MATHEMATICS OF COMPUTATION, VOLUME 31, NUMBER 139
JULY 1977, PAGES 740–751

Computational Complexity of Fourier Transforms Over Finite Fields*

By F. P. Preparata and D. V. Sarwate**

Abstract. In this paper we describe a method for computing the Discrete Fourier Transform (DFT) of a sequence of n elements over a finite field $\mathrm{GF}(p^m)$ with a number of bit operations $O(nm \log(nm) \cdot P(q))$ where $P(q)$ is the number of bit operations required to multiply two q-bit integers and $q \cong 2 \log_2 n + 4 \log_2 m + 4 \log_2 p$. This method is uniformly applicable to all instances and its order of complexity is not inferior to that of methods whose success depends upon the existence of certain primes. Our algorithm is a combination of known and novel techniques. In particular, the finite-field DFT is at first converted into a finite field convolution; the latter is then implemented as a two-dimensional Fourier transform over the complex field. The key feature of the method is the fusion of these two basic operations into a single integrated procedure centered on the Fast Fourier Transform algorithm.

I. Introduction. The Discrete Fourier Transform (DFT) over a finite field occurs in many applications. It occurs, under the name of Mattson-Solomon polynomials, in the theory of error-correcting codes [1], [2]. Also, in applications of BCH error-correcting codes, the computation of the syndrome of the error pattern and the determination of the locations and values of the errors all correspond to the computation of the DFT of a sequence of elements from a finite field. The DFT over a finite field has been suggested as a means of computing exactly the results of the convolution of sequences of integers [3]–[8] and for implementing the multiplication of very large integers [3], [9]. For these reasons, it is desirable to devise efficient methods of computing the DFT over a finite field.

Pollard [3] has shown that an algorithm, which is the finite field analogue of the well-known Fast Fourier Transform (FFT) algorithm [12], can be applied to computation of the DFT over a finite field. This algorithm requires $O(n(n_1 + n_2 + \cdots + n_l))$ arithmetic operations in the field to compute the DFT of a sequence of n elements where $n = n_1 \cdot n_2 \cdot \ldots \cdot n_l$. Thus, the finite-field FFT algorithm is efficient only when n is highly composite; and in this case, the algorithm requires $O(n \log n)$ arith-

Received August 30, 1976.

AMS (MOS) subject classifications (1970). Primary 68A20, 42A68, 42A96, 12C15; Secondary 68A10, 94A10.

Key words and phrases. Computational complexity, discrete Fourier transform, finite fields, convolution.

*This work was supported in part by the National Science Foundation under Grants MCS76-17321 and GK-24879 and in part by the Joint Services Electronics Program (U.S. Army, U. S. Navy, and U.S. Air Force) under contract DAAB-07-72-C-0259.

**The authors are with the Coordinated Science Laboratory and the Department of Electrical Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

metic operations in the field. Several authors [6]–[9] have considered special cases of the FFT for particular applications, using special properties of carefully chosen fields to achieve computational efficiency. For the general case, however, the efficiency of the FFT algorithm depends upon the vagaries of the factorization of n.

In this paper, we present a method of computing the DFT in a finite field that is uniformly applicable to all finite fields. The method used is based on some ideas that have been discussed in the past ([3] and [9]–[11]) as well as on some novel techniques. The basic idea is to convert the computation of the DFT of a sequence of length n into the computation of the cyclic convolution of two appropriately defined sequences. The techniques for doing this are due to Bluestein [10] (also briefly discussed by Pollard [3] for application in certain special cases only) and to Rader [11]. The main feature of these techniques is that the length of the sequences to be convolved can be chosen as any integer $N \geqslant 2n - 1$. Since, by virtue of the well-known convolution theorem, convolutions can be implemented via Discrete Fourier Transforms over a field which we are now free to choose, N will be selected as a power of 2 for optimal FFT-implementation of the transforms.

The latter transforms are computed in the complex field. In order to apply the complex field DFT to elements of a finite field GF(p^m), we resort to the standard representation of the latter as polynomials with integer coefficients and of degree less than m. The convolution of sequences over GF(p^m) could then be recovered by combining the results of m^2 convolutions of integer sequences; however, by observing that the latter combination can be formulated as a cyclic convolution, the entire operation is implemented via a two-dimensional FFT. Finally, by noting that Bluestein's technique, i.e. the conversion of the DFT to a cyclic convolution, involves a pre-conditioning and a post-conditioning consisting of multiplications in GF(p^m), a considerable computational saving is obtained by combining preconditioning, convolution, and post-conditioning into a single procedure. In this procedure, which involves computations in several transform domains, those three conceptual steps essentially blur their confines. The result is that the DFT of sequences of n elements over GF(p^m) can be computed with $O(nm \cdot \log(nm) \cdot P(q))$ bit operations, where $P(q)$ is the number of bit operations required to multiply two q-bit integers and $q \cong 2 \log_2 n + 4 \log_2 m + 4 \log_2 p$.

