Tridiagonal Factorizations of Fourier Matrices and Applications to Parallel Computations of Discrete Fourier Transforms

Paul D. Gader
*Honeywell Systems and Research Center**
Minneapolis, Minnesota

Submitted by Donald J. Rose

ABSTRACT

Let $\omega_n = e^{-2\pi i/n}$, and F_n be the $n \times n$ matrix defined by

$$F_n = \frac{1}{\sqrt{n}}\left(\omega_n^{ij}\right),$$

where i and j run from 0 to $n-1$. Two different methods are developed for factoring F_n into products of tridiagonal and permutation matrices. One method is based on matrix identities associated with FFTs and the Rader prime algorithm. The other method is based on a numerical technique, never before applied to Fourier matrices, called minimal-variable oblique elimination. New results established in this paper include the establishment of necessary and sufficient conditions for which minimal-variable oblique elimination can be used to compute tridiagonal decompositions of arbitrary square matrices and explicit descriptions of minimal-variable solutions in the case that the descriptions are satisfied, proof that minimal-variable oblique elimination can be applied successfully to Fourier matrices of all orders, and explicit descriptions of various tridiagonal decompositions of PF_n for any $n \times n$ permutation matrix P. Complexity estimates are derived for both the parallel algorithms resulting from the decompositions and computation of the decompositions, and timings are estimated for processor arrays constructed using GAPP chips.

1. INTRODUCTION

The discrete Fourier transform (DFT) is used in many areas, one of these areas being digital image processing. In digital image processing, the DFT is used for image enhancement, image reconstruction, image encoding, feature

*Research performed while the author was at the University of Florida.

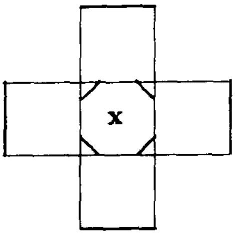

FIG. 1. The von Neumann neighborhood.

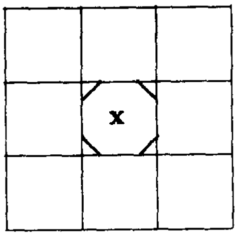

FIG. 2. The Moore neighborhood.

extraction, and computing convolutions and correlations [8, 12, 14, 36, 24, 25]. Although there are many algorithms for computing DFTs quickly, the large number of data used to represent a typical digital image and the real-time requirements of many digital image-processing tasks create a need for even more efficient methods of implementing these transforms. Parallel processing of digital images offers the realization of faster implementations of DFTs. In this article, we show how matrix techniques can be used to help achieve parallel implementations of DFTs.

A digital image can be thought of as a function defined on the discrete rectangle $X = \{(i, j): 0 \leqslant i \leqslant m-1,\ 0 \leqslant j \leqslant n-1\}$. An image-to-image transformation is a mapping with domain and range contained in the set of all such functions. For many useful image to image transformations, it happens that the new value at a point $x \in X$ depends only on the values in a neighborhood of the point. Typical neighborhoods are the von Neumann and Moore neighborhoods depicted in Figures 1 and 2.

Due to advances in VLSI (very-large-scale integrated circuit) technology, it has become feasible to build large rectangular arrays of simple processors. The processors in these arrays each have their own small memory and direct access to the memory of the processors in their neighborhoods, typically von Neumann or Moore neighborhoods. Thus, these processors can compute functions of the values in their neighborhoods. An image-to-image transformation of this type is called a local transformation. Some of the arrays that have been built according to this design are the Massively Parallel Processor (MPP) [5, 23, 31], the Distributed Array Processor (ICL DAP) [13, 20], the Geometric Arithmetic Parallel Processor (GAPP) [2, 29], and the CLIP4 [4, 5]. These arrays of processors are generally referred to as massively parallel processors, cellular array processors, or mesh-connected arrays. The term cellular array is due to the fact that the design of mesh-connected arrays is based on the concept of a cellular automation [19, 35].

In order that mesh-connected arrays be useful as general purpose image processors, efficient methods for implementing global transforms, particularly the DFT, need to be developed. Radix-two FFTs have been implemented on mesh-connected arrays by Jesshope [15] and Strong [31]. Due to image-sensor characteristics, however, images may be digitized in a variety of dimensions, such as 120×360 [26]. Furthermore, architectural considerations sometimes result in mesh-connected arrays being built with dimensions other than powers of two. Examples of such arrays are those built using GAPP chips, which have dimensions that are multiples of six [2, 29]. Moreover, even if it were possible on a mesh-connected array, experts in the field warn against enlarging the data set by appending enough zeros to force power-of-two dimensions [11]. Hence, there is a need for methods of implementing arbitrary discrete Fourier transforms locally. Since the DFT is a separable linear transform, it is sufficient to consider the problem of implementing a one-dimensional DFT locally on a linear array.

Let $n > 1$ and $\omega_n = e^{-2\pi i/n}$, where $i = \sqrt{-1}$, and let F_n denote that $n \times n$ matrix defined by

$$F_n = \frac{1}{\sqrt{n}}\left(\omega_n^{ij}\right), \qquad \text{where} \quad i, j = 0, 1, \ldots n-1.$$

Then, if $x \in C^n$, the transformation $x \to F_n x$ is the one-dimensional DFT of x. Tchuente has pointed out that M is any $n \times n$ matrix, then algorithms for implementing the transform $x \to Mx$ locally can be developed by factoring M into a product of tridiagonal matrices and permutation matrices [32]. The tridiagonal matrices represent local transformations, and the permutation matrices can be implemented using parallel sorting algorithms. We shall call such factorizations tridiagonal decompositions. Pease was also aware of this fact and derived tridiagonal decompositions of F_n for n a power of two [22].

In this paper, we develop two methods for computing tridiagonal decompositions of F_n for arbitrary n. In Section 2, we show how matrix algebra associated with FFTs can be used to accomplish this. We use identities established by Rose [27] as well as new identities established here which are related to the Rader prime algorithm [1, 18]. In Section 3, we take a completely novel approach to deriving methods for computing DFTs in parallel by using a numerical technique, called minimal-variable oblique elimination, to obtain tridiagonal decompositions of F_n. We derive necessary and sufficient conditions which an arbitrary square matrix must satisfy in order that minimal-variable oblique elimination can be applied to it. An explicit formula for computing the minimal-variable solution given. This new result is not restricted to the Fourier matrices but is applicable to all square matrices. The best result known prior to this was necessary and sufficient conditions for one stage of oblique elimination to be successful on a lower (or upper) triangular banded matrix [32]. We use the new conditions to establish that minimal-variable oblique elimination can be applied successfully to Fourier matrices. In Section 4, we derive complexity estimates for both the parallel algorithms resulting from the decompositions and computation of the decompositions and timing estimates for implementation on processor arrays constructed using GAPP chips. These estimates imply that algorithms for computing 100-point DFTs can be constructed using these decompositions which take, at most, 8 parallel multiplication steps, 26 parallel addition steps, and 316 parallel elementary permutation steps. By elementary permutation step we mean the action of switching data (complex numbers) in adjacent processors. Using timings provided for the MPP (very similar to GAPP), we estimate that a 100×100 DFT could be computed in as little as 0.00258 second and probably no more than 0.005 second counting overhead. These estimates suggest that the decompositions developed in this paper should be useful in computing DFTs in parallel.

Matrix representations of DFTs and FFTs have played a significant role in the development of FFT algorithms [9, 10, 16, 21, 27, 34]. Rose has given a description of most of this background and assimilated much of the work in a common framework [27]. He expressed the hope that the identities established there could be used to develop better FFT algorithms. In the same spirit, careful implementation of the decompositions presented in this article should result in efficient parallel algorithms for computing DFTs.

2. FFT-BASED DECOMPOSITIONS

In this section, we use matrix identities associated with FFTs to develop tridiagonal decompositions of Fourier matrices. We use identities developed by Rose [27] to express F_n as a product of permutation matrices and block-diagonal matrices in two different ways. The diagonal blocks of these matrices are of the form F_p where p is a prime divisor of n. We then formulate a matrix identity associated with the Rader prime algorithm and the circular convolution theorem [1, 3, 18]. This identity is then used to factor the Fourier matrices of prime order into products of permutation matrices, Fourier matrices of lower, composite order, and certain sparse triangular matrices. We show how the sparse matrices that appear can be factored. Taken together, these theorems provide an algorithm for deriving tridiagonal decompositions of F_n for arbitrary n.

We first establish notation and state identities which will be required in deriving the tridiagonal decompositions based on FFT identities. Unless otherwise stated, we consider matrices as ordered from 0 to $n-1$. We denote by Σ_n the group of permutations of n objects, and by Z_n the ring of integers modulo n. We think of Σ_n as acting on Z_n. Let $m \in \mathbf{Z}$ with $m > 1$, and let $\omega_m \equiv e^{-2\pi i/m}$, where $i = (-1)^{1/2}$. We sometimes use the symbolism $[j, k]$ to denote the set of integers $\{j, j+1, \dots, k\}$.

DEFINITION 1. Let A be an $m \times n$ matrix and B an $s \times t$ matrix, both with entries in a field F. The *Kronecker*, or *tensor*, *product* of A and B, denoted $A \otimes B$, is the $ms \times nt$ matrix given by

$$A \otimes B = \begin{bmatrix} a_{0,0}B & \cdots & a_{0,n-1}B \\ \vdots & & \vdots \\ a_{m-1,0}B & \cdots & a_{m-1,n-1}B \end{bmatrix}.$$

It is well known that $(A \otimes B)(C \otimes D) = AC \otimes BD$ provided that the matrices are all of the appropriate dimensions. Hence $\left(\prod_{i=0}^{k} A_i\right) \otimes I_j = \prod_{i=0}^{k}(A_i \otimes I_j)$.

DEFINITION 2. Let $C = (c_{ij})$ be an $n \times n$ matrix. We say that C is a *circulant matrix* of order n if and only if for every $k \in \mathbf{Z}$,

$$c_{ij} = c_{(i+k)(\text{mod } n),(j+k)(\text{mod } n)}.$$

Thus

$$C = \begin{bmatrix} c_0 & c_1 & \cdots & c_{n-1} \\ c_{n-1} & c_0 & \cdots & c_{n-2} \\ \vdots & \vdots & & \vdots \\ c_1 & c_2 & \cdots & c_0 \end{bmatrix}.$$

We write $C = \text{circ}(c_0, c_1, \dots, c_{n-1})$.

Let $P = \operatorname{circ}(0,1,0,\dots,0)$ be $n \times n$. If $C = \operatorname{circ}(c_0, c_1, \dots, c_{n-1})$ is any circulant matrix of order n, then let $f_C(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$. Then $C = f_C(P)$. Let $\Omega = \operatorname{diag}(\omega_n^i)$, $i = 0, 1, \dots, n-1$. Then, using the fact that

$$\sum_{i=0}^{n-1} \omega_n^{ik} = \begin{cases} n & \text{if} \quad k \equiv 0 \pmod{n}, \\ 0 & \text{else,} \end{cases}$$

it can be seen that $P = F_n \Omega F_n^*$ where $*$ denotes the conjugate transpose. It follows that if C is circulant, then $C = F_n \Lambda F_n^*$ where $\Lambda = \operatorname{diag}(f_C(\omega_n^i))$, $i = 0, 1, \dots, n-1$. This is the matrix formulation of the circular convolution theorem.

Definition 3. For every $\sigma \in \Sigma_n$ we define the $n \times n$ permutation matrix P_σ by

$$P_\sigma = (p_{ij}), \qquad \text{where} \quad p_{ij} = \begin{cases} 1 & \text{if } j = \sigma(i), \\ 0 & \text{otherwise.} \end{cases}$$

Assume that $n = mk$, where $m, k > 1$.

Definition 4. Define $\sigma_{mk} : \mathbf{Z}_n \to \mathbf{Z}_n$ by the following rule: If $i \in \mathbf{Z}_n$ and $i = a + bm$ with $0 \leqslant a < m$ and $0 \leqslant b < k$, then $\sigma_{mk}(i) = ak + b$. σ_{mk} is called a *shuffle* permutation.

Definition 5. Define $P(m,k) \in M_n$ by $P(m,k) \equiv P_{\sigma_{mk}}$. $P(m,k)$ is called a *shuffle permutation matrix*. Note that $P(m,k) = P(k,m)^t = P(k,m)^{-1}$.

For any $p, k > 1$ we denote $D_{p^k} \equiv D(p^{k-1}, p)$ and $P_{p^k} \equiv P(p^{k-1}, p)$. For convenience, we take $P(n,1) \equiv I_n$, the $n \times n$ identity matrix, and $I_1 \equiv 1$.

