_{Q}^{(i-1)(q-1)}\gamma_i^{(P,Q)}$$

where

$$\gamma_i^{(P,Q)} \equiv \left[\omega_{PQ}^{(i-1)(0)};\ \omega_{PQ}^{(i-1)(1)};\ \ldots;\ \omega_{PQ}^{(i-1)(P-1)}\right]$$

denotes the first P entries in the i-th row of W_{PQ}.

Next, partition the rows of W_{PQ} into P groups of Q successive rows. The rows in the p-th group are

$$\begin{bmatrix} \omega_Q^{(0+(p-1)Q)(0)}\gamma_{1+(p-1)Q}^{(P,Q)};\ \ldots;\ \omega_Q^{(0+(p-1)Q)(Q-1)}\gamma_{1+(p-1)Q}^{(P,Q)} \\ \vdots \\ \omega_Q^{(Q-1+(p-1)Q)(0)}\gamma_{Q+(p-1)Q}^{(P,Q)};\ \ldots;\ \omega_Q^{(Q-1+(p-1)Q)(Q-1)}\gamma_{Q+(p-1)Q}^{(P,Q)} \end{bmatrix} .$$

Observe that each member of this group contains the term

$$\omega_Q^{(p-1)Q} = 1 ,$$

therefore the p-th group admits the representation

$$\begin{bmatrix} \gamma_{1+(p-1)Q}^{(P,Q)} & & 0 \\ & \ddots & \\ 0 & & \gamma_{Q+(p-1)Q}^{(P,Q)} \end{bmatrix} \{I_P \otimes W_Q\} .$$

Here the matrix in square brackets is a block diagonal matrix, each block a $1 \times P$ matrix, where the i-th diagonal block is

$$\gamma^{(P,Q)}_{i+(p-1)Q}$$

and I_P is the identity matrix of order P.

These results allow us to prove the following

Lemma: The DFT matrix W_{PQ} admits the factorization

$$W_{P,Q} = F^{(P,Q)}\{I_P \otimes W_Q\}$$

where

$$\gamma_i^{(P,Q)} = \left[\omega_{PQ}^{(i-1)(0)};\ \omega_{PQ}^{(i-1)(1)};\ \ldots;\ \omega_{PQ}^{(i-1)(P-1)}\right]$$

denotes the first P entries in the i-th row of W_{PQ}, and

$$F^{(P,Q)} = \begin{bmatrix} \gamma^{(P,Q)}_{1+(0)Q} & \cdots & \gamma^{(P,Q)}_{Q+(0)Q} \\ \gamma^{(P,Q)}_{1+(1)Q} & \cdots & \gamma^{(P,Q)}_{Q+(1)Q} \\ & \vdots & \\ \gamma^{(P,Q)}_{1+(P-1)Q} & \cdots & \gamma^{(P,Q)}_{Q+(P-1)Q} \end{bmatrix}$$

is a $PQ \times Q$ block matrix with $1 \times P$ blocks.

Proof: From the definition of $F^{(P,Q)}$ we find that the p-th group of Q successive rows of

$$F^{(P,Q)}\{I_P \otimes W_Q\}$$

is

$$\begin{bmatrix} \gamma^{(P,Q)}_{1+(p-1)Q} & \cdots & \gamma^{(P,Q)}_{Q+(p-1)Q} \end{bmatrix} \{I_P \otimes W_Q\},$$

which our previous computations have shown to be the p-th group of Q successive rows of W_{PQ}. Since p was arbitrary we therefore conclude that

$$W_{PQ} = F^{(P,Q)}\{I_P \otimes W_Q\}. \qquad \blacksquare$$

The matrix $F^{(P,Q)}$ defined in the above lemma has several interesting limiting cases, in particular

$$F^{(P,1)} = W_P \quad \text{and} \quad F^{(1,Q)} = I_Q .$$

These observations aid us in the proof of the following

Theorem. Let N admit the representation

$$N = R_1 R_2 \cdots R_K$$

as a product of K positive integers $R_1, R_2, \ldots, R_K$. Then W_N admits the representation

$$W_N = F_1 F_2 \cdots F_K$$

as a product of K sparse matrices $F_1, F_2, \ldots, F_K$ where

$$F_L = I_{R_1 \cdots R_{L-1}} \otimes F^{(R_L, R_{L+1} \cdots R_K)} .$$

(The products $R_1 \cdots R_{L-1}$ for $L = 1$ and $R_{L+1} \cdots R_K$ for $L = K$ are defined to be 1.)