The method proposed in this paper basically resorts to the complex field FFT for computing integer convolutions and departs from the prevalent trend. In fact, in recent years several authors [3]–[8] have suggested that the complex field FFT not be used for computing integer convolutions because of the problems of round-off error. The alternative is, of course, the finite field FFT based on a careful choice of the finite field. We have not followed the latter approach for the following reasons: (i) we are interested in techniques which are uniformly applicable to all cases, whereas the success of the FFT over a finite field depends upon the existence of primes of a special type [13] and, as is easily shown, no higher efficiency is achieved, even for the case of prime fields; (ii) the round-off error problem is entirely controllable, since real numbers can be economically represented so that in all cases the results will be correctly interpreted.

II. Discrete Fourier Transforms via Convolutions. Let p denote a prime,

$\mathrm{GF}(p^m)$ the finite field containing p^m elements,

n a divisor of $p^m - 1$,

α a primitive nth root of unity in $\mathrm{GF}(p^m)$.

In what follows we shall also make use of a short-hand notation for vectors: $(a_i)_k^l$ with $k < l$ denotes the vector $\langle a_k, a_{k+1}, \ldots, a_l\rangle$ and $(0)^t$ denotes the vector $\langle 0, 0, \ldots, 0\rangle$ whose t components are all equal to 0; also uv denotes the concatenation of two vectors u and v.

Let $a_i \in \mathrm{GF}(p^m)$ for $i = 0, 1, \ldots, n-1$. The Discrete Fourier Transform of $(a_i)_0^{n-1}$ is defined to be the sequence $(A_i)_0^{n-1}$ where

$$A_j = \sum_{i=0}^{n-1} a_i\alpha^{ij}, \qquad j \in [0, n-1]. \tag{1}$$

The inverse Fourier Transform to (1) is given by

$$a_i = (n)^{-1} \sum_{j=0}^{n-1} A_j\alpha^{-ij}, \qquad i \in [0, n-1], \tag{2}$$

where $(n)^{-1}$ is the multiplicative inverse in $\mathrm{GF}(p^m)$ of the field integer n. As is well known, the direct method of computing the DFT requires $O(n^2)$ arithmetic operations in $\mathrm{GF}(p^m)$. If n is composite, with $n = n_1 \cdot n_2 \cdot \ldots \cdot n_l$, then the finite-field analogue of the FFT algorithm over the complex field can be used to compute the DFT [3]. This algorithm requires $O(n(n_1 + n_2 + \cdots + n_l))$ arithmetic operations in $\mathrm{GF}(p^m)$. Thus for highly composite n, the DFT can be computed in $O(n \log n)$ arithmetic operations in $\mathrm{GF}(p^m)$.

When n is not highly composite, or is a prime, the saving in computation achieved by resorting to the FFT can be small (or nonexistent). In such cases the DFT problem is reformulated as a convolution problem, to the solution of which more general methods can be applied. There are two main techniques for such reformulation, which we now briefly review for the reader's benefit.

The first technique is due to Bluestein [10], and is based on the following expression for A_j:

$$\begin{aligned} A_j &= \sum_{i=0}^{n-1} a_i\alpha^{ij} = \sum_{i=0}^{n-1} a_i\alpha^{\frac{1}{2}[i^2+j^2-(j-i)^2]} \\ &= \beta^{j^2} \sum_{i=0}^{n-1} (a_i\beta^{i^2})(\beta^{-(j-i)^2}), \quad \text{where } \beta = \sqrt{\alpha}. \end{aligned} \tag{3}$$

The sum in (3) can be recognized to be a term of the convolution of the sequence $(a_i\beta^{i^2})_0^{n-1}$ with the sequence $(\beta^{-i^2})_{1-n}^{n-1}$. It is easy to see that the convolution of these sequences can be obtained from the cyclic (or periodic) convolution of the sequences $(a_i\beta^{i^2})_0^{n-1}(0)^{N-n}$ and $(\beta^{-i^2})_{1-n}^{n-1}(0)^{N-(2n-1)}$, of common length N, where N must be no smaller than $2n-1$, and, in view of optimally executing the FFT, is conveniently chosen as a power of 2. Thus, $N = 2^s$, where $s \geqslant \lceil \log_2(2n-1)\rceil$.