Definition 6. Let A be any $m \times m$ matrix, and define $T_k(A)$ by

$$T_k(A) \equiv \operatorname{blockdiag}[A^0, A^1, A^2, \dots, A^{k-1}].$$

Let

$$\Omega_m \equiv \operatorname{diag}\left(1, \omega_n, \omega_n^2, \dots, \omega_n^{m-1}\right).$$

Define the $n \times n$ diagonal matrix $D(m,k)$ by

$$D(m,k) \equiv T_k(\Omega_m).$$

The matrices $D(m,k)$ are called *twiddle factors* or *twiddle matrices*. Note that if $D(m,k)=\operatorname{diag}(d_i)$, then $d_i=\omega_n^{rq}$, where r and q are the unique integers satisfying $i=qm+r$ with $0\leqslant r<m$, that is, $r=i \pmod m$ and $q=(i-r)/m$.

We now state some basic identities concerning Fourier matrices, Kronecker products, and permutation matrices which were developed by Rose [27]. We will use these identities as building blocks for some of our derivations.

Let C_n denote the $n\times n$ circulant permutation matrix

$$C_n\equiv \operatorname{circ}(0,0,\dots,0,1).$$

For any integer s, let $Q(m,k,s)$ denote the $n\times n$ permutation matrix

$$Q(m,k,s)\equiv P(k,m)T_k(C_m^s)P(m,k).$$

FACTS.

(1) *If A is an $n\times n$ matrix and B is an $m\times m$ matrix, then $P(n,m)(A\otimes B)P(m,n)=B\otimes A$.*

(2) General radix identity (GRI):

$$F_n=(F_m\otimes I_k)D(k,m)(I_m\otimes F_k)P(k,m).$$

(3) Twiddle-free identity (TFI): *Assume that m and k are relatively prime, and let $m^*\equiv m^{-1} \pmod k$ and $k^*\equiv k^{-1} \pmod m$. Then*

$$F_n=Q(m,k,-k^*)(F_m\otimes F_k)T_m(C_k^{m^*})P(k,m).$$

We now use these identities to express the Fourier matrices as products of permutation matrices and block-diagonal matrices, the blocks of which are Fourier matrices of prime order. We first assume that n is not a power of a prime.

THEOREM 7. *If $n=\prod_{i=1}^{s}p_i^{k_i}$ with $s\geqslant 2$ and*

$$n_j=\begin{cases}\prod_{i=1}^{j}p_i^{k_i} & \text{if } 1\leqslant j\leqslant s,\\ 1 & \text{if } j=-1,0,\end{cases}$$

$c_j = n/n_j$ for $-1 \leqslant j \leqslant s$, $q_j = p_j^{k_j}$ for $1 \leqslant j \leqslant s$, and $q_0 = 1$, then there exists permutation matrices $Q_1, Q_2, \ldots, Q_{s-2}$, and H such that

$$F_n = \left(\prod_{j=0}^{s-2} \left(I_{n_j} \otimes Q_{2j+1}\right)\left(I_{n_j} \otimes F_{q_{j+1}} \otimes I_{c_{j+1}}\right) \right)$$
$$\times \left(I_{n_{s-1}} \otimes F_{q_s}\right) \left(\prod_{j=1}^{s-1} \left(I_{n_{s-j-1}} \otimes Q_{2(s-j)}\right) \right).$$

Proof. The proof is by induction on the number of primes, s.

Assume $s = 2$. Then, since $c_1 = q_2$, $n_1 = q_1$, and $n_0 = 1$, by Fact (3) there exist permutation matrices Q_1 and Q_2, namely $Q_1 = Q(q_1, q_2, -q_2^*)$ and $Q_2 = T_{q_1}(C_{q_2}^{q_1^*})P(q_2, q_1)$, such that $F_n = Q_1(F_{q_1} \otimes I_{q_2})(I_{q_1} \otimes F_{q_2})Q_2$, which is exactly the statement of the theorem.

Now assume that the theorem is true for all integers with prime factorization of length less than s. Then, in particular, the theorem is true for c_1. Thus, letting $\hat{n}_j = \prod_{i=2}^{j} p_i^{k_i}$ and $\hat{n}_1 = 1$, we have that there exist $Q_3, \ldots, Q_{2s-3}, Q_4, \ldots, Q_{2s-2}$ such that

$$F_{c_1} = \left(\prod_{j=1}^{s-2} \left(I_{\hat{n}_j} \otimes Q_{2j+1}\right)\left(I_{\hat{n}_j} \otimes F_{q_{j+1}} \otimes I_{c_{j+1}}\right) \right)$$
$$\times \left(I_{\hat{n}_{s-1}} \otimes F_{q_s}\right) \left(\prod_{j=2}^{s-2} \left(I_{\hat{n}_{s-j-1}} \otimes Q_{2(s-j)}\right) \right).$$

Moreover, there exist Q_1 and Q_2 such that

$$F_n = Q_1\left(F_{q_1} \otimes I_{c_1}\right)\left(I_{q_1} \otimes F_{c_1}\right)Q_2.$$

Putting these equations together and observing that $\hat{n}_j q_1 = n_j$ yields the desired result. ∎

The identity of Theorem 7 can be rearranged so as to obtain a block-diagonal representation of F_n.

COROLLARY 8. *Using the notation of Theorem 7, we have that*

$$F_n = \left(\prod_{j=0}^{s-2} R_j E_j \right) \left[I_{n_{s-2}} \otimes P(q_{s-1}, c_{s-1}) \right] \left(I_{n_{s-1}} \otimes F_{q_s}\right) \left(\prod_{j=1}^{s-1} \left(I_{n_{s-j-1}} \otimes Q_{2(s-j)}\right) \right),$$

where

$$R_j = \left(I_{n_{j-1}} \otimes P(q_j, c_j)\right)\left(I_{n_j} \otimes Q_{2j+1}\right)\left[I_{n_j} \otimes P(c_{j+1}, q_{j+1})\right]$$

and

$$E_j = \left(I_{n_j c_{j+1}} \otimes F_{q_{j+1}}\right).$$

One can also use the general radix identity in a similar way to obtain alternative decompositions of F_n. Verification of these decompositions can be accomplished in a fashion similar to that used in the proof of Theorem 7.

THEOREM 9. *Assume that $n = \prod_{j=1}^{s} k_i$ is any nontrivial factorization of n as a product of positive integers. Let $n_i \equiv \prod_{j=1}^{i} k_i$, $n_0 \equiv 1$, and $c_i \equiv n/n_i$. Then*

$$F_n = \left(\prod_{j=1}^{s-1} \left[I_{n_{j-1}} \otimes P(c_j, k_j)\right]\left(I_{n_{j-1}c_j} \otimes F_{k_j}\right) \right.$$
$$\left. \times \left[I_{n_{j-1}} \otimes P(k_j, c_j)\right]\left[I_{n_{j-1}} \otimes D(c_j, k_j)\right] \right)$$
$$\times \left(I_{n_{s-1}} \otimes F_{k_s}\right)\left(\prod_{j=2}^{s} \left[I_{n_{s-j}} \otimes P(c_{s-j+1}, k_{s-j+1})\right] \right).$$

COROLLARY 10. *Let n, n_i, and c_i be as in Theorem 7, and denote*

$$P_i = P\left(c_{i+1}, p_{i+1}^{k_{i+1}}\right) \quad \textit{and} \quad D_i = D\left(c_{i+1}, p_{i+1}^{k_{i+1}}\right).$$

Then

$$F_n = \left(\prod_{j=1}^{s-1} \left(I_{n_{j-1}} \otimes P_{j-1}\right)\left(I_{n_{j-1}c_j} \otimes F_{p_j^{k_j}}\right)\left(I_{n_{j-1}} \otimes P_{j-1}^{t}\right)\left(I_{n_{j-1}} \otimes D_{j-1}\right) \right)$$
$$\times \left(I_{n_{s-1}} \otimes F_{p_s^{k_s}}\right)\left(\prod_{j=2}^{s} \left(I_{n_{s-j}} \otimes P_{s-j}\right) \right).$$

COROLLARY 11. *If $k \geqslant 2$, then $F_{p^k} = G_p(I_{p^{k-1}} \otimes F_p)H_p$, where*

$$G_p = \prod_{m=0}^{k-2} \left[\left(I_{p^m} \otimes P_{p^{k-m}}\right)\left(I_{p^{k-1}} \otimes F_p\right)\left(I_{p^m} \otimes P_{p^{k-m}}^{t}\right)\left(I_{p^m} \otimes D_{p^{k-m}}\right)\right]$$

and

$$H_p = \prod_{m=2}^{k} \left[I_{p^{k-m}} \otimes P_{p^m} \right].$$

Note that Corollary 11 results in tridiagonal decompositions in the case $p = 2$. This is essentially the decomposition developed by Pease [22].

To obtain tridiagonal decompositions of the Fourier matrices of arbitrary size, methods based on identities other than the twiddle-free and the general radix identities must be used. Towards this end we formulate a matrix identity associated with the Rader prime algorithm.

Let p be an odd prime, and let $\alpha \in Z_p$ with $\alpha \neq 0, 1$. Let $R_{1,p}(\alpha)$ and $R_{2,p}(\alpha)$ be the permutation matrices corresponding to the permutations on Z_p defined by

$$\sigma_1 : \begin{cases} \alpha^i \to i \\ 0 \to 0 \end{cases} \qquad \sigma_2 : \begin{cases} i \to \alpha^{-i} \\ 0 \to 0 \end{cases}$$

Let $c_i \equiv \omega_p^{\alpha^{p-i-1}} - 1$, and let C_p be the $(p-1)\times(p-1)$ circulant matrix $C_p \equiv \operatorname{circ}(c_0, c_1, \dots, c_{p-2})$. Let A be any $n \times n$ matrix. We denote by $A^{(m)}$ the $(n+m)\times(n+m)$ matrix

$$A^{(m)} = \begin{bmatrix} I_m & 0 \\ 0 & A \end{bmatrix}.$$

We denote by U_p the $p \times p$ matrix with all ones in the first column, ones down the diagonal, and zeros elsewhere. For example, if $p = 5$, then

$$U_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Finally, let $\hat{F}_p$ denote the $p \times p$ matrix defined by $\hat{F}_p \equiv \operatorname{blockdiag}[1, (\omega_p^{ij} - 1)]$, where $i, j \in [1, p-1]$. The matrix formulation of the Rader prime algorithm is then given in parts (1) and (2) of the following theorem.

THEOREM 12. *Let $f_p(x)$ be the polynomial defined by $f_p(x) \equiv c_0 + c_1 x + c_2 x^2 + \cdots + c_{p-2} x^{p-2}$, and Λ_p the $p \times p$ matrix $\Lambda_p \equiv \operatorname{diag}(1, f_p(\omega_{p-1}^{i-1}))$,*

$i = 1, 2, \ldots, p-1$. *Then*

(1) $F_p = U_p \hat{F}_p U_p^t$,

(2) $\hat{F}_p = R_{1,p}(\alpha) C_p^{(1)} R_{2,p}(\alpha)$,

(3) $C_p^{(1)} = F_{p-1}^{(1)} \Lambda_p F_{p-1}^{*(1)}$.

Proof. The identity (1) was observed by Parlett [21]. The identity (3) follows from the matrix form of the convolution theorem.

To prove (2) we write

$$R_{1,p}(\alpha) = \left[\begin{array}{c|c} 1 & \\ \hline & R \end{array}\right], \qquad R_{2,p}(\alpha) = \left[\begin{array}{c|c} 1 & \\ \hline & S \end{array}\right],$$

where R and S are the permutation matrices corresponding to the restrictions of σ_1 and σ_2 to the set $\{1, 2, \ldots, p-1\}$, and the blank spaces represent zero entries. We also denote

$$\hat{F}_p = \left[\begin{array}{c|c} 1 & \\ \hline & F \end{array}\right].$$

where $F = (\omega_p^{ij} - 1)$. Since $R^{-1} = R^t$ and $S^{-1} = S^t$, it is sufficient to show that $R^t F S^t = C_p$. If we denote $A = (a_{ij}) = R^t F S^t$, where $i, j \in [1, p-1]$, then $a_{ij} = \omega^{\sigma_1^{-1}(i)\sigma_2(j)} - 1$. Note that if $j \in [1, p-1]$, then $\sigma_1^{-1}(1)\sigma_2(j) = \alpha\alpha^{-j} = \alpha^p \alpha^{-j}$ which implies that $a_{1j} = c_{j-1}$ so the first row of A is equal to the first row of C_p. The fact that A is circulant follows from the identity

$$\sigma_1^{-1}(i+k)\sigma_2(j+k) = \alpha^{i+k}\alpha^{-(j+k)} = \alpha^i \alpha^{-j} = \sigma_1^{-1}(i)\sigma_2(j).$$

Therefore, $A = C_p$ and the theorem is proved. ∎

Putting these identities together yields

$$F_p = U_p R_{1,p}(\alpha) F_{p-1}^{(1)} \Lambda_p F_{p-1}^{*(1)} R_{2,p}(\alpha) U_p^t$$

where * denotes the conjugate transpose.