Proof: The previous lemma, with $P = R_1$ and $Q = R_2 \cdots R_K$, states that

$$W_N = F_1 \{ I_{R_1} \otimes W_{R_2 \cdots R_K} \} .$$

Therefore the identity

$$W_N = F_1 \cdots F_{L-1} \{ I_{R_1 \cdots R_{L-1}} \otimes W_{R_L \cdots R_K} \}$$

holds for $L = 2$. Let us suppose the identity holds for some $L < K$. The previous lemma, with $P = R_L$ and $Q = R_{L+1} \cdots R_K$, states that

$$W_{R_L \cdots R_K} = F^{(R_L, R_{L+1} \cdots R_K)} \{ I_{R_L} \otimes W_{R_{L+1} \cdots R_K} \} ,$$

and so

$$I_{R_1 \cdots R_{L-1}} \otimes W_{R_L \cdots R_K} = F_L \{ I_{R_1 \cdots R_L} \otimes W_{R_{L+1} \cdots R_K} \} .$$

Consequently, if the identity holds for some $L < K$ then it holds for $L + 1$ too. Therefore the identity must hold for $L = K$, i.e.

$$W_N = F_1 F_2 \cdots F_K$$

where we have noted that

$$I_{R_1 \cdots R_{K-1}} \otimes W_{R_K} = I_{R_1 \cdots R_{K-1}} \otimes F^{(R_K, 1)} = F_K . \qquad \blacksquare$$

3. A FORTRAN Implementation of Glassman's Fast Fourier Transform

The previous theorem, due to Glassman, allows us to easily code a FFT. For to compute

$$W_N u$$

with the result stored over u, we only need apply the factors $F_1, F_2, \ldots, F_K$ of W_N to u in the reverse order.

Suppose that we have just applied the factor

$$F_{L+1} = I_B \otimes F^{(C,A)}$$

to u, where (A = after, B = before, and C = current)

$$A = R_{L+2} \cdots R_K ,$$

$$B = R_1 \cdots R_L , \text{ and}$$

$$C = R_{L+1} .$$

Then we should next apply the factor

$$F_L = I_{B/R_L} \otimes F^{(R_L, AR_{L+1})}$$

to u. This computation can be described as

1. $A \leftarrow A \times C$
2. Let C be the divisor R_L of B
3. $B \leftarrow B/C$
4. $u \leftarrow I_B \otimes F^{(C,A)} u$.

Since the order of the divisors $R_1, R_2, \ldots, R_K$ of N is unimportant we find that the entire algorithm may be described as

1. $A \leftarrow 1$
2. $B \leftarrow N$
3. $C \leftarrow 1$
4. While $B > 1$ do
5. $\quad A \leftarrow A \times C$

6. Let $C > 1$ be a divisor of B

7. $B \leftarrow B/C$

8. $u \leftarrow I_B \otimes F^{(C,A)}u$

9. endwhile .

With the exception of steps 6 and 8, each step of this algorithm can be directly implemented in FORTRAN. Observe that step 6 admits the expansion

6.1 $C \leftarrow 2$

6.2 While B modulo $C \neq 0$ do

6.3 $C = C + 1$

6.4 endwhile

into steps that can be directly implemented in FORTRAN. We next consider the expansion of step 8.

Let the product RS of the integers R,S be a divisor of N. For any N-vector w we define $w^{(R)}$ to be the FORTRAN array of dimension $(R,N/R)$ which is equivalent to w, and $w^{(R,S)}$ to be the FORTRAN array of dimension $(R,S,N/RS)$ which is equivalent to w. This definition merely implies that

$$w^{(R)}_{i,j} = w_{i+(j-1)R}, \quad \text{and}$$

$$w^{(R,S)}_{i,j,k} = w_{i+(j-1)R+(k-1)RS} .$$

Let

$$v = I_B \otimes F^{(C,A)}u$$

denote the result of the computation described in step 8. As shown in the appendix we find that