Notice, however, since β is not necessarily an element of $\mathrm{GF}(p^m)$, the two length-

N sequences to be convolved are also not always over $\mathrm{GF}(p^m)$. Indeed, they are over $\mathrm{GF}(p^m)$ in the following two cases: (i) $p = 2$, since, in this case, $\beta = \sqrt{\alpha} = \alpha^{\frac{1}{2}(n+1)} \in \mathrm{GF}(2^m)$, because $(n + 1)/2$ is an integer; (ii) $p \neq 2$ and $2n|(p^m - 1)$, since β is primitive of order $2n$ in $\mathrm{GF}(p^m)$. Otherwise, when $p \neq 2$ and $2n \nmid (p^m - 1)$, β is a primitive $2n$th root of unity in $\mathrm{GF}(p^{2m})$ and we have a convolution of sequences over $\mathrm{GF}(p^{2m})$. In this case we carry out the computation in this larger field. However, the result of multiplying by β^{j^2} the corresponding term of the convolution is equal to A_j, which, by (1), is an element of $\mathrm{GF}(p^m)$. Thus, in the sequel we shall simply refer to convolutions of sequences over $\mathrm{GF}(p^m)$, with the understanding that whenever this field must be replaced by $\mathrm{GF}(p^{2m})$, the complexity analysis is correspondingly modified.

A second technique to compute DFTs via convolution is due to Rader [11]. This technique is unfortunately restricted to the case in which n is a prime, but in some instances it is slightly more efficient than the more general Bluestein's technique, as illustrated below. Let the integer θ be a primitive root of unity modulo n and let $\phi(i)$ be the smallest integer such that $\theta^{\phi(i)} \bmod n = i$ for $i \in [1, n - 1]$. Then

$$A_j - a_0 = \sum_{i=1}^{n-1} a_i \alpha^{ij},$$

i.e.

$$A_{(\theta^{\phi(j)} \bmod n)} - a_0 = \sum_{i=1}^{n-1} a_{(\theta^{\phi(i)} \bmod n)} \alpha^{\theta^{\phi(i)+\phi(j)}}.$$

Since $(\phi(i))_1^{n-1}$ is simply a permutation of $(i)_1^{n-1}$, we can set $l = \phi(j)$ and use $k = \phi(i)$ as the index of summation to get

$$A_{(\theta^l \bmod n)} - a_0 = \sum_{k=1}^{n-1} a_{(\theta^k \bmod n)} \alpha^{\theta^{k+l}}.$$

This shows that the sequence $(A_{(\theta^l \bmod n)} - a_0)_1^{n-1}$ is the cyclic correlation of the sequences $(a_{(\theta^k \bmod n)})_1^{n-1}$ and $(\alpha^{\theta^k})_1^{n-1}$ of length $n - 1$. We obtain a cyclic convolution by reversing one of the sequences. The advantage of Rader's method is that both sequences are always over $\mathrm{GF}(p^m)$ and are shorter. In particular, if the prime n happens to be of the form $2^k + 1$, Rader's method gives a convolution of sequences of length 2^k whereas Bluestein's method would give a convolution of sequences of length 2^{k+2}. In this paper we shall not further consider Rader's method due to its limited applicability, and marginal superiority when applicable.

The preceding discussion indicates that to compute the DFT of a sequence $(a_i)_0^{n-1}$ over $\mathrm{GF}(p^m)$ using (3) we must

(i) construct the sequences $(\beta^{i^2})_0^{n-1}$, $(\beta^{-i^2})_{1-n}^{n-1}$, and $(a_i\beta^{i^2})_0^{n-1}$, and extend the latter two to length N with appropriate strings of zeros;

(ii) compute the cyclic convolution of the latter two sequences;

(iii) multiply by β^{j^2}, $j \in [0, n - 1]$, the corresponding term of the convolution.

In the next section we shall show how the preceding three steps, which are separately stated for conceptual convenience, can be integrated into a single efficient computational procedure.