Observe that if A and B are any $n \times n$ matrices, then $(I_m \otimes A^*) = (I_m \otimes A)^*$ and $A^{(m)} B^{(m)} = (AB)^{(m)}$. Therefore, if $F_{p-1} = \prod T_i$ is a tridiagonal decomposition of F_{p-1}, then $I_m \otimes F_{p-1}^{(1)} = I_m \otimes (\prod T_i)^{(1)} = I_m \otimes \prod (T_i^{(1)}) = \prod (I_m \otimes T_i^{(1)})$ is a tridiagonal decomposition of $I_m \otimes F_{p-1}^{(1)}$. Hence, if we can determine tridiagonal decompositions of U_p for p an odd prime, then we can proceed inductively to factor $F_{p-1}^{(1)}$ until $p = 2$ or $p = 3$. Thus, we need only determine methods for factoring the matrices U_p when p is an odd prime to complete our description of the FFT-based decompositions.

THEOREM 13. *Let p be an odd prime. Then*

$$U_p = \left(\prod_{k=p-1}^{2} \begin{bmatrix} I_k & 0 & 0 \\ (0,\ldots,0,1) & 1 & 0 \\ 0 & 0 & I_{p-k-1} \end{bmatrix} \right)$$

$$\times \begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 0 \\ 1 & 1 & 0 & & & & & \cdot & \cdot \\ 0 & -1 & 1 & \cdot & & & & \cdot & \cdot \\ \cdot & 0 & -1 & \cdot & \cdot & & & \cdot & \cdot \\ \cdot & \cdot & 0 & \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot & \cdot & 0 & \cdot \\ \cdot & \cdot & \cdot & & & \cdot & \cdot & 1 & 0 \\ 0 & 0 & 0 & \cdot & \cdot & \cdot & 0 & -1 & 1 \end{bmatrix}.$$

Proof. Let

$$U \equiv \left(\prod_{k=p-1}^{2} \begin{bmatrix} I_k & 0 & 0 \\ (0,\ldots,0,1) & 1 & 0 \\ 0 & 0 & I_{p-k-1} \end{bmatrix} \right)$$

$$\times \begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 0 \\ 1 & 1 & 0 & & & & & \cdot & \cdot \\ 0 & -1 & 1 & \cdot & & & & \cdot & \cdot \\ \cdot & 0 & -1 & \cdot & \cdot & & & \cdot & \cdot \\ \cdot & \cdot & 0 & \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot & \cdot & 0 & \cdot \\ \cdot & \cdot & \cdot & & & \cdot & \cdot & 1 & 0 \\ 0 & 0 & 0 & \cdot & \cdot & \cdot & 0 & -1 & 1 \end{bmatrix} \equiv (u_{ij}),$$

and denote

$$V_k \equiv \begin{bmatrix} I_k & 0 & 0 \\ (0,\ldots,0,1) & 1 & 0 \\ 0 & 0 & I_{p-k-1} \end{bmatrix}$$

and

$$V \equiv \begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 0 \\ 1 & 1 & 0 & & & & & \cdot & \cdot \\ 0 & -1 & 1 & \cdot & & & & \cdot & \cdot \\ \cdot & 0 & -1 & \cdot & \cdot & & & \cdot & \cdot \\ \cdot & \cdot & 0 & \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot & \cdot & 0 & \cdot \\ \cdot & \cdot & \cdot & & & \cdot & \cdot & 1 & 0 \\ 0 & 0 & 0 & \cdot & \cdot & \cdot & 0 & -1 & 1 \end{bmatrix}.$$

Let $\mathbf{x} = (x_0, x_1, \ldots, x_{p-1})^t \in \mathbf{C}^p$. We show that $U_p\mathbf{x} = U\mathbf{x}$. Note that

$$U_p\mathbf{x} = (x_0, x_1 + x_0, x_2 + x_0, \ldots, x_{p-1} + x_0)^t.$$

Now

$$V\mathbf{x} = (x_0, x_0 + x_1, x_2 - x_1, x_3 - x_2, \ldots, x_{p-1} - x_{p-2})^t.$$

Furthermore, if $\mathbf{y} = (y_0, y_1, \ldots, y_{p-1})^t \in \mathbf{C}^p$, then

$$V_k\mathbf{y} = (y_0, y_1, \ldots, y_{k-1}, y_k + y_{k-1}, y_{k+1}, \ldots, y_{p-1})^t.$$

Thus,

$$V_2(V\mathbf{x}) = (x_0, x_0 + x_1, x_0 + x_2, x_3 - x_2, \ldots, x_{p-1} - x_{p-2})^t,$$

and one can show by induction that

$$\begin{aligned} &V_j(V_{j-1}V_{j-2}\cdots V_2V\mathbf{x}) \\ &\quad = (x_0, x_0 + x_1, \ldots, x_0 + x_j, x_{j+1} - x_j, \ldots, x_{p-1} - x_{p-2})^t \end{aligned}$$

for $j \in [2, p-1]$. Hence, we may conclude that $U\mathbf{x} = U_p\mathbf{x}$ for every $\mathbf{x} \in \mathbf{C}^p$, which implies that $U = U_p$. ∎

We have shown how matrix identities associated with the FFT can be used to develop tridiagonal decompositions of Fourier matrices. We now derive a completely different method for computing tridiagonal decompositions of Fourier matrices.

3. TRIDIAGONAL DECOMPOSITIONS OF F_n VIA OBLIQUE ELIMINATION

In this section, we show how a technique called oblique elimination can be used to develop alternative tridiagonal decompositions of the Fourier matrices. We first describe the theoretical basis of oblique elimination. We then use this basis to derive an algorithm, which we call minimal variable oblique elimination, for implementing oblique elimination in certain cases, and we develop necessary and sufficient conditions for this algorithm to be successful in computing tridiagonal decompositions of a given matrix. We show that the minimum-variable oblique elimination algorithm can be used to compute tridiagonal decompositions of the Fourier matrices.

Throughout this section, let n be an arbitrary positive integer with $n > 3$, and assume that all matrices and vectors are taken over the complex numbers. In this section we consider matrices and vectors to be ordered from 1 to n, rather than 0 to $n-1$ as in the last section. When referring to a matrix it will always be assumed that the entries of the matrix are denoted by the same letter as the matrix unless explicitly stated otherwise.

Definition 14. Let A be any $n \times n$ matrix. We say that A has an *LU decomposition* if there exists an $n \times n$ unit lower triangular matrix L and an upper triangular matrix U such that $A = LU$.

Computing the LU decomposition of a matrix will be the first step in the oblique elimination algorithm. Not every matrix has an LU decomposition. It can be shown that if A is any $n \times n$ matrix, then there exists an $n \times n$ permutation matrix P such that PA has an LU decomposition [7, 32]. The most common method used for computing LU decompositions is Gaussian elimination with partial pivoting. A discussion of these topics can be found in almost any numerical-analysis book [7, 30]. We remark that, even if a matrix A has an LU decomposition, it may be desirable, for reasons of numerical stability, to compute the LU decomposition of PA for some permutation matrix P rather than that of A.

Definition 15. Let A be an $n \times n$ lower triangular matrix. Let $i \in [1, n]$. The ith *oblique* of A is the set $\{a_{n-i+1,1}, a_{n-i+2,2}, \dots, a_{n,i}\}$.

Note that we have defined the obliques relative to the lower triangular matrices. We could just as well have done so for upper triangular matrices. Throughout this section we shall work mainly with lower triangular matrices. The techniques can all be applied to the transposes of the upper triangular matrices that appear in the decompositions.

Definition 16. Let A be an $n\times n$ matrix. We say that A has *lower bandwidth* r if $a_{ij}=0$ whenever $i>j+r$.

An $n\times n$ lower triangular matrix A with lower bandwidth $n-j$ for some $j\in[1,n-1]$ has the property that the kth oblique is $\{0\}$ for every $k\in[1,j]$.

The oblique elimination method as applied to an $n\times n$ matrix M can be summarized in the following two steps:

(1) Determine a permutation matrix P such that $PM=LU$ is an LU decomposition of PM.

(2) Construct matrices $L_1^{-1}, L_2^{-1},\dots,L_{n-2}^{-1}$ such that for every $j\in[1,n-2]$ the matrix L_j is unit lower bidiagonal and the matrix $L_j^{-1}L_{j-1}^{-1}\cdots L_1^{-1}L$ has lower bandwidth $n-j$. Similarly, construct matrices $U_1^{-1},U_2^{-1},\dots,U_{n-2}^{-1}$ such that for every $j\in[1,n-2]$ the matrix U_j^t is unit lower bidiagonal and the matrix $(U_j^t)^{-1}(U_{j-1}^t)^{-1}\cdots(U_1^t)^{-1}U^t$ has lower bandwidth $n-j$.

Then

$$A=L_{n-2}^{-1}L_{n-3}^{-1}\cdots L_1^{-1}L,$$

$$B=UU_1^{-1}U_2^{-1}\cdots U_{n-2}^{-1}$$

are lower and upper bidiagonal matrices respectively. Thus,

$$M=P^{-1}L_1L_2\cdots L_{n-2}ABU_{n-2}U_{n-3}\cdots U_1$$

is a tridiagonal decomposition of M. This methodology does not always work, because one cannot always construct the matrices U_j and L_j. A method for doing so, which was suggested by Tchuente, is the following:

Let $x_1,x_2,\dots,x_{n-1}$ denote indeterminates, and let X denote the $n\times n$ matrix

$$X=\begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & \cdot & 0\\ -x_1 & 1 & 0 & & & & \cdot\\ 0 & -x_2 & 1 & \ddots & & & \cdot\\ \cdot & 0 & x_3 & \ddots & 0 & & \cdot\\ \cdot & & 0 & \ddots & 1 & 0 & \cdot\\ \cdot & & \cdot & \ddots & -x_{n-2} & 1 & 0\\ 0 & \cdot & \cdot & \cdot & 0 & -x_{n-1} & 1 \end{bmatrix}.$$

Then, since $I_n - X$ is nilpotent,

$$X^{-1} = [I - (I - X)]^{-1}$$

$$= \begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ x_1 & 1 & 0 & & & & & \cdot \\ x_1x_2 & x_2 & 1 & 0 & & & & \cdot \\ x_1x_2x_3 & x_2x_3 & x_3 & 1 & \cdot & & & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & & \cdot & 0 & \cdot \\ \cdot & \cdot & \cdot & & & \cdot & 1 & 0 \\ x_1x_2\cdots x_{n-1} & x_2\cdots x_{n-1} & x_3\cdots x_{n-1} & \cdot & \cdot & \cdot & x_{n-1} & 1 \end{bmatrix}.$$

Thus, one can attempt to construct the matrices $L_1^{-1}, L_2^{-1},\dots, L_{n-2}^{-1}$ and $(U_j^t)^{-1},(U_{j-1}^t)^{-1},\dots,(U_1^t)^{-1}$ in the form of X^{-1}. Let A be an $n \times n$ lower triangular matrix with lower bandwidth $n-i+1$. Then there exists an $n \times n$ matrix X^{-1} as above such that the matrix $B = X^{-1}A$ has lower bandwidth $n-i$ if and only if the nonlinear system of equations

$$x_1x_2\cdots x_{n-i}a_{1,1} + x_2x_3\cdots x_{n-i}a_{2,1} + \cdots + x_{n-i}a_{n-i,1} + a_{n-i+1,1} = 0,$$

$$x_2x_3\cdots x_{n-i+1}a_{2,2} + x_3x_4\cdots x_{n-i+1}a_{3,2} + \cdots + x_{n-i+1}a_{n-i+1,2} + a_{n-i+2,2} = 0,$$

$$\vdots$$

$$x_ix_{i+1}\cdots x_{n-1}a_{i,i} + x_{i+1}x_{i+2}\cdots x_{n-1}a_{i+1,i} + \cdots + x_{n-1}a_{n-1,i} + a_{n,i} = 0$$

has a solution in the indeterminates $x_1, x_2,\dots, x_{n-1}$. This can be seen by writing the product $X^{-1}A$ out elementwise and setting the appropriate terms equal to zero.