$$v^{(B)} = u^{(B)}F^{(C,A)T} ,$$

or equivalently that

$$v^{(B)}_{i,j} = \sum_{k=1}^{AC} u^{(B)}_{i,k}F^{(C,A)}_{j,k}$$

for $i = 1,2,\ldots,B$ and $j = 1,2,\ldots,AC$. If we express j in the form

$$j = j_A + (j_C - 1)A ,$$

with $1 \leq j_A \leq A$ and $1 \leq j_C \leq C$, then the nonzero entries in the j-th row of [illegible] are the numbers

$$\omega_{AC}^{(j-1)(\ell-1)}$$

in columns $k = \ell + (j_A - 1)C$ for $\ell = 1,2,\ldots,C$. We therefore find that

$$v_{i,j_A+(j_C-1)A}^{(B)} = \sum_{\ell=1}^{C} u_{i,\ell+(j_A-1)C}^{(B)} \omega_{AC}^{(j_A-1+(j_C-1)A)(\ell-1)} ,$$

or equivalently that

$$v_{i,j_A,j_C}^{(B,A)} = \sum_{\ell=1}^{C} u_{i,\ell,j_A}^{(B,C)} \omega_{AC}^{(j_A-1+(j_C-1)A)(\ell-1)} ,$$

for $i = 1,2,\ldots,B$, $j_A = 1,2,\ldots,A$ and $j_C = 1,2,\ldots,C$. Consequently, step 8 admits the expansion

8.1 For $j_C = 1,2,\ldots,C$

8.2 For $j_A = 1,2,\ldots,A$

8.3 For $i = 1,2,\ldots,B$

8.4. $v_{i,j_A j_C}^{(B,A)} \leftarrow \sum_{\ell=1}^{C} u_{i,\ell,j_A}^{(B,C)} \omega_{AC}^{(j_A-1+(j_C-1)A)(\ell-1)}$

8.5 Next i

8.6 Next j_A

8.7 Next j_C

into steps that can be directly implemented in FORTRAN.

Figure 1 presents a FORTRAN version of Glassman's FFT. For comparison we present de Boor's [1] FFT in Figure 2. I have found that Glassman's FFT runs several percent faster than de Boor's FFT on the University of Wisconsin's UNIVAC 1110. This increase in speed is probably due to the fact that the loop structure used in Glassman's FFT can more efficiently be implemented in FORTRAN than the loop structure used in de Boor's FFT. This increase in speed would therefore vanish if one were to hand code both FFT's using [illegible] language.

```
      =====  FFT  =====

      SUBROUTINE FFT(N,U,WORK,INVRS)
      INTEGER N
      COMPLEX U(N),WORK(N)
      LOGICAL INVRS
C
C
C
C     *** INPUT ***
C
C     N       ... INTEGER
C     U       ... A COMPLEX N-VECTOR TO BE TRANSFORMED
C     INVRS   ... A LOGICAL VARIABLE
C
C     *** OUTPUT ***
C
C     U       ... THE DFT OF  U  IF  INVRS  IS .FALSE. , OR
C                 N TIMES THE INVERSE DFT OF  U  IF INVRS
C                 IS .TRUE.
C
C     *** WORKING STORAGE ***
C
C     WORK    ... A COMPLEX N-VECTOR
C
C
C
      INTEGER A,B,C
      LOGICAL INU
C
      A = 1
      B = N
      C = 1
      INU = .TRUE.
C
   10 IF (B.GT.1) GO TO 30
        IF (INU) RETURN
        DO 20 I=1,N
          U(I) = WORK(I)
   20   CONTINUE
        RETURN
C
   30 A = C*A
C
      DO 40 C=2,B
        IF (MOD(B,C).EQ.0) GO TO 50
   40 CONTINUE
C
   50 B = B/C
C
      IF (      INU) CALL GLASMN(A,B,C,U,WORK,INVRS)
      IF (.NOT.INU) CALL GLASMN(A,B,C,WORK,U,INVRS)
      INU = .NOT.INU
C
      GO TO 10
C
      END
```