III. An Integrated Procedure For DFT Implementation. The objective, embodied by Bluestein's and Rader's techniques, of reformulating DFTs as cyclic convolutions of sequences, is that the common lengths of the latter can be chosen so that the FFT technique is always optimally applicable. In fact, as is very well known, efficient computation of convolutions is based on the convolution theorem [3], which states: Let $(w_i)_0^{N-1}$ denote the result of cyclically convolving $(x_i)_0^{N-1}$ and $(y_i)_0^{N-1}$, i.e.

$$w_j = \sum_{i=0}^{N-1} x_i y_{((j-i))} \quad \text{for } j \in [0, N-1],$$

where $((j-i))$ denotes $(j-i)$ mod N. If $(W_i)_0^{N-1}$, $(X_i)_0^{N-1}$, and $(Y_i)_0^{N-1}$ denote their respective N-point Fourier Transforms, then

$$W_j = X_j Y_j \quad \text{for } j \in [0, N-1]. \tag{4}$$

Suppose at first that GF(p^m) contains an Nth root of unity. We can find $(X_i)_0^{N-1}$ and $(Y_i)_0^{N-1}$ with $O(N \log N)$ arithmetic operations, find the W_j's from (4) with N multiplications and then find the w_j's with $O(N \log N)$ arithmetic operations, all in GF(p^m). This implies that, when GF(p^m) contains an Nth root of unity, the Fourier Transform of a sequence of length n can be found in $O(n \log n)$ arithmetic operations in GF(p^m). However, recall that, by the original formulation of the DFT problem, GF(p^m) is assumed to contain an nth root of unity. Thus, we must have both $n|(p^m - 1)$ and $N|(p^m - 1)$. Now, notice that N, a power of 2, is highly composite, while n is presumably not so; therefore, if n and N have a very small G.C.D., then we may reasonably conclude that n is $O(\sqrt{p^m})$ or smaller; thus the event that GF(p^m) contains an Nth root of unity is likely to be quite rare.

On the other hand, it has been suggested in a similar context [3] that since an Nth root unity will exist in some extension field of GF(p^m),*** the DFT can be found in $O(n \log n)$ arithmetic operations in this extension field. This approach can be quite deceptive, however, because the extension field could be very large and the complexity of the arithmetic could far outweigh the reduction of the number of operations from $O(n^2)$ to $O(n \log n)$, as the following examples indicate.

Example 1. Take $p = 3$, $m = 3$, $n = 26$. Since $2n \nmid (p^m - 1)$, we have to compute a convolution of sequences of length 2^6 over GF(3^6). Now, a 64th root of unity exists in GF(3^{16}); however, the smallest extension field of GF(3^6) containing a 64th root of unity is GF(3^{48}).

Example 2. Take $p = 2$, $m = 5$, $n = 31$. Here we have to compute a convolution of sequences of length 3^4.*** An 81st root of unity exists in GF(2^{54}) and the smallest extension field of GF(2^5) containing an 81st root is GF(2^{270}).

Actually, one can use any highly composite integer N', satisfying $N' \geqslant 2n - 1$ and $p \nmid N'$, as the length of the sequences to be convolved; and try to find a (N')th

***A word of caution is in order. No finite field of characteristic 2 can contain an Nth root of unity. To avoid this difficulty, when $p = 2$ one could choose $N = 3^s \geqslant 2n - 1$, still permitting an efficient implementation of the FFT.

root of unity in an extension field of $\mathrm{GF}(p^m)$. Thus, we have

Example 2 (continued). Choose $N' = 63 \geqslant 61$. A 63rd root of unity exists in $\mathrm{GF}(2^6)$, and the smallest extension field of $\mathrm{GF}(2^5)$ containing a 63rd root is $\mathrm{GF}(2^{30})$.

The preceding discussion indicates that computation of the convolution in $\mathrm{GF}(p^m)$ or its extensions is not viable in general. What is desirable instead is a method universally applicable to all instances of p and m and of easily determinable complexity, although for specific isolated choices of p and m other approaches may be slightly more efficient.

To this end we propose to transform the problem of convolving sequences over $\mathrm{GF}(p^m)$ into the problem of convolving arrays of integers, i.e. natural numbers. This approach is not novel: in fact it was proposed by Pollard [3, Section 8] in connection with the use of Bluestein's method for the very particular case $m = 1$ and $2n|(p-1)$, i.e., for a prime field containing a $2n$th root of unity. Both restrictions, however, are unnecessary. In fact, we have shown in Section II that if the field does not contain a $2n$th root of unity, we merely need to compute the convolution in a somewhat larger field. The condition $m = 1$ was imposed so that the sequences could be treated as sequences of integers in the range $[0, p-1]$ and convolved by the methods of [3, Section 3]. We now show that this restriction can be easily removed.

First of all, we review the standard representation of elements of $\mathrm{GF}(p^m)$ as polynomials. Let ζ be a root of a monic irreducible polynomial $g(z)$ of degree m over $\mathrm{GF}(p)$. Then every element $x_i \in \mathrm{GF}(p^m)$ can be represented as a polynomial $x_i(\zeta)$ of degree less than m in ζ, i.e.

$$x_i(\zeta) = \sum_{k=0}^{m-1} x_{i,k}\zeta^k, \qquad x_{i,k} \in \mathrm{GF}(p). \tag{5}$$

As is well known, addition in $\mathrm{GF}(p^m)$ corresponds to addition of polynomials over $\mathrm{GF}(p)$, and multiplication in $\mathrm{GF}(p^m)$ corresponds to multiplication of polynomials over $\mathrm{GF}(p)$ modulo $g(\zeta)$.