In this paper we employ an algorithm which attempts to solve the system using a minimal number of variables. We call the algorithm *minimal-variable oblique elimination*. If we take $x_1 = x_2 = \cdots = x_{n-i+1} = 0$, then the system reduces to

$$x_{n-i}a_{n-i,1} + a_{n-i+1,1} = 0,$$

$$x_{n-i}x_{n-i+1}a_{n-i,2} + x_{n-i+1}a_{n-i+1,2} + a_{n-i+2,2} = 0,$$

$$\vdots$$

$$x_{n-i}x_{n-i+1}\cdots x_{n-1}a_{n-i,i}$$
$$+ x_{n-i+1}x_{n-i+2}\cdots x_{n-1}a_{n-i+1,i} + \cdots + x_{n-1}a_{n-1,i} + a_{n,i} = 0.$$

A solution to this system, if it exists, is given by

$$x_{n-i} = -\frac{a_{n-i+1,1}}{a_{n-i,1}},$$

$$x_{n-i+1} = -\frac{a_{n-i+2,2}}{x_{n-i}a_{n-i,2} + a_{n-i+1,2}},$$

$$\vdots$$

$$x_{n-1} = -\frac{a_{n,i}}{x_{n-i}x_{n-i+1}\cdots x_{n-2}a_{n-i,i} + \cdots + x_{n-2}a_{n-2,i} + a_{n-1,i}}.$$

We shall refer to this particular solution set as S. Note that the existence of the solution set S is not equivalent to the existence of a solution to the original system of equations, even with $x_1 = x_2 = \cdots = x_{n-i+1} = 0$, since if A is the zero matrix, then the system is solved trivially but the solution set S is undefined. We shall concern ourselves with determining conditions under which this solution exists.

Definition 17. Let A be an $n \times n$ lower triangular matrix with lower bandwidth $n - i + 1$. If the above solution exists for A, then we call it the *minimal-variable solution.* If X^{-1} is constructed from this solution in the fashion described in this section, then we say that $B = X^{-1}A$ is computed from A using minimal-variable oblique elimination. X is called the *minimal-variable solution matrix* for A.

Definition 18. Let L be an $n \times n$ lower triangular matrix, and denote $L_0 \equiv L^{-1}$. We say that *minimal-variable oblique elimination is successful for* L if there exists matrices $L_1, L_2, \ldots, L_{n-2}$ such that L_i is the minimal-variable solution matrix for $L_{i-1}^{-1}L_{i-2}^{-1}\cdots L_1^{-1}L_0$ for every $i \in [1, n-2]$.

Definition 19. Let A be any $n \times n$ matrix. We say that *minimal-variable oblique elimination is successful for* A if the following two conditions hold:

(1) A has an LU decomposition $A = LU$.
(2) Minimal-variable oblique elimination is successful for L and U^t.

Necessary and Sufficient Conditions for the Minimal-Variable Solution to Exist

We now derive necessary and sufficient conditions for the minimal-variable solution to exist for a lower triangular banded matrix A. We shall show that the solution exists if and only if certain submatrices of A are invertible. We then use these conditions to develop necessary and sufficient conditions for minimal-variable oblique elimination to be successful for any square matrix.

If B is a square matrix, then $\det(B)$ denotes the determinant of B. Fix $i \in [1, n]$, and let A be a lower triangular matrix with lower bandwidth $n - i + 1$, that is,

$$A = \begin{bmatrix}
a_{1,1} & 0 & \cdots & \cdots & \cdots & \cdots & 0\\
\vdots & a_{2,2} & \ddots & & & & \vdots\\
\vdots & & \ddots & \ddots & & & \vdots\\
a_{n-i+1,1} & & & \ddots & 0 & & \vdots\\
0 & a_{n-i+2,2} & \cdots & \cdots & a_{i,i} & \ddots & \vdots\\
\vdots & 0 & \ddots & & & \ddots & \vdots\\
\vdots & \vdots & & \ddots & & & 0\\
0 & 0 & \cdots & 0 & a_{n,i} & \cdots & a_{n,n}
\end{bmatrix}.$$

For each $k \in [1, i]$, let M_k denote the $k \times k$ submatrix of A defined by

$$M_k \equiv \begin{bmatrix}
a_{n-i,1} & a_{n-i,2} & \cdots & \cdots & a_{n-i,k-1} & a_{n-i,k}\\
a_{n-i+1,1} & a_{n-i+1,2} & \cdots & \cdots & a_{n-i+1,k-1} & a_{n-i+1,k}\\
0 & a_{n-i+2,2} & \cdots & \cdots & a_{n-i+2,k-1} & a_{n-i+2,k}\\
0 & 0 & & & \vdots & \vdots\\
\vdots & \vdots & \ddots & & \vdots & \vdots\\
0 & 0 & \cdots & 0 & a_{n-i+k-1,k-1} & a_{n-i+k-1,k}
\end{bmatrix}.$$

For convenience we take $M_0 \equiv (1)$. Note that the M_k are in upper Hessenberg form. We will show that the minimum variable solution exists for A if and only if the M_k are all invertible.

For every $k \in [0, i-1]$ we formally define the symbol d_{n-i+k} by

$$d_{n-i+k} \equiv \left(\prod_{j=0}^{k-1} x_{n-i+j} \right) a_{n-i,k+1} + \left(\prod_{j=1}^{k-1} x_{n-i+j} \right) a_{n-i+1,k+1}$$
$$+ \cdots + x_{n-i+k-1} a_{n-i+k-1,k+1} + a_{n-i+k,k+1}.$$

The above definition is formal in the sense that if upon substituting values for the symbols x_{n-i+j} in a sequential fashion, it happens that $d_{n-i+j} = 0$ for some j, then x_{n-i+k} is undefined for $k \geqslant j$. To show that the minimal variable solution exists it must be shown that the d_{n-i+k} are all defined and nonzero.

In what follows, some of the proofs are by induction. For purposes of illustrating the reasons for the truth of the theorems, the first several cases are sometimes shown to be true even though this is not a logical necessity.

LEMMA 20. *Assume that there exists an* $l \in [1, i]$ *such that* $\det(M_k) \neq 0$ *for every* $k \in [0, l-1]$. *Then for every such* k, d_{n-i+k} *exists and* $d_{n-i+k} = \det(M_{k+1})/\det(M_k)$.

Proof. The proof is by induction on k. If $k = 0$, then

$$d_{n-i} = a_{n-i,1} = \frac{\det(M_1)}{\det(M_0)}.$$

If $l = 1$, then we are done. Otherwise $d_{n-i} \neq 0$ by assumption. If $k = 1$, then

$$d_{n-i+1} = -\frac{a_{n-i+1,1}}{a_{n-i,1}} a_{n-i,2} + a_{n-i+1,2} = \frac{\det(M_2)}{\det(M_1)}.$$

Assume that the lemma is true for $d_{n-i}, d_{n-i+1}, \ldots, d_{n-i+k-1}$ for some $k \in [1, l-1]$. Then each x_{n-i+j} for $j \in [0, k-1]$ is defined so d_{n-i+k} is defined and is given by

$$d_{n-i+k} = \left(\prod_{j=0}^{k-1} x_{n-i+j} \right) a_{n-i,k+1} + \left(\prod_{j=1}^{k-1} x_{n-i+j} \right) a_{n-i+1,k+1}$$
$$+ \cdots + x_{n-i+k-1} a_{n-i+k-1,k+1} + a_{n-i+k,k+1}.$$

By the induction hypothesis,

$$d_{n-i+k} = (-1)^k \left(\frac{a_{n-i+1,1}}{\det(M_1)} \right) \left(\frac{\frac{a_{n-i+2,2}}{\det(M_2)}}{\det(M_1)} \right) \left(\frac{\frac{a_{n-i+3,3}}{\det(M_3)}}{\det(M_2)} \right) \cdots \left(\frac{\frac{a_{n-i+k,k}}{\det(M_k)}}{\det(M_{k-1})} \right) a_{n-i,k+1}$$

$$+(-1)^{k-1} \left(\frac{\frac{a_{n-i+2,2}}{\det(M_2)}}{\det(M_1)} \right) \left(\frac{\frac{a_{n-i+3,3}}{\det(M_3)}}{\det(M_2)} \right) \cdots \left(\frac{\frac{a_{n-i+k,k}}{\det(M_k)}}{\det(M_{k-1})} \right) a_{n-i+1,k+1}$$

$$+ \cdots + (-1) \left(\frac{\frac{a_{n-i+k,k}}{\det(M_k)}}{\det(M_{k-1})} \right) a_{n-i+k-1,k+1} + a_{n-i+k,k+1}$$

$$= \frac{\begin{array}{c}(-1)^k \left(\prod_{j=1}^{k} a_{n-i+j,j} \right) a_{n-i,k+1} + (-1)^{k-1} \left(\prod_{j=2}^{k} a_{n-i+j,j} \right) a_{n-i+1,k+1} + \\ \cdots + (-1) \det(M_{k-1}) a_{n-i+k,k} a_{n-i+k-1,k+1} + \det(M_k) a_{n-i+k,k+1}\end{array}}{\det(M_k)}$$

Since M_{k+1} is in upper Hessenberg form, the numerator of the last expression is $\det(M_{k+1})$ expanded along the column $k+1$. This proves the lemma. ∎

LEMMA 21. *Assume that there exists an $l \in [0, i-1]$ such that the d_{n-i+k} are defined and nonzero for every $k \in [0, l]$. Then $\det(M_k) \neq 0$ for every $k \in [0, l]$.*

Proof. The proof is by induction on k. If $k = 0$, then $\det(M_0) = 1$. If $l = 0$, then we are done. Otherwise if $k = 1$, then $\det(M_1) = a_{n-i,1} = d_{n-i} \neq 0$. If $k = 2$, then

$$d_{n-i+1} = -\frac{a_{n-i+1,1}}{a_{n-i,1}} a_{n-i,2} + a_{n-i+1,2} = -\frac{1}{a_{n-i,1}} \det(M_2) \neq 0.$$

Assume that the lemma is true for $M_0, M_1, \ldots, M_{k-1}$ for some $k \in [2, l]$. Since $\det(M_0), \det(M_1), \ldots, \det(M_{k-1}) \neq 0$, by the calculations of the previous lemma,

$$d_{n-i+k-1} = \frac{\det(M_k)}{\det(M_{k-1})}.$$

By hypothesis, $d_{n-i+k-1} \neq 0$, so we may conclude that $\det(M_k) \neq 0$, which proves the lemma. ■

Lemmas 20 and 21 can be combined to yield necessary and sufficient conditions for the minimal-variable solution to exist.

THEOREM 22. *The minimal-variable solution exists for A if and only if the matrices $M_1, M_2, \ldots, M_i$ are invertible. In this case we have*

$$x_{n-i+k} = -a_{n-i+k+1,k+1}\frac{\det(M_k)}{\det(M_{k+1})}$$

for every $k \in [0, i-1]$.

Proof. Assume that the minimal-variable solution exists for A. Then $d_{n-i+k} \neq 0$ for every $k \in [0, i-1]$. By Lemma 21, $\det(M_k) \neq 0$ for every $k \in [0, i-1]$. By Lemma 20, $d_{n-i+k} = \det(M_{k+1})/\det(M_k)$ which yields the desired expression for x_{n-i+k}.

Conversely, assume that the matrices $M_1, M_2, \ldots, M_i$ are invertible. Then, by Lemma 20, for every $k \in [0, i-1]$, d_{n-i+k} exists and is nonzero. Hence, the minimal-variable solution exists. ■

We now assume that $L \equiv L_0$ is lower triangular and that A is of the form $A = L_{i-1}^{-1}, L_{i-2}^{-1} \cdots L_1 L_0$, where $L_1, L_2, \ldots,$ and L_{i-1} have been constructed using minimal-variable oblique elimination. In the next theorem, we establish criteria which will allow us to deduce the existence (or nonexistence) of the minimal-variable solution for A using information contained in the matrix L. We shall show that, due to the sparse structure of the L_i, one of the matrices M_k will be singular if and only if certain submatrices of L are singular.