Figure 1

```
=====  GLASMN  =====

      SUBROUTINE GLASMN(A,B,C,UIN,UOUT,INVRS)
      INTEGER A,B,C
      COMPLEX UIN(B,C,A),UOUT(B,A,C)
      LOGICAL INVRS
C
C
C
C     THIS SUBROUTINE IS CALLED FROM SUBROUTINE *FFT*
C
C
C
      COMPLEX DELTA,OMEGA,SUM
      DATA TWOPI/6.2831 85307 17958/
C
      ANGLE = TWOPI/FLOAT(A*C)
      DELTA = CMPLX(COS(ANGLE),-SIN(ANGLE))
      IF (INVRS) DELTA = CONJG(DELTA)
C
      OMEGA = CMPLX(1.,0.)
      DO 40 IC=1,C
        DO 30 IA=1,A
          DO 20 IB=1,B
            SUM = UIN(IB,C,IA)
            DO 10 JCR=2,C
              JC = C+1-JCR
              SUM = UIN(IB,JC,IA) + OMEGA*SUM
   10       CONTINUE
            UOUT(IB,IA,IC) = SUM
   20     CONTINUE
          OMEGA = DELTA*OMEGA
   30   CONTINUE
   40 CONTINUE
C
      RETURN
C
      END
```

Figure 1 - Cont'd.

```
===== FFT =====
      SUBROUTINE FFT(N,U,WORK,INVRS)
      INTEGER N
      COMPLEX U(N),WORK(N)
      LOGICAL INVRS
C
C
C
C     *** INPUT ***
C
C     N     ... INTEGER
C     U     ... A COMPLEX N-VECTOR TO BE TRANSFORMED
C     INVRS ... A LOGICAL VARIABLE
C
C     *** OUTPUT ***
C
C     U     ... THE DFT OF  U  IF  INVRS  IS .FALSE. , OR
C               N TIMES THE INVERSE DFT OF  U  IF INVRS
C               IS .TRUE.
C
C     *** WORKING STORAGE ***
C
C     WORK  ... A COMPLEX N-VECTOR
C
C
C
      INTEGER A,B,C
      LOGICAL INU
C
      A = 1
      B = N
      C = 1
      INU = .TRUE.
C
   10 IF (B.GT.1) GO TO 30
        IF (INU) RETURN
        DO 20 I=1,N
          U(I) = WORK(I)
   20   CONTINUE
        RETURN
C
   30 A = A*C
C
      DO 40 C=2,B
        IF (MOD(B,C).EQ.0) GO TO 50
   40 CONTINUE
C
   50 B = B/C
C
      IF (     INU) CALL DEBOOR(A,B,C,U,WORK,INVRS)
      IF (.NOT.INU) CALL DEBOOR(A,B,C,WORK,U,INVRS)
      INU = .NOT.INU
C
      GO TO 10
C
      END
```

Figure 2

```
      ===== DEROOD =====
      SUBROUTINE DEROOP(A,B,C,UIN,UOUT,INVRS)
      INTEGER A,B,C
      COMPLEX UIN(A,B,C),UOUT(A,C,B)
      LOGICAL INVRS
C
C
C
C     THIS SUBROUTINE IS CALLED FROM SUBROUTINE *FFT*
C
C
C
      COMPLEX OMEGA,DELTA,SUM
      DATA TWOPI/6.2831 85307 17958/
C
      ANGLE = TWOPI/FLOAT(A*C)
      DELTA = CMPLX(COS(ANGLE),-SIN(ANGLE))
      IF (INVRS) DELTA = CONJG(DELTA)
C
      OMEGA = CMPLX(1.,0.)
      DO 40 IC=1,C
        DO 30 IA=1,A
          DO 20 IB=1,B
            SUM = UIN(IA,IB,C)
            DO 10 JCR=2,C
              JC = C+1-JCR
              SUM = UIN(IA,IB,JC) + OMEGA*SUM
   10       CONTINUE
            UOUT(IA,IC,IB) = SUM
   20     CONTINUE
          OMEGA = DELTA*OMEGA
   30   CONTINUE
   40 CONTINUE
C
      RETURN
C
      END
```

Figure 2 - Cont'd.

4. Conclusion

Observe that Glassman's FFT requires an N-vector of working storage to compute the DFT of an N-vector, for during the computation

$$u \leftarrow I_B \otimes F^{(C,A)}u$$

we need an N-vector to store the result

$$v = I_B \otimes F^{(C,A)}u \ .$$

As explained in the following paragraph, this N-vector of working storage can be replaced by a C-vector of working storage at the expense of additional computational effort.