As before, let $(w_i)_0^{N-1}$ denote the result of the cyclic convolution of $(x_i)_0^{N-1}$ and $(y_i)_0^{N-1}$. Then we have

$$w_j = \sum_{i=0}^{N-1} x_i y_{((j-i))}$$

and, using the representation (5) of elements of $\mathrm{GF}(p^m)$ as polynomials, we have

$$\begin{aligned} w_j(\zeta) &= \sum_{i=0}^{N-1} (x_i(\zeta) y_{((j-i))}(\zeta)) \mod g(\zeta) \\ &= \sum_{i=0}^{N-1}\left(\sum_{t=0}^{2m-2}\left(\sum_{k=0}^{t} x_{i,k} y_{((j-i)),t-k}\right)\zeta^t\right) \mod g(\zeta), \end{aligned}$$

where $x_{i,k}$ and $y_{((j-i)),t-k}$ are taken to be zero if the second subscript exceeds $m-1$. Thus,

$$w_j(\zeta) = \left[\sum_{t=0}^{2m-2}\left(\sum_{k=0}^{t}\sum_{i=0}^{N-1} x_{i,k}y_{((j-i)),t-k}\right)\zeta^t\right] \mod g(\zeta)$$

$$= \left(\sum_{t=0}^{2m-2} z_{j,t}\zeta^t\right) \mod g(\zeta),$$

where

$$z_{j,t} \overset{\Delta}{=} \sum_{k=0}^{t}\sum_{i=0}^{N-1} x_{i,k}y_{((j-i)),t-k} \in \mathrm{GF}(p).$$

Now suppose we treat all the terms of $(x_{i,k})_0^{N-1}$ and $(y_{i,k})_0^{N-1}$ as elements of $\mathbf{Z}$, the set of the integers, rather than of GF(p). With this stipulation, we define the integers

$$\bar{z}_{j,t} = \sum_{k=0}^{t}\left(\sum_{i=0}^{N-1} x_{i,k}y_{((j-i)),t-k}\right) \quad \text{for } j \in [0, N-1] \text{ and } t \in [0, 2m-2] \tag{6}$$

and, obviously, $\bar{z}_{j,t} \bmod p = z_{j,t}$. The right side of (6) has the form of a double, or two-dimensional, convolution which is a periodic convolution in one dimension and an aperiodic convolution in the other. The latter can be easily converted into a periodic convolution by extending the range of k and t to $[0, M-1]$ where $M \geqslant 2m-1$. Letting $[[t-k]]$ denote $(t-k)$ mod M, (6) can be rewritten as

$$\bar{z}_{j,t} = \sum_{k=0}^{M-1}\sum_{i=0}^{N-1} x_{i,k}y_{((j-i)),[[t-k]]}. \tag{7}$$

It is convenient to think of the set $\{x_{i,k}\}$, $0 \leqslant i \leqslant N-1$, $0 \leqslant k \leqslant M-1$ as a two-dimensional $N \times M$ array $[x_{i,k}]$, and of (7) as an array convolution. Let $[X_{j,l}]$ denote the two-dimensional Fourier Transform of $[x_{i,k}]$ over the complex field, i.e.,

$$X_{j,l} = \sum_{k=0}^{M-1}\sum_{i=0}^{N-1} x_{i,k}\Omega_N^{ij}\Omega_M^{kl}, \tag{8}$$

where $\Omega_N = e^{j2\pi/N}$ and $\Omega_M = e^{j2\pi/M}$ are primitive roots of unity of order N and M, respectively, in the complex field. If $[\bar{Z}_{j,l}]$ and $[Y_{j,l}]$ denote the transforms of the arrays $[\bar{z}_{i,k}]$ and $[y_{i,k}]$, respectively, then we have

$$\bar{Z}_{j,t} = X_{j,t}Y_{j,t} \quad \text{for } j \in [0, N-1] \text{ and } t \in [0, M-1]. \tag{9}$$