DEFINITION 23. If L is an $n \times n$ lower triangular matrix, then for every pair of integers (j, k) with $j \in [1, n-1]$ and $k \in [1, n-j]$ define

$$L_{kj} = \begin{bmatrix} l_{j,1} & \cdots & l_{j,k} \\ \vdots & & \vdots \\ l_{j+k-1,1} & \cdots & l_{j+k-1,k} \end{bmatrix}.$$

Similarly, if U is an $n \times n$ upper triangular matrix, then define

$$U_{kj} \equiv \left[(U^t)_{kj}\right]^t.$$

THEOREM 24. *Assume that $L = L_0$ is an $n \times n$ lower triangular matrix and that either $i = 1$ or $i \in [2, n-2]$ and $n \times n$ matrices $L_1, L_2, \ldots,$ and L_{i-1} have been constructed using minimal-variable oblique elimination. Suppose that $A = L_{i-1}^{-1}L_{i-2}^{-1} \cdots L_1^{-1}L_0$ and that the minimal-variable solution does not exist for A. Let j be the smallest positive integer such that M_j is singular. Then $L_{j,n-i}$ is singular.*

Proof. We show that $\det(L_{j,n-i}) = 0$. By construction, for $k \in [1, i-1]$,

$$L_k = \left[\begin{array}{c|cccccc}
I_{n-k} & 0 & 0 & \cdot & \cdot & \cdot & 0\\ \hline
(0,\ldots,0 - x^{(k)}_{n-k}) & 1 & 0 & \cdot & \cdot & \cdot & 0\\ \hline
0 & -x^{(k)}_{n-k+1} & 1 & 0 & \cdot & \cdot & 0\\
0 & 0 & -x^{(k)}_{n-k+2} & 1 & & & \cdot\\
\cdot & \cdot & 0 & \cdot & \cdot & & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0\\
0 & 0 & 0 & \cdot & 0 & -x^{(k)}_{n-1} & 1
\end{array}\right].$$

where $\left\{-x^{(k)}_{n-k+j}\right\}_{j=0}^{k-1}$ denotes the minimal-variable solutions for the matrix $L_{k-1}^{-1}L_{k-2}^{-1} \cdots L_1^{-1}$. We use the notation $R_k \to R_k + \alpha R_m$ to express the fact that row k of a matrix is replaced by itself plus a multiple, α, of row m. If B is any $n \times n$ matrix and $k \in [1, i-1]$, then multiplying B on the left by L_k has the effect

$$R_m \to R_m \qquad \text{for} \quad m \in [1, n-k],$$

$$R_m \to R_m - x^{(k)}_{m-1}R_{m-1} \qquad \text{for} \quad m \in [n-k+1, n].$$

Thus, multiplying A on the left by L_{i-1} leaves rows 1 to $n-i+1$ fixed and replaces rows $n-i+2$ to n of A with a linear combination of them and the row directly above them, multiplying $L_{i-1}A$ on the left by L_{i-2} leaves rows 1 to $n-i+2$ fixed and replaces rows $n-i+3$ to n of $L_{i-1}A$ with a linear combination of them and the row directly above them,..., and multiplying $L_{i-j+1} \cdots L_{i-1}A$ on the left by L_{i-j} leaves rows 1 to $n-i+j$ fixed and replaces rows $n-i+j$ to n of $L_{i-j+1} \cdots L_{i-1}A$ with a linear combination of them and the row directly above them. Since the other left multiplications by $L_1, L_2, \ldots, L_{i-j}$ leave rows $n-i, n-i+1, \ldots, n-i+j-1$ of $L_{i-j+1} \cdots L_{i-1}A$ unchanged and $L = (\prod_{m=1}^{i-1} L_m)A$, it follows that rows $n-i, n-i+1, \ldots, n-i+j-1$ of L are linear combinations of rows

$n-i, n-i+1, \ldots, n-i+j-1$ of A. In fact, we can write $L_{j,n-i} = \hat{L}_1\hat{L}_2 \cdots \hat{L}_{i-1}M_j$, where

$$\hat{L}_k = \left[\begin{array}{c|cccccccc} I_{i-k} & 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ \hline (0,\ldots,0 - x^{(k)}_{n-k}) & 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ \hline 0 & -x^{(k)}_{n-k+1} & 1 & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & 0 & -x^{(k)}_{n-k+2} & 1 & \cdot & & & & \cdot \\ \cdot & \cdot & 0 & \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot & & \cdot & 0 \\ 0 & 0 & 0 & \cdot & \cdot & \cdot & 0 & -x^{(k)}_{k+j-i} & 1 \end{array}\right].$$

That is, $\hat{L}_k$ is the submatrix of L_k occupying the same position as M_j does in A. Since each $\hat{L}_k$ is unit lower triangular, $\det(\hat{L}_k) = 1$. Hence, $\det(L_{j,n-i}) = \det(M_j) = 0$, which proves the theorem. ∎

Note that if $M = LU$ is an LU decomposition of a $n \times n$ matrix M, then the theorem holds for both L and U^t.

We are now in a position to state necessary and sufficient conditions for minimal-variable oblique elimination to be successful for an arbitrary $n \times n$ matrix. It is immediate from the definition that the matrix must have an LU decomposition. Therefore, we assume this condition in the next theorem.

THEOREM 25. *Assume that B is an $n \times n$ matrix with LU decomposition $B = LU$. Minimal-variable oblique elimination is successful for B if and only if the submatrices L_{kj} and U_{kj} of L and U are invertible for every $j \in [2, n-1]$ and $k \in [1, n-j]$.*

Proof. Assume that the minimal-variable oblique elimination method is successful for B, and assume by way of contradiction that there exists $j \in [1, n-1]$ and $k \in [1, n-j]$ such that L_{kj} is not invertible. Denote $L \equiv L_0$. By Definitions 18 and 19, there exists matrices $L_1, L_2, \ldots, L_{n-2}$ such that for $i \in [1, n-2]$, L_i is the minimal-variable solution matrix for $L_{i-1}^{-1}L_{i-2}^{-1} \cdots L_1^{-1}L$. Let $A = L_{n-j-1}^{-1}L_{n-j-2}^{-1} \cdots L_1^{-1}L$. As was shown in the proof of Theorem 24, $\det(L_{kj}) = \det(M_k)$. By Theorem 22, M_k is invertible, so $\det(M_k) \neq 0$. Since L_{kj} is singular, $\det(L_{kj}) = 0$, which is a contradiction. The same argument can be applied to the matrices U_{kj}.

Conversely, assume that for every $j \in [1, n-1]$ and $k \in [1, n-j]$ the matrices L_{kj} and U_{kj} are invertible and that minimal-variable oblique elimination is not successful for B. Then it is not successful for either L or U^t. Assume that it is not successful for L. Let i be such that we are able to construct the matrices $L_1, L_2, \dots, L_{i-1}$ using minimal-variable oblique elimination and such that the minimal-variable solution does not exist for $A = L_{i-1}^{-1} L_{i-2}^{i-2} \cdots L_1^{-1} L$. Let j be the smallest integer such that M_j is not invertible. By Theorem 5.12, $L_{j,n-j}$ is not invertible. By definition of M_j, $1 \leqslant j \leqslant i \leqslant n-1$, so $n - j \geqslant 1$. Therefore, $L_{j,n-j}$ is invertible, which is a contradiction. If minimal-variable oblique elimination is not successful for U^t, then the same argument can be applied. ■

We have developed criteria which can be applied to the factors of the LU decomposition of a matrix to determine whether or not oblique elimination will be successful for that matrix. This criteria is much easier to use than checking to see if the original system of nonlinear equations has a solution at each step. The conditions stated in Theorem 25 are stringent. One can easily think of a great many examples of matrices which do not satisfy the conditions. Fortunately, all of the Fourier matrices do satisfy the conditions.

Application of Minimal-Variable Oblique Elimination to the Fourier Matrices

We now show that minimal variable oblique elimination is successful for any Fourier matrix. In fact, we show that the technique is successful for PF_n, where P is any permutation matrix. We first show that PF_n has an LU decomposition for any permutation matrix P. We then show that these LU decompositions all satisfy the conditions of Theorem 25 of the previous section. Recall that we denote $\omega \equiv \omega_n \equiv e^{-2\pi i/n}$, where $i = \sqrt{-1}$.

Definition 26. Let B be any $n \times n$ matrix, and let $k \in [1, n]$. The *leading $k \times k$ principal submatrix of B* is the matrix

$$B_k \equiv \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,k} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,k} \\ \vdots & \vdots & & \vdots \\ b_{k,1} & b_{k,2} & \cdots & b_{k,k} \end{bmatrix}.$$

Lemma 27. *Let $k \in [1, n]$, and let $s_1, s_2, \dots, s_k \in [0, n-1]$ be distinct. Let $j \in [0, n-1]$ such that $j + k - 1 < n$. For each $i \in [1, k]$, define*

$$\vec{v} \equiv \left(\omega^{(j+i-1)s_1}, \omega^{(j+i-1)s_2}, \dots, \omega^{(j+i-1)s_k} \right).$$

The set $\{\vec{v}_i\}_{i=1}^{k}$ is a linearly independent set.

Proof. Assume that the lemma is false, that is, that there exists constants $c_1, c_2, \dots, c_k$ not all zero such that $\sum_{i=1}^{k} c_i \vec{v}_i = 0$. Writing the sum out yields

$$\omega^{js_1}\left(c_1 + c_2\omega^{s_1} + c_3\omega^{2s_1} + \cdots + c_k\omega^{(k-1)s_1}\right) = 0,$$

$$\omega^{js_2}\left(c_1 + c_2\omega^{s_2} + c_3\omega^{2s_2} + \cdots + c_k\omega^{(k-1)s_2}\right) = 0,$$

$$\vdots$$

$$\omega^{js_k}\left(c_1 + c_2\omega^{s_k} + c_3\omega^{2s_k} + \cdots + c_k\omega^{(k-1)s_k}\right) = 0.$$

Define a polynomial $p(x) \in \mathbb{C}[x]$ by $p(x) = c_1 + c_2 x + \cdots + c_k x^{k-1}$. Then, since $s_i \neq s_j$ if $i \neq j$ and $s_i \in [0, n-1]$ for every $i \in [1, n]$, the set $\{\omega^{s_1}, \omega^{s_2}, \dots, \omega^{s_k}\}$ is a set of k distinct roots of $p(x)$. By definition of $p(x)$, $\deg(p(x)) \leqslant k - 1$, which implies that $p(x)$ is the zero polynomial. Therefore, $c_1 = c_2 = \cdots = c_k = 0$, which is a contradiction. ■

THEOREM 28. *Let P be an $n \times n$ permutation matrix. Every leading $k \times k$ principal submatrix of PF_n is invertible.*

Proof. Assume that P represents the permutation σ, that is, $P = P_\sigma$. Since in this section we consider matrices and vectors to be ordered from 1 to n rather than 0 to $n - 1$, we consider σ as acting on the set $\{1, 2, \dots, n\}$. Denote

$$F_n = \begin{bmatrix} \vec{f}_1 \\ \vec{f}_2 \\ \vdots \\ \vec{f}_n \end{bmatrix}.$$

where $\vec{f}_i \equiv (1, \omega^{(i-1)}, \omega^{2(i-1)}, \dots, \omega^{(n-1)(i-1)})$. Then

$$PF_n = \begin{bmatrix} \vec{f}_{\sigma(1)} \\ \vec{f}_{\sigma(2)} \\ \vdots \\ \vec{f}_{\sigma(n)} \end{bmatrix}.$$

Let $k \in [1, n]$, and let B_k denote the leading $k \times k$ principal submatrix of PF_n. Thus, letting $s_i \equiv \sigma(i) - 1$,

$$B_k = \begin{bmatrix} 1 & \omega^{s_1} & \omega^{2s_1} & \cdots & \omega^{(k-1)s_1} \\ 1 & \omega^{s_2} & \omega^{2s_2} & \cdots & \omega^{(k-1)s_2} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{s_k} & \omega^{2s_k} & \cdots & \omega^{(k-1)s_k} \end{bmatrix}.$$

Since σ is a permutation, the s_i are distinct. By Lemma 27, the columns of B_k are linearly independent. ■

COROLLARY 29. *Let P be an $n \times n$ permutation matrix. The matrix PF_n has an LU decomposition. In particular, F_n has an LU decomposition.*

Proof. Theorem 28 shows that PF_n satisfies the required conditions [7]. ■

THEOREM 30. *Let P be an $n \times n$ permutation matrix, and assume that $PF_n = LU$ is the LU decomposition of PF_n. Let $j \in [1, n-1]$, $k \in [1, n-j]$, and L_{jk} and U_{jk} be as in Definition 23. The matrices L_{jk} and U_{jk} are invertible.*

Proof. We first show that L_{jk} is invertible. Denote

$$\begin{aligned} \vec{u}_{1k} &\equiv (u_{11}, 0, \dots, 0)^t, \\ \vec{u}_{2k} &\equiv (u_{12}, u_{22}, 0, \dots, 0)^t, \\ \vec{u}_{3k} &\equiv (u_{13}, u_{23}, u_{33}, 0, \dots, 0)^t, \\ &\vdots \\ \vec{u}_{kk} &\equiv (u_{1k}, u_{2k}, u_{3k}, \dots, u_{kk})^t. \end{aligned}$$

Since F_n is nonsingular and L is unit lower triangular, $\det(F_n) = \det(P^{-1})\det(L)\det(U) = \det(P^{-1})\det(U) \neq 0$. Hence, $\det(U) = \prod_{i=1}^{n} u_{ii} \neq 0$. This implies that the scalars $u_{11}, u_{22}, \dots, u_{kk}$ are nonzero and therefore that $\{\vec{u}_{ik}\}_{i=1}^{k}$ is a linearly independent set. Since a square matrix is invertible if and only if it takes a basis to a basis, it is sufficient to show that $\{L_{jk}\vec{u}_{ik}\}_{i=1}^{k}$ is a linearly independent set.