Let $P^{(C,A)}$ denote the permutation matrix of order AC which sends row $j_C + (j_A - 1)C$ of the vector w into row $j_A + (j_C - 1)A$ of the vector $P^{(C,A)}w$. Consequently

$$I_B \otimes P^{(C,A)}v = I_B \otimes P^{(C,A)}F^{(C,A)}u$$

where $P^{(C,A)}F^{(C,A)}$ is a block diagonal matrix with $C \times C$ blocks. Therefore the computation

$$u \leftarrow I_B \otimes F^{(C,A)}u$$

can be replaced by the equivalent computation

$$u \leftarrow I_B \otimes P^{(C,A)}F^{(C,A)}u \ ,$$
$$u \leftarrow I_B \otimes P^{(C,A)^T}u \ .$$

Careful consideration reveals that this latter sequence of calculations requires only a C-vector of working storage.

It is also possible to incorporate any FFT which computes the DFT of an N-vector for special values of N into Glassman's FFT. Recall that

$$W_N = F_1F_2 \cdots F_k$$

where

$$F_k = I_{R_1R_2 \cdots R_{k-1}} \otimes W_{R_k} \ .$$

Therefore any FFT which computes the DFT of an R_k-vector can be used when the factor F_k is to be applied to the vector being transformed.

Appendix: Some Properties of the Tensor Product

We have defined the tensor product $A \otimes B$ of two matrices A,B as the matrix which, when partitioned into blocks the size of A, has $Ab_{i,j}$ as the entry in block row i and block column j.

Consider now any N-vector w. If R is a divisor of N we define $w^{(R)}$ to be the FORTRAN array of dimension $(R,N/R)$ equivalent to w, i.e.

$$w^{(R)}_{i,j} = w_{i+(j-1)R} .$$

With these definitions in mind let us now prove the following

Property 1: Let A,B be rectangular matrices where A is a $R \times C$ matrix. Then

$$v = (A \otimes B)u$$

if and only if

$$v^{(R)} = Au^{(C)}B^T .$$

Proof: Let

$$v = (A \otimes B)u .$$

From the definition of the tensor product $A \otimes B$ we observe, for each i, that

$$v^{(R)}_{*,i} = \sum_j Ab_{i,j}u^{(C)}_{*,j} = A\Big\{\sum_j b_{i,j}u^{(C)}_{*,j}\Big\} .$$

The sum within the curly brackets is easily identified as the i-th column of

$$u^{(C)}B^T ,$$

consequently we infer that

$$v^{(R)} = Au^{(C)}B^T .$$

The proof of the converse is obtained by reversing the argument presented above. ∎

Carl de Boor [2] has noted that this property allows one to easily compute

$$v = (A \otimes B)u$$

given u. For if A is an $R \times C$ matrix then

$$v^{(R)} = Au^{(C)}B^T = \{B(Au^{(C)})^T\}^T ,$$

consequently programs which apply A and B to vectors can easily be used to apply $A \otimes B$ to vectors. This property also allows us to easily prove the following

Property 2: Let the products A_1A_2 and B_1B_2 be defined. Then

$$(A_1A_2) \otimes (B_1B_2) = (A_1 \otimes B_1)(A_2 \otimes B_2) .$$

Proof: Let A_k be an $R_k \times C_k$ matrix for $k = 1,2$. Observe that $C_1 = R_2$ because the product A_1A_2 is defined. Let u be an arbitrary vector and define

$$w = \{(A_1A_2) \otimes (B_1B_2)\}u .$$

Property 1 implies that

$$w^{(R_1)} = (A_1A_2)u^{(C_2)}(B_1B_2)^T = A_1(A_2u^{(C_2)}B_2^T)B_1^T .$$

If we define

$$v = (A_2 \otimes B_2)u$$

then property 1 implies that

$$v^{(R_2)} = A_2u^{(C_2)}B_2^T , \text{ and}$$

$$w^{(R_1)} = A_1v^{(R_2)}B_1^T$$

since $C_1 = R_2$. Using property 1 once more we find that

$$w = (A_1 \otimes B_1)v , \text{ and so}$$

$$w = (A_1 \otimes B_1)(A_2 \otimes B_2)u .$$

Consequently, for an arbitrary vector u we have

$$\{(A_1A_2) \otimes (B_1B_2)\}u = (A_1 \otimes B_1)(A_2 \otimes B_2)u ,$$

therefore

$$(A_1A_2) \otimes (B_1B_2) = (A_1 \otimes B_1)(A_2 \otimes B_2) .$$

The last tensor product property that we will need is described as follows.