Recall from Section II that one of the two sequences to be convolved is $(a_i\beta^{i^2})_0^{n-1}(0)^{N-n}$, i.e. it is the term-by-term product of the two sequences $(a_i)_0^{n-1}(0)^{N-n}$ and $(\beta^{i^2})_0^{n-1}(0)^{N-n}$. Representing the field elements as polynomials as in (5), the element $a_i\beta^{i^2}$ is a polynomial of degree $(2m-2)$ before it is reduced mod $g(\zeta)$. If we neglect to do this reduction, the convolution in (3) will give a polynomial of degree $(3m-3)$ and the post-convolution multiplication by β^{j^2} will give a polynomial of degree $(4m-4)$. Reducing this last polynomial mod $g(\zeta)$ produces the same result as a reduction mod $g(\zeta)$ after each multiplication. With this in mind, we choose $M = 2^r$ where $r = \lceil\log_2(4m-3)\rceil$ and define the $N \times M$ arrays $[a_{i,k}]$, $[b_{i,k}]$, and $[y_{i,k}]$ corresponding to the sequences $(a_i)_0^{n-1}(0)^{N-n}$, $(\beta^{i^2})_0^{n-1}(0)^{N-n}$ and

$(\beta^{-i^2})_{1-n}^{n-1}(0)^{N-2n+1}$. Let us also define the row-Fourier Transform (RFT) of the array $[x_{i,k}]$ as the array $[x'_{i,l}]$ where

$$x'_{i,l} = \sum_{k=0}^{M-1} x_{i,k}\Omega_M^{kl} \tag{10}$$

and the column-Fourier Transform (CFT) as the array $[x''_{j,k}]$ where

$$x''_{j,k} = \sum_{i=0}^{N-1} x_{i,k}\Omega_N^{ij}. \tag{11}$$

Using the notation $[x'_{i,l}] = \mathrm{RFT}[x_{i,k}]$ and $[x''_{j,k}] = \mathrm{CFT}[x_{i,k}]$, we have

$$\mathrm{CFT}[\mathrm{RFT}[x_{i,k}]] = \mathrm{RFT}[\mathrm{CFT}[x_{i,k}]] = \mathrm{CRFT}[x_{i,k}] = [X_{j,l}],$$

where $X_{j,l}$ was defined in (8).

We thus have the following algorithm for the computation of the discrete Fourier Transform of a sequence of n elements of $\mathrm{GF}(p^m)$.

Algorithm.

Step 1. $[a'_{i,l}] \longleftarrow \mathrm{RFT}[a_{i,k}]$, $[b'_{i,l}] \longleftarrow \mathrm{RFT}[b_{i,k}]$.

Step 2. $[x'_{i,l}] \longleftarrow [a'_{i,l} \cdot b'_{i,l}]$.

(*Comment*: The rows of the array $[x'_{i,l}]$ are the transform domain representations of the polynomials $a_i\beta^{i^2}$.)

Step 3. $[X_{j,l}] \longleftarrow \mathrm{CFT}[x'_{i,l}]$.

Step 4. $[Y_{j,l}] \longleftarrow \mathrm{CRFT}[y_{i,k}]$.

Step 5. $[\overline{Z}_{j,l}] \longleftarrow [X_{j,l} \cdot Y_{j,l}]$.

(*Comment*: $[\overline{Z}_{j,l}]$ is the transform domain representation of the convolution in (3).)

Step 6. $[\overline{z}'_{i,l}] \longleftarrow \mathrm{CFT}^{-1}[\overline{Z}_{j,l}]$.

(*Comment*: The rows of the array $[\overline{z}'_{i,l}]$ are the transform domain representations of the polynomials defined by $\Sigma_{j=0}^{n-1}(a_j\beta^{j^2})\beta^{-(i-j)^2}$.)

Step 7. $[\overline{u}'_{i,l}] \longleftarrow [\overline{z}'_{i,l} \cdot b'_{i,l}]$.

Step 8. $[\overline{u}_{i,k}] \longleftarrow \mathrm{RFT}^{-1}[\overline{u}'_{i,l}]$.

(*Comment*: The array $[\overline{u}_{i,k}]$ represents the desired Fourier Transform as polynomials of degree $4m - 4$ with integer coefficients.)

Step 9. $[u_{i,k}] \longleftarrow [\overline{u}_{i,k} \bmod p]$.

Step 10. $A_i \longleftarrow (\Sigma_{k=0}^{4m-4} u_{i,k}\zeta^k) \bmod g(\zeta)$ for $i = 0, 1, \ldots, n - 1$.

We note that, in certain special cases, our algorithm is not necessarily the best. For example, if one of the convolutions in (6) is quite short, then it can be computed more efficiently using the methods in [15]. However, our algorithm does have the virtue of being uniformly applicable to all cases of computation of the DFT over finite fields, and of having order of complexity not inferior to other algorithms.

IV. Performance Evaluation of the Algorithm. The following analysis is based on the so-called "logarithmic model" [4, Chapter 1], that is, the time required by the algorithms will be estimated in terms of "bit operations" rather than of "operand operations" (as is the case with the so-called "uniform model"). This choice is justified on the grounds that the length of the operands used by the algorithms depends upon the parameter n, the original sequence length. Thus, assuming a fixed operand size would be erroneous; on the other hand, the running time on any given conventional machine of the random access memory type will be proportional to the bit complexity to be evaluated.

A basic operation used by the convolution algorithm described in the preceding section is the Fourier Transform of a sequence of integers in the complex field. The choice of this field is obviously motivated by the fact that a primitive root of unity exists for every order N; thus N can be chosen as the one which yields an optimal implementation of the FFT algorithm. Resorting to the complex-field FFT is by no means novel; it was proposed in the past in connection with similar problems, notably by Schönhage and Strassen [9] as a device for fast integer multiplication. The main problem encountered with this approach is that the accuracy with which complex numbers are represented must be sufficient to guarantee that the rounded-off results exactly yield the correct integers. We now estimate in detail the number of bits that must be used to represent the powers of the primitive roots of unity Ω_M and Ω_N. A similar analysis, applied however to a considerably simpler situation, can be found in [9].

Let $N_0 = 2^t$. As is well known, the ith stage ($i = 0, 1, \ldots, t-1$) in the execution of the FFT consists of a set of "butterfly operations", in each of which two complex numbers A_i and B_i are combined to yield two outputs A_{i+1} and B_{i+1} where

$$A_{i+1} = A_i + \Omega B_i, \qquad B_{i+1} = A_i - \Omega B_i, \tag{12}$$

where Ω is some power of $\Omega_{N_0} = e^{j2\pi/N_0}$. Let η be the maximum error in any power of Ω_{N_0}, let ϵ_i be the maximum error in the inputs to the ith stage and let L_i be the maximum modulus of the inputs to the ith stage. From (12) we have

$$L_{i+1} \leqslant 2L_i \leqslant 2^{i+1}L_0$$

and

$$\epsilon_{i+1} \leqslant \epsilon_i + B_i\eta + \Omega\epsilon_i \leqslant L_i\eta + 2\epsilon_i \leqslant 2^iL_0\eta + 2\epsilon_i$$

since $|B_i| \leqslant L_i$ and $|\Omega| = 1$. Therefore, we have

$$\epsilon_{i+1} \leqslant (i+1)2^iL_0\eta + 2^{i+1}\epsilon_0;$$

and

$$\epsilon_t \leqslant t \cdot 2^{t-1}L_0\eta + 2^t\epsilon_0 = \tfrac{1}{2}tN_0L_0\eta + N_0\epsilon_0$$

is the bound on the error of the Fourier Transform.

For the inverse Fourier Transform, the equations (12) are slightly modified. Using $\hat{A}_i$, $\hat{B}_i$, $\hat{L}_i$, and $\hat{\epsilon}_i$ to denote the corresponding quantities for the inverse transform, we have

$$\begin{cases} \hat{A}_{i+1} = \tfrac{1}{2}(\hat{A}_i + \Omega\hat{B}_i), \\ \hat{B}_{i+1} = \tfrac{1}{2}(\hat{A}_i - \Omega\hat{B}_i). \end{cases}$$

Therefore, we can choose $\hat{L}_0 = \hat{L}_1 = \cdots = \hat{L}_t$, whence $\hat{\epsilon}_{i+1} \leqslant \frac{1}{2}[\hat{\epsilon}_i + \hat{B}_i\eta + \Omega\hat{\epsilon}_i] \leqslant \hat{\epsilon}_i + \frac{1}{2}\hat{L}_0\eta$.[†] Hence, we have $\hat{\epsilon}_t \leqslant \hat{\epsilon}_0 + \frac{1}{2}t\hat{L}_0\eta$.

From this, it is easy to determine the maximum moduli of and maximum errors in the results produced at each of Steps 1–8 of the algorithm. These are given in Table 1.

TABLE 1

Maximum moduli of and errors in complex numbers produced in Algorithm

Step	Maximum modulus of result	Maximum error in result
1	Mp	$\frac{1}{2}rMp\eta$
2	M^2p^2	$rM^2p^2\eta$
3	NM^2p^2	$(r + \frac{1}{2}s)NM^2p^2\eta$
4	NMp	$\frac{1}{2}(r + s)NMp\eta$
5	$N^2M^3p^3$	$(3r/2 + s)N^2M^3p^3\eta$
6	$N^2M^3p^3$	$(3/2)(r + s)N^2M^3p^3\eta$
7	$N^2M^4p^4$	$(2r + 3s)N^2M^4p^4\eta$
8	$N^2M^4p^4$	$\frac{1}{2}(5r + 3s)N^2M^4p^4\eta$

The results at Step 8 must have a maximum error of less than ½ in order that the results may be correctly interpreted as integers. It follows that the approximation error η in the roots of unity Ω must satisfy $\eta < 1/(5r + 3s)N^2M^4p^4$, or, equivalently, the number of bits used to represent the powers of the roots must be at least

$$q \triangleq \lceil 4 \log_2 p + \log_2(5r + 3s)\rceil + 2s + 4r.$$

We can now evaluate the performance of the Fourier Transform algorithm. Let $P(q)$ denote the number of bit operations required to multiply two q bit numbers. We represent the real and imaginary parts of both the powers of Ω and the terms A_i, B_i previously defined using q bits for each, and note that a complex field multiplication corresponds to four[††] multiplications of real numbers. Note that the integer parts of the results grow at each step of the algorithm, but concurrently their fractional parts lose accuracy so that the significant operand length remains constant. It follows that Steps 1 and 8 require $O(NM \log_2 M \cdot P(q))$ bit operations, that Steps 3 and 6 require $O(NM \log_2 N \cdot P(q))$ bit operations, that Step 4 requires $O(NM \log_2(NM) \cdot P(q))$ bit operations, and that Steps 2, 5 and 7 require $O(NM \cdot P(q))$ bit operations. Obviously, the complexity of Step 4 dominates those of Steps 1–8.

In Step 9, MN integers, represented by q bits each, are divided by p, which is represented with $\lceil \log_2 p \rceil < q$ bits. Thus, the complexity of this step is dominated by that of Steps 2 or 6 and hence by that of Step 4. Finally, in Step 10, N polynomials

[†] When the final maximum modulus $\hat{L}_t$ is known, we can derive the bound $\hat{\epsilon}_t \leqslant \hat{\epsilon}_0 + 2^t\hat{L}_t\eta$, which is based on the inequality $\hat{L}_i \leqslant 2^{t-i}\hat{L}_t$. However, this bound does not improve our result in any significant way.

[††] Or three real-number multiplications, using Buneman's method [14, p. 444].

over GF(p) of degree $4m - 4$ are divided by a polynomial of degree m. Let $D_p(m)$ denote the number of bit operations required to divide a polynomial of degree $2m$ by a polynomial of degree m in GF(p). Then, Step 10 requires at most $O(N \cdot D_p(2m))$ bit operations. If $m \ll N$, i.e., m is $O(\log N)$ or less, then a long division method can be used at Step 10 and the complexity of Step 10 is $O(Nm^2P(\log p))$ which is dominated by the complexity of Steps 3 and 6. On the other hand, if m is large compared to N, then fast polynomial division methods [4], [5] can be used at Step 10. The complexity is then $O(NM \log mP(\log m + \log p))$ bit operations, and this is dominated by the complexity of Steps 1 and 8. We thus have

THEOREM. *The Fourier Transform of a sequence of n elements of* GF(p^m), $n|(p^m - 1)$, *can be computed with* $O(nm \cdot \log(nm) \cdot P(q))$ *bit operations where* $P(q)$ *is the number of bit operations required to multiply two q-bit numbers and* $q \cong 2 \log_2 n + 4 \log_2 m + 4 \log_2 p$.

As a general observation, note that the number of arithmetic operations depends on the degree of the extension of GF(p) but not on the characteristic p itself. The latter affects the bit complexity of arithmetic operations only. The following special cases of the theorem are also of interest.

(i) If $m = 1$, the bit complexity reduces to $O(n \log nP(q))$ where $q \cong 2 \log n + 4 \log p < 6 \log p$, i.e., $P(q)$ is proportional to the complexity of arithmetic in GF(p). Thus, the Fourier Transform over GF(p) can be computed using $O(n \log n)$ arithmetic operations whose bit complexity is essentially that of arithmetic operations in GF(p).

(ii) If $p = 2$ and $n = 2^m - 1$, (which is a case of great interest in coding theory [1], [2]) then $P(q)$ is proportional to the complexity of arithmetic in GF(2^m). Thus, the Fourier Transform of a sequence of length $2^m - 1$ over GF(2^m) can be computed using $O(n \log^2 n)$ arithmetic operations whose bit complexity is essentially that of arithmetic operations in GF(2^m).

Remark. The algorithm proposed in this paper can also be used (with appropriate modifications) to compute the convolution of sequences over GF(p^m) of arbitrary length