For $i \in [1, k]$, let $\vec{v}_i \equiv (v_{1i}, v_{2i}, \dots, v_{ki})^t \equiv L_{jk}\vec{u}_{ik}$. Since L_{jk} is $k \times k$ and the first k columns of U have zeros in rows $k+1, k+2, \dots, n$, v_{mi} is the $(j+m-1, i)$ element of PF_n. Hence, letting $s_m \equiv \sigma(j+m-1)-1$ so that $v_{mi} = \omega^{(i-1)s_m}$, we have

$$\begin{aligned}
\vec{v}_1 &= (1, 1, \dots, 1)^t,\\
\vec{v}_2 &= \left(\omega^{s_1}, \omega^{s_2}, \dots, \omega^{s_k}\right)^t,\\
\vec{v}_3 &= \left(\omega^{2s_1}, \omega^{2s_2}, \dots, \omega^{2s_k}\right)^t,\\
&\vdots\\
\vec{v}_k &= \left(\omega^{(k-1)s_1}, \omega^{(k-1)s_2}, \dots, \omega^{(k-1)s_k}\right)^t.
\end{aligned}$$

By Lemma 27 with $j = 0$, $\{\vec{v}_i\}_{i=1}^k$ is a linearly independent set. Hence, L_{jk} is invertible.

We now show that U_{jk} is invertible. The argument is almost the same, but some modifications are necessary. Denote

$$\begin{aligned}
\vec{l}_{1k} &\equiv (l_{11}, 0, \dots, 0)^t,\\
\vec{l}_{2k} &\equiv (l_{12}, l_{22}, 0, \dots, 0)^t,\\
\vec{l}_{3k} &\equiv (l_{13}, l_{23}, l_{33}, 0, \dots, 0)^t,\\
&\vdots\\
\vec{l}_{kk} &\equiv (l_{1k}, l_{2k}, l_{3k}, \dots, l_{kk})^t.
\end{aligned}$$

Since L is unit lower triangular, $\{\vec{l}_{ik}\}_{i=1}^k$ is a linearly independent set. We show that $\{U_{jk}\vec{l}_{ik}\}_{i=1}^k$ is a linearly independent set. For $i \in [1, k]$, let $\vec{v}_i \equiv (v_{1i}, v_{2i}, \dots, v_{ki})^t \equiv U_{jk}^t\vec{l}_{ik}$. Then v_{mi} is the $(j+m-1, i)$ element of $(PF_n)^t$ and is therefore the $(i, j+m-1)$ element of PF_n. Hence, letting $s_i \equiv \sigma(i) - 1$, we have

$$\begin{aligned}
\vec{v}_1 &= \left(\omega^{js_1}, \omega^{(j+1)s_1}, \dots, \omega^{(j+k-1)s_1}\right)^t,\\
\vec{v}_2 &= \left(\omega^{js_2}, \omega^{(j+1)s_2}, \dots, \omega^{(j+k-1)s_2}\right)^t,\\
&\vdots\\
\vec{v}_k &= \left(\omega^{js_k}, \omega^{(j+1)s_k}, \dots, \omega^{(j+k-1)s_k}\right)^t.
\end{aligned}$$

Let M be the $k \times k$ matrix having $\vec{v}_i$ in the ith row. Since the column rank of a matrix is equal to the row rank, the $\vec{v}_i$ will be linearly independent if and only if the columns of M are. By Lemma 27, the columns of M are linearly independent. Hence, U_{jk} is invertible. ■

We are now in a position to prove the main result of this section.

COROLLARY 31. *Let P be an $n \times n$ permutation matrix. Minimal-variable oblique elimination is successful for PF_n, that is, there exists unit lower bidiagonal matrices $L_1, L_2, \dots, L_{n-2}$, unit upper bidiagonal matrices $U_1, U_2, \dots, U_{n-2}$, and bidiagonal matrices B and C such that*

$$PF_n = L_1 L_2 \cdots L_{n-2} B C U_{n-2} U_{n-3} \cdots U_1.$$

Proof. By Theorem 30, for every $j \in [1, n]$ and $k \in [1, n-j+1]$, the matrices L_{jk} and U_{jk} are invertible. By Theorem 25, this implies that minimal-variable oblique elimination is successful for PF_n. ■

The next theorem could be of interest for implementation purposes.

THEOREM 31. *For every $n > 2$, the matrices B and C in Corollary 31 can be chosen such that $U_j = L_j^t$ for $j \in [1, n-2]$, that is,*

$$F_n = L_1 L_2 \cdots L_{n-2} B C L_{n-2}^t L_{n-3}^t \cdots L_1^t.$$

Proof. Let $F_n = LU$ be the LU decomposition of F_n. Since F_n is symmetric, $U = DL^t$, where D is a diagonal matrix [7]. Suppose that $L = L_1 L_2 \cdots L_{n-2} B$ is the tridiagonal decomposition of L computed using minimal-variable oblique elimination. Then $U^t = LD = L_1 L_2 \cdots L_{n-2} BD$. Take $C = DB^t$. ■

4. COMPLEXITY CONSIDERATIONS

In this section we derive upper bounds on the number of parallel steps required to implement the algorithms implied by the decompositions derived in the previous sections. The form of the expressions for the number of steps required using the FFT-based method is highly dependent on n. In the case of the parallel algorithms resulting from minimal-variable oblique elimination, it is straightforward to see that the number of parallel arithmetic steps

TABLE 1

ESTIMATED NUMBER OF PARALLEL STEPS REQUIRED TO IMPLEMENT F_n LOCALLY[a]

	C_r		C_m		
n	GR	TF	GR	TF	$C_a(F_n)$
100 (25×4)	227	316	9	8	26
120 $(8\times5\times3)$	367	400	8	6	21
128	356	—	6	—	7
256	736	—	7	—	8
360 $(9\times8\times5)$	1096	1161	11	9	27
500 (125×4)	1506	1648	21	20	38
512	1500	—	8	—	9

[a]GR denotes use of the general radix identity. TF denotes use of the twiddle-free identity. C_r denotes permutation steps. C_m denotes multiplication steps. C_a denotes addition steps.

required is $2n-2$ for every n, where each step requires one addition and one multiplication. We also show that oblique elimination can be implemented using $n^3+o(n^2)$ complex floating-point operations. Given a tridiagonal decomposition of a matrix, we consider the parallel complexity to be the number of tridiagonal matrices appearing in the decomposition plus the number of parallel steps required to implement the permutations.

We first consider the FFT-based decompositions. Due to the structure of the FFT-based decompositions, the complexity can be divided into multiplication, addition, and permutation terms. We briefly consider the parallel multiplicative and additive complexity of the algorithms implied by the previous decompositions. Since these decompositions are related to FFTs, these complexity counts are similar to those of FFTs. The number of parallel steps required to implement the permutations is then considered. It will be seen that it is the latter that dominates the overall complexity in most cases in terms of total number of steps. This can be attributed to the fact that FFTs are based on reindexing schemes. When implementing an algorithm on an architecture such as mesh-connected arrays, reindexing can be an expensive operation. A local permutation step is much less expensive than a floating-point multiplication, however. We present here Tables 1 and 2, indicating the estimated number of parallel steps required in certain cases and the estimated amount of time required in certain cases, respectively. The numbers in Table 1 were calculated using the results of this section. The timing estimates in Table 2 were calculated using the numbers in Table 1 and specific timings for floating-point operations and shifts on the MPP [31]. It is reasonable to assume that these timings are accurate for mesh-connected arrays constructed

TABLE 2
ESTIMATED TIMINGS FOR IMPLEMENTING F_n LOCALLY

	Time (sec)		
n	GR	TF	MVOE[a]
100	.00258	.00269	.02446
120	.00249	.00243	.02958
128	.00148	—	.03156
256	.00233	—	.06324
360	.00444	.00444	.08898
500	.00648	.00673	.12364
512	.00392	—	.12661

[a]Minimal-variable oblique elimination.

using GAPP chips, since in both cases the processing elements (PEs) are one-bit processors with a 10-Mhz clock. The MPP executed up to 400 million floating-point operations per second, the computation being spread out across $16 \times 16 = 4096$ PEs. Moreover, it takes four real multiplications and two additions to execute a complex multiplication. Thus, one can expect approximately $4 \times 10^8/6(4096)$ complex floating-point operations per second. Note that this figure will tend to overestimate the required time, since a complex addition requires only two real additions. The elementary permutations must take two steps, since the data can only be transferred in one direction at a time. Again from [31], it takes 96 cycles to transfer a complex number with 16-bit real and imaginary parts to an adjacent processor. Hence, it takes (2.96 cycles)$(1/10^8$ sec/cycle$) = 192/10^8$ second to execute an elementary permutation step. Table 2 was constructed using this reasoning.

Some observations concerning Table 2 can be made. The timings in the table suggest that either technique could be useful. It appears that the FFT-based method would run faster. In fact, this may not be the case, since the FFT-based method requires a large variety of different instructions to be executed in an intermingled fashion, whereas the MVOE method requires only the repetition of two instructions. Thus, it would be worthwhile to implement both algorithms on mesh-connected arrays.

For any matrix A we denote by $C_m(A)$ and $C_a(A)$ the number of parallel multiplication and addition steps required to implement A locally on a linear array. By a parallel permutation step we mean the action of switching the data in horizontally or vertically adjacent cells. We denote by $C_r(A)$ the number of parallel permutation, or data-routing, steps required to implement A locally. We acknowledge that there is some ambiguity in this notation, since there can be more than one decomposition available for a given A, and the complexity measures with respect to the different decompositions may be different. Either it will be explicitly stated which decomposition is being used or it will be clear from the context. Note that if $x = m$, a, or r, then $C_x(AB) = C_x(A) + C_x(B)$ and, since we are considering parallel complexity, $C_x(I_j \otimes A) = C_x(A)$ for any matrices A and B and positive integer j.

We first give expressions for the number of parallel multiplication and addition steps. Since these quantities are so similar to standard FFT operation counts, we simply state them for completeness. They follow immediately from the decompositions given in Section 2. We suppress the multiplication by $n^{-1/2}$.

THEOREM 32. *Let n be as in Theorem* 7.

(1) *Using Corollary* 8 *followed by Corollary* 11 *to decompose* F_n *results in*

$$C_m(F_n) = \left[\sum_{j=1}^{s} \left(k_j C_m(F_{p_j}) + k_j \right) \right] - s.$$

(2) *Using Corollary* 10 *followed by Corollary* 11 *to decompose* F_n *results in*

$$C_m(F_n) = \left[\sum_{j=1}^{s} \left(k_j C_m(F_{p_j}) + k_j \right) \right] - 1,$$

where we allow s to be equal to 1.

(3) *If p is an odd prime, then using Theorem* 12 *results in*

$$C_m(F_p) = 2C_m(F_{p-1}) + 1.$$

The difference in the number of parallel multiplications required by the two different methods is $s - 1$ multiplication steps. Note that $C_m(F_2) = 0$ and that if $n = 2^k$ then $C_m(n) = \log_2 n - 1$.

THEOREM 33. *Using either method results in the following expressions for the additive complexities*:

(1) *If* $n = \prod_{j=1}^{s} p_j^{k_j}$ *is not prime, then* $C_a(F_n) = \sum_{j=1}^{s} k_j(C_a(F_{p_j}))$.