Property 3: For arbitrary matrices A_1, A_2 and A_3

$$A_1 \otimes (A_2 \otimes A_3) = (A_1 \otimes A_2) \otimes A_3 .$$

Proof: Let A_k be an $R_k \times C_k$ matrix for $k = 1,2,$ and 3. Let $e_i^{(B)}$ be the B-vector obtained by replacing the i-th component of the zero B-vector by 1. Let $a_{i,j}^{(k)}$ denote the entry of A_k in row i and column j. Observe that

$$A_k = \sum_{i,j} a_{i,j}^{(k)} e_i^{(R_k)} e_j^{(C_k)T} \quad \text{for } k = 1,2,3 .$$

Consequently

$$A_1 \bullet (A_2 \bullet A_3) = \sum a_{i,j}^{(1)} a_{k,\ell}^{(2)} a_{m,n}^{(3)} [e_i^{(R_1)} e_j^{(C_1)T}] \bullet ([e_k^{(R_2)} e_\ell^{(C_2)T}] \bullet [e_m^{(R_3)} e_n^{(C_3)T}])$$

and

$$(A_1 \bullet A_2) \bullet A_3 = \sum a_{i,j}^{(1)} a_{k,\ell}^{(2)} a_{m,n}^{(3)} ([e_i^{(R_1)} e_j^{(C_1)T}] \bullet [e_k^{(R_2)} e_\ell^{(C_2)T}]) \bullet [e_m^{(R_3)} e_n^{(C_3)T}] .$$

From the easily verified identity

$$[e_i^{(R_1)} e_j^{(C_1)T}] \bullet ([e_k^{(R_2)} e_\ell^{(C_2)T}] \bullet [e_m^{(R_3)} e_n^{(C_3)T}]) =$$
$$([e_i^{(R_1)} e_j^{(C_1)T}] \bullet [e_k^{(R_2)} e_\ell^{(C_2)T}]) \bullet [e_m^{(R_3)} e_n^{(C_3)T}]$$

we deduce that

$$A_1 \bullet (A_2 \bullet A_3) = (A_1 \bullet A_2) \bullet A_3 . \quad \blacksquare$$

REFERENCES

1. C. de Boor, FFT as nested multiplication, with a twist, Mathematics Research Center, University of Wisconsin-Madison, Technical Summary Report #1968 (1979).
2. C. de Boor, Efficient computer manipulation of tensor products, Mathematics Research Center, University of Wisconsin-Madison, Technical Summary Report #1810 (1978).
3. J. W. Cooley and J. W. Tukey, An algorithm for the machine calculation of complex Fourier series, Math.Comp. 19 (1965), pp. 297-301.
4. M. Drubin, Kronecker product factorization of the FFT matrix, IEEE Trans. on Computers, C-20 (1971), pp. 590-593.
5. J. A. Glassman, A generalization of the fast Fourier transform, IEEE Trans. on Computers, C-19 (1970), pp. 105-116.
6. R. C. Singleton, Algorithm 339: An algol procedure for the fast Fourier transform with arbitrary factors, Comm. Assoc. Comp. Mach. 11 (1968), pp. 776-779.
7. S. Winograd, On computing the discrete Fourier transform, Math. Comp. 32 (1978), pp. 175-199.

WEF/scr

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 2029	2. GOVT ACCESSION NO.: AD-A083 811
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): A SIMPLE DERIVATION OF GLASSMAN'S GENERAL N FAST FOURIER TRANSFORM	5. TYPE OF REPORT & PERIOD COVERED: Summary Report - no specific reporting period
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Warren E. Ferguson, Jr.	8. CONTRACT OR GRANT NUMBER(s): DAAG29-75-C-0024 MCS78-09525
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: Work Unit Number 8 - Computer Science
11. CONTROLLING OFFICE NAME AND ADDRESS: See Item 18 below.	12. REPORT DATE: December 1979
	13. NUMBER OF PAGES: 17
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

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

National Science Foundation
Washington, D. C. 20550

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

FFT

Fast Fourier transform factorization

Discrete Fourier transform

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

A simple derivation of Glassman's general N fast Fourier transform, and corresponding FORTRAN program, is presented. This fast Fourier transform is based upon a representation of the discrete Fourier transform matrix as a product of sparse matrices.

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)