(2) *If p is an odd prime, then* $C_a(F_p) = 2C_a(F_{p-1}) + 2(p-1)$.

Note that the additions due to the matrices U_p will have a significant influence on the additive complexity. This is because the method of computing the U_p, although local, is, except for one step, essentially serial.

We now move to a discussion of the permutation complexity. The permutations can be executed locally by the odd-even transposition sort (OETS), which is a parallel algorithm for sorting data arranged in a one-dimensional array. The OETS as applied to a linear array $a(x)$ can be described as follows: Alternate the steps (1) and (2) until no changes take place:

(1) If x is odd and $a(x) < a(x+1)$, then switch $a(x)$ and $a(x+1)$.
(2) If x is even and $a(x) < a(x+1)$, then switch $a(x)$ and $a(x+1)$.

It has been shown that the OETS will execute an arbitrary permutation of n objects on a linear array in at most n parallel steps [17, 33]. Furthermore, it can be shown that, because of the special structure of the shuffle permutations, the OETS will execute the permutation σ_{mk} corresponding to $P(m,k)$ in either $(m-1)(k-1)$ or $(m-1)(k-1)+1$ parallel steps [6]. The proof of this fact is tedious, and we do not give it here. It is fairly easy to convince oneself of the validity of the claim by writing a few of these permutations out and performing the OETS by hand. It is also easy to write a computer program which will perform the OETS on a linear array. In what follows, we make the convention that $\sum_{j=m}^{n} x_j \equiv 0$ if $n < m$.

THEOREM 34. *Upper bounds on the number of parallel permutation steps required to implement the various algorithms are*:

(1) *If n is as in Theorem 7 and Corollary 8 is used to decompose F_n, then*

$$C_r(F_n) \leqslant n + \sum_{j=0}^{s-2} c_{j-1} + \left(p_{s-1}^{k_{s-1}} - 1\right)\left(p_s^{k_s} - 1\right) + 1 + \sum_{j=1}^{s} C_r\left(F_{p_j^{k_j}}\right).$$

(2) *If n is as in Theorem 7 and Corollary 10 is used to decompose F_n, then*

$$C_r(F_n) \leqslant 3n - 2\sum_{j=1}^{s} p_j^{k_j} + 4(s-1) + \sum_{j=1}^{s} C_r\left(F_{p_j^{k_j}}\right).$$

(3) *If $n = p^k$ for $k \geqslant 2$ and Corollary 11 is used to decompose F_n, then*

$$C_r(F_n) \leqslant 3n - 2pk + kC_r(F_p).$$

(4) *If $n = p$ is an odd prime and Theorem* 12 *is used to decompose F_n, then*

$$C_r(F_n) \leqslant 2(n-1) + 2C_r(F_{n-1}).$$

Thus, in any case, the upper bounds on the number of parallel permutation steps is linear in n, that is, $C_r(F_n)$ is $o(n)$.

Proof. (1): Denote $R \equiv \sum_{j=1}^{s} C_r(F_{p_j^{k_j}})$. Since $c_{s-1} = p_s^{k_s}$,

$$C_r(F_n) \leqslant \left[\sum_{j=0}^{s-2} C_r(R_j)\right] + C_r\left(P\left(p_{s-1}^{k_{s-1}}, p_s^{k_s}\right)\right) + R + C_r\left(\prod_{j=1}^{s-1}\left(I_{n_{s-j-1}} \otimes Q_{2(s-j)}\right)\right).$$

By definition of R_j and due to the structure of the shuffle permutations,

$$C_r(F_n) \leqslant C_r\Big(P(q_j, c_j)\big(I_{q_j} \otimes Q_{2j+1}\big)\big(I_{q_j} \otimes P(c_{j+1}, Q_{j+1})\big)\Big) \leqslant q_j c_j = c_{j-1}.$$

Since $P(p_{s-1}^{k_{s-1}}, p_s^{k_s})$ is a shuffle permutation, $C_r(P(p_{s-1}^{k_{s-1}}, p_s^{k_s})) \leqslant (p_{s-1}^{k_{s-1}} - 1)(p_s^{k_s} - 1) + 1$.

The permutation $Q = \prod_{j=1}^{s-1}(I_{n_{s-j-1}} \otimes Q_{2(s-j)})$ can be implemented as a sequence of permutations or as a one-step permutation. Note that $Q_2 = T_{q_1}(C_{c_1}^{q_1^*})P(c_1, q_1)$, so $C_r(Q_2) \leqslant c_1 + (c_1 - 1)(q_1 - 1) + 1 = n - q_1 + 2$, which can be as large as n. Therefore, it is advisable, in general, to implement Q as a one-step permutation. Combining these observations yields

$$C_r(F_n) \leqslant C_r(Q) + \sum_{j=0}^{s-2} c_{j-1} + \left(p_{s-1}^{k_{s-1}} - 1\right)\left(p_s^{k_s} - 1\right) + 1 + R,$$

which proves (1).

(2): Let R be as in (1). By Corollary 10,

$$C_r(F_n) \leqslant 2\sum_{j=1}^{s-1} C_r\left(P_{c_{j-1}}\right) + C_r\left(\prod_{j=2}^{s}\left(I_{n_{s-j}} \otimes P_{c_{s-j}}\right)\right) + R.$$

Since the $P_{c_{j-1}} = P(c_j, p_j^{k_j})$ are shuffle permutations, $C_r(P_{c_{j-1}}) \leqslant (c_j - 1)(p_j^{k_j} - 1) + 1$. Note that $c_j p_j^{k_j} = c_{j-1}$. Hence

$$\sum_{j=1}^{s-1} C_r\left(P_{c_{j-1}}\right) \leqslant \sum_{j=1}^{s-1} c_{j-1} - \sum_{j=1}^{s-1} c_j - \sum_{j=1}^{s-1} p_j^{k_j} + 2(s-1) = n - \sum_{j=1}^{s} p_j^{k_j} + 2(s-1).$$

By similar reasoning to that used in (1) for the permutation matrix Q, $C_r(\prod_{j=2}^{s}(I_{n_{s-j}} \otimes P_{c_{s-j}})) \leqslant n$. Thus, $C_r(F_n) \leqslant 2n + 4(s-1) - 2\sum_{j=1}^{s-1} p_j^{k_j} + n + R$, which proves (2).

(3): By Corollary 11

$$\begin{aligned} C_r(F_n) &= C_r\big(G_p(I_{p^{k-1}} \otimes F_p)H_p\big) \\ &= 2\sum_{m=0}^{k-2} C_r\big(P(p^{k-m-1}, p)\big) + \sum_{m=0}^{k-1} C_r(F_p) + C_r(H_p) \\ &\leqslant 2\sum_{m=0}^{k-2} \big[(p^{k-m-1}-1)(p-1)+1\big] + kC_r(F_p) + n. \end{aligned}$$

A little algebra yields

$$\begin{aligned} &\sum_{m=0}^{k-2} \big[(p^{k-m-1}-1)(p-1)+1\big] \\ &\qquad = p^{k-1}(p-1)\left[\sum_{m=0}^{k-2}\left(\frac{1}{p}\right)^m\right] - (k-1)(p-1) + (k-1) \\ &\qquad = n - pk. \end{aligned}$$

Therefore $C_r(F_n) \leqslant 3n - 2pk + kC_r(F_p)$, which proves (3).

(4): This inequality follows immediately from Theorem 4.12. ■

The upper bounds on the permutation complexities can be sharpened in certain special cases. This is because the structure of the permutations is not destroyed (or clouded over) by the presence of too many permutation matrices. Note that the twiddle-free identity can be rewritten as

$$F_n = P(k,m)T_k(C_m^{-k*})(I_k \otimes F_m)P(m,k)(I_m \otimes F_k)T_m(C_k^{m*})P(k,m).$$

Using this expression, it is straightforward to verify the following theorem.

THEOREM 35. *If n is as in Theorem 7 with $s = 2$, then:*

(1) *If the twiddle-free identity is used to decompose F_n, then*

$$C_r(F_n) \leqslant 3n + 6 - 2p_2^{k_2} - 2p_1^{k_1} + C_r(F_{p_1^{k_1}}) + C_r(F_{p_2^{k_2}}).$$

(2) *If the general radix identity is used to decompose F_n, then*

$$C_r(F_n) \leqslant 3(n+2) - 3\left(p_1^{k_1} + p_2^{k_2}\right) + C_r\left(F_{p_1^{k_1}}\right) + C_r\left(F_{p_2^{k_2}}\right).$$

(3) *If $n = p^2$, then*

$$C_r(F_n) \leqslant 3(p-1)^2 + 3 + 2C_r(F_p).$$

We now derive expressions for the time complexity of the parallel and serial algorithms resulting from the decompositions developed using minimal-variable oblique elimination. We also establish upper bounds on the number of floating-point operations required to compute the decompositions using minimal-variable oblique elimination. Recall that the algorithms resulting from the decompositions are generally required to be real-time algorithms, whereas the computation of the tridiagonal factors need not be.

THEOREM 36. *Let M be an $n \times n$ matrix, and assume that minimal-variable oblique elimination is successful for M. It takes $2n-2$ parallel steps—all steps but one consisting of, at most, one multiplication and one addition per point, and the other consisting of two multiplications and one addition per point—to implement the transformation $\vec{v} \to M\vec{v}$ using the tridiagonal decomposition of M obtained from using minimal-variable oblique elimination.*

Proof. The result follows immediately from Corollary 31. ■

We shall use the concept of a flop (floating-point operation) to quantify the complexity of a serial computation. Golub defines a flop to be "more or less the work associated with the statement $s := s + a_{ik}b_{kj}$" [7]. We take s, a_{ik}, and b_{kj} to be complex numbers. Note that it takes at least three real multiplications and five additions or four real multiplications and two additions to perform a complex multiplication [1].

THEOREM 37. *Let M be an $n \times n$ matrix, and assume that minimal-variable oblique elimination is successful for M. The mapping $\vec{v} \to M\vec{v}$ can be accomplished with $n^2 + 2$ flops using the decomposition resulting from oblique elimination.*

Proof. If $\vec{x} \in \mathbf{C}^n$ and $\vec{y} = L_i\vec{x}$ for some $i \in [1, n-2]$, then, by the way the L_i are constructed, it takes one flop to compute y_k if $k \geqslant n - i + 1$, and zero otherwise. The same statement holds true for the U_i. If $\vec{y} = C\vec{x}$, then it takes two flops to compute y_k for $k \in [1, n]$. If $\vec{y} = B\vec{x}$, then, since B is unit lower bidiagonal, it takes one flop to compute y_k for $k \in [1, n]$. Hence, it takes a total of $2\Sigma_{i=1}^{n-2} i + 3n = n^2 + 2$ flops. ■

The operation count associated with using the tridiagonal decompositions to implement linear transforms on a serial machine is essentially the same as the operation count of the straightforward method. Therefore, these decompositions do not lead to good serial algorithms for computing linear transforms in general.

We now show that if minimal-variable oblique elimination is successful, then it takes $o(n^3)$ flops to compute the decomposition. The computations required to compute the decompositions can be divided into three categories. One category consists of the computations necessary to compute the LU decomposition. The other categories consist of the computations necessary to construct the minimal-variable solution matrices and the matrix multiplications necessary to compute the intermediate results, that is, the multiplications of the form $L_i^{-1}(L_{i-1}^{-1} \cdots L_1^{-1}L)$. The first category is well studied. We examine the last two categories.

We first consider the computations necessary to construct the minimal-variable solution matrices. In the next theorem, we establish an upper bound on the number of flops required to compute the minimal-variable solution matrix for a lower triangular, banded matrix A, assuming that a Horner-type algorithm is used.

THEOREM 38. *Let $i \in [1, n-2]$, and let A be an $n \times n$ matrix with lower bandwidth $n - i + 1$. For each $k \in [0, i-1]$ assume that x_{n-i+k} is computed by first computing d_{n-i+k} in the nested fashion*

$$d_{n-i+k} = x_{n-i+k-1}\Big(x_{n-i+k-2}\Big(\cdots\big(x_{n-i+1}\big(x_{n-i}a_{n-i,k+1} + a_{n-i+1,k+1}\big)$$

$$+ a_{n-i+2,k+1}\big) + \cdots\Big) + a_{n-i+k-1,k+1}\Big) + a_{n-i+k,k+1}a_{n-i+k,k+1}$$

and then computing

$$x_{n-i+k} = -\frac{a_{n-i+k+1,k+1}}{d_{n-i+k}}.$$

It takes $i(i+1)/2$ flops to compute the minimal variable solution for A.

Proof. It takes k flops to compute d_{n-i+k}, one for each x_{n-i+j}, $j = 0, 1, \ldots, k-1$. It takes one more division to compute x_{n-i+k}. Hence it takes $\sum_{k=0}^{i-1}(k+1) = i(i+1)/2$ flops altogether. ∎

Corollary 39. *Let M be an $n \times n$ matrix, and assume that minimal-variable oblique elimination is successful for M. Using the method of Theorem 38 it takes $\frac{1}{3}n(n-1)(n-2) = o(n^3)$ flops to construct the minimal-variable solution matrices required to factor M.*

Proof. The solution matrices must be computed for $A_1 = L_1^{-1}L$, $A_2 = L_2^{-1}L_1^{-1}L, \ldots, A_{n-2} = L_{n-2}^{-1} \cdots L_1^{-1}L$. It takes $i(i+1)/2$ flops to do so for each A_i. The same must be done for U^t. Hence, it takes $2\sum_{i=1}^{n-2} i(i+1)/2 = \sum_{i=1}^{n-2} i^2 + \sum_{i=1}^{n-2} i = \frac{1}{3}n(n-1)(n-2)$ flops altogether. ∎

We now derive upper bounds on the number of flops required to carry out the matrix multiplications needed to construct the intermediate results. Recall that if $i \in [1, n-2]$, then

$$L_i^{-1} = \left[\begin{array}{c|cccccc} I_{n-i-1} & & & 0 & & & \\ \hline & 1 & 0 & \cdot & \cdots & \cdot & 0 \\ & x_{n-i} & 1 & 0 & & & \cdot \\ & x_{n-i}x_{n-i+1} & x_{n-i+1} & 1 & \cdot & & \cdot \\ 0 & \vdots & \vdots & & \ddots & & \vdots \\ & \cdot & \cdot & & \cdot & 0 & \cdot \\ & \cdot & \cdot & & & 1 & 0 \\ & \prod_{j=0}^{i-1} x_{n-i+j} & \prod_{j=1}^{i-1} x_{n-i+j} & \cdot & \cdots & x_{n-1} & 1 \end{array}\right].$$

Theorem 40. *Let $i \in [1, n-2]$, and let $A = L_{i-1}^{-1} \cdots L_1^{-1}L$. Let R_k denote the k*th *row of A. Assume that the matrix multiplication is computed in the following nested fashion:*

$$R_{n-i+j} \to x_{n-i+j-1}\Big(x_{n-i+j-2}\big(\cdots x_{n-i+1}(x_{n-i}R_{n-i} + R_{n-i+1}) + R_{n-i+2}\big) + \cdots + R_{n-i+j-1}\Big) + R_{n-i+j}$$

for $j \in [1, i]$. Then the matrix multiplication can be carried out using $i(n-i+2)$ flops.

Proof. The multiplication leaves rows 1 to $n-i+1$ unchanged. The computation of the $n-i+j$th row is of the form $R_{n-i+j} \to x_{n-i+j-1}E + R_{n-i+j}$, where E is the value used to compute row $n-i+j-1$. Therefore, it takes one flop per row element to compute the new row. Since A is lower triangular with lower bandwidth $n-i+1$, there are at most $n-i+1$ nonzero elements in each row. Furthermore, the nonzero part of each row is offset by one location from the nonzero part of the row directly above and below it. Hence, the linear combination needs to be carried out for only $n-i+2$ elements in each row. Since only i rows are changed, the matrix multiplication can be carried out in $i(n-i+2)$ flops. ■

COROLLARY 41. *Let M be an $n \times n$ matrix, and assume that minimal-variable oblique elimination is successful for M. Let $M = LU$ be the LU decomposition of M. The matrix multiplications necessary to compute the minimal-variable tridiagonal decompositions of L and U can be carried out using $\frac{1}{3}(n-1)(n-2)(n+9)$ flops.*

Proof. The multiplications $L_1^{-1}L$, $L_2^{-1}(L_1^{-1})L, \dots,$ and $L_{n-2}^{-1}(L_{n-3}^{-1} \cdots L_1^{-1}L)$, each of which can be done using $i(n-i+2)$ flops, must be computed. The same must be done for U^t. Hence, it takes a total of $2\Sigma_{i=1}^{n-2} i(n-i+2) = \frac{1}{3}(n-1)(n-2)(n+9)$ flops. ■

We now combine the results obtained in this section to show that oblique elimination is, at most, an $o(n^3)$ operation.

COROLLARY 42. *If minimal-variable oblique elimination is successful for the $n \times n$ matrix M, then it can be carried out using $o(n^3)$ flops.*

Proof. By Corollaries 39 and 41, it takes at the most $o(n^3)$ to compute the tridiagonal decompositions of L and U. Standard algorithms can be used to compute the LU decomposition in $o(n^3)$ flops. Thus, the whole procedure takes $o(n^3)$ flops. ■

In fact, each of the separate operation counts is of the form $\frac{1}{3}n^3 + o(n^2)$, so the total operation count is of the form $n^3 + o(n^2)$.

5. CONCLUSION

We have shown how two different methods can be used to derive tridiagonal decompositions of F_n for arbitrary values of n. The first method is based on use of matrix identities associated with FFTs, and the second is based on use of oblique elimination. These decompositions can be used to develop parallel algorithms for implementing DFTs on linear, or mesh-connected, arrays of processors.

The FFT-based method has the advantage that the number of parallel arithmetic steps required to implement the resulting parallel algorithms is small. A disadvantage associated with this technique is that a large number of permutation matrices appear in the decompositions, which implies a great deal of data manipulation. Another consideration is that these decompositions can require quite different programs to implement for different values of n, as is always the case with FFTs.

The major advantage of the oblique elimination method is that there is very little data manipulation required to implement the algorithms implied by these decompositions. Another advantage is that the parallel arithmetic operation counts are all the same linear function of n for every n regardless of the compositeness of n. Thus, a library of parallel algorithms for computing DFTs of any size (within some upper bound) should be easy to construct using this technique. The same program can be used for any n, which is far from the case with FFT-based methods. This is due to the fact that oblique elimination is a method based on numerical linear algebra rather than traditional discrete Fourier-transform methods. A disadvantage is that the number of multiplication steps is higher than for the FFT-based method, particularly for values of n such as powers of two. We point out, however, that, given a linear array of processors, the oblique elimination method yields a parallel algorithm which takes $o(n)$ arithmetic steps for any n, which is better than any serial FFT algorithm.

The parallel algorithms resulting from the oblique elimination method can be used alone or in conjunction with the FFT-based method. Specifically, the FFT-based method could be used to break the computation into prime components. The method described in this chapter could then be used in place of the Rader prime algorithm and the convolution theorem, since the use of those techniques requires a number of arithmetic and data-manipulation steps. We feel that this will be the best way of using the oblique elimination method.

Oblique elimination is not peculiar to the DFT. Thus, it may be that oblique elimination will work for many special linear transforms. It has been pointed out that the efficient computation of linear transforms is one of the outstanding problems in the development of real-time scene analysis algorithms for robots [28]. Perhaps parallel processing and oblique elimination can be combined to help solve these problems.

I would like to thank Professor Gerhard X. Ritter for encouraging me to work on these problems. This work was partially supported by the Air Force Armament Division and DARPA under Contract F08635-84-C-0295. I am grateful to Mr. Neal Urquhart for his support of this research.

REFERENCES

1 R. Blahut, *Fast Algorithms for Digital Signal Processing*, Addison-Wesley, Reading, Mass., 1985.

2 E. Cloud and W. Holsztynski, Higher efficiency for parallel processors, in *Proceedings IEEE Southcon 84*, Mar. 1984, pp. 416–422.

3 P. J. Davis, *Circulant Matrices*, Wiley, New York, 1979.

4 M. Duff, CLIP4: A large scale integrated circuit array parallel processor, in *3rd International Joint Conference on Pattern Recognition*, 1976.

5 M. Duff and S. Levialdi (Eds.), *Languages and Architectures for Image Processing*, Academic, New York, 1981.

6 P. D. Gader, Image Algebra Techniques for Parallel Computation of Discrete Fourier Transforms and General Linear Transforms, Ph.D. Dissertation, Univ. of Florida, Gainesville, 1986.

7 G. H. Golub and C. F. van Loan, *Matrix Computations*, Johns Hopkins U.P., Baltimore, 1983.

8 R. C. Gonzalez and P. Wintz, *Digital Image Processing*, Addison-Wesley, Reading, Mass., 1977.

9 I. J. Good, The interaction algorithm and practical Fourier analysis, *J. Roy. Statist. Soc.* 20:270–275 (1958).

10 I. J. Good, The relationship between two fast Fourier transforms, *IEEE Trans. Comput.* C-20(3):310–317 (Mar. 1971).

11 R. W. Hamming, *Digital Filters*, Prentice-Hall, Englewood Cliffs, N.J., 1977.

12 G. T. Herman, *Image Reconstruction from Projections*, Academic, New York, 1980.

13 R. Hockney and C. Jesshope, *Parallel Computers: Architecture, Programming, and Algorithms*, Adam Hilger, Bristol, 1981.

14 S. N. Jayaramamurthy, Texture discrimination using digital deconvolution filters, in *Proceedings of the Fifth International Conference on Pattern Recognition*, Miami Beach, Dec. 1980.

15 C. R. Jesshope, The implementation of fast radix two transforms on array processors, *IEEE Trans. Comput.* C-29(1):20–27 (Jan. 1980).

16 D. Kahaner, Matrix description of the fast Fourier transform, *IEEE Trans. Audio Electroacoust.* AU-18:442–450 (1970).

17 D. E. Knuth, *The Art of Computer Programming, Sorting and Searching*, Addison-Wesley, Reading, Mass., 1973.

18 J. H. McClellan and C. M. Rader (Eds.), *Number Theory in Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, N.J., 1979.

19 J. von Neumann, *Theory of Self-Reproducing Automata*, Univ. of Illinois Press, Urbana, Ill., 1966.

20 D. E. Oldfield and S. F. Reddaway, An image understanding performance study on the ICL distributed array processor, in *IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, 1985, pp. 256–265.

21 B. N. Parlett, Winograd's Fourier transform via circulants, *Linear Algebra Appl.* 45:137–155 (1982).

22 M. C. Pease, An adaptation of the fast Fourier transform for parallel processing, *J. Assoc. Comput. Mach.* 15(2):253–264 (Apr. 1968).

23 J. L. Potter, Image processing on the massively parallel processor, *Computer* 16(1):62–68 (Jan. 1983).

24 W. K. Pratt, *Digital Image Processing*, Wiley, New York, 1978.

25 M. M. Rahman, E. A. Quincy, R. G. Jacquot, and M. J. Magee, Pattern recognition techniques in cloud research, in *Proceedings of the Fifth International Conference on Pattern Recognition*, Miami Beach, Dec. 1980.

26 G. X. Ritter, P. D. Gader, and J. L. Davidson, Automated bridge detection in FLIR images, in *Proceedings of the Eighth International Conference on Pattern Recognition*, Paris, to appear.

27 D. J. Rose, Matrix identities of the fast Fourier transform, *Linear Algebra Appl.* 29:423–443 (1980).

28 J. T. Schwartz, Mathematics addresses problems in computer vision for advanced robots, *SIAM News* 18(3):3 (1985).

29 T. M. Silverberg, The Hough transform on the geometric arithmetic array processor, in *IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, Nov. 1985, pp. 387–394.

30 J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer, New York, 1980.

31 J. P. Strong, The Fourier transform on mesh connected arrays such as the massively parallel processor, in *IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, Nov. 1985, pp. 190–197.

32 M. Tchuente, Parallel calculationa of a linear mapping on a computer network, *Linear Algebra Appl.* 28:223–247 (1979).

33 M. Tchuente, Parallel realization of permutations over trees, *Discrete Math.* 39:211–214 (1982).

34 F. Theilheimer, A matrix version of the fast Fourier transform, *IEEE Trans. Audio Electroacoust.* AU-17:158–161 (1969).

35 S. M. Ulam, On Some New Possibilities in the Organization and Use of Computing Machines, IBM Research Report RC68, May, 1957.

36 J. S. Wiejak, H. Buxton, and B. F. Buxton, Convolution with separable masks for early image processing, *Comput. Vision Graphics and Image Process.* 32:279–290 (1985).

Received 18 August 1986; final manuscript accepted 11 June 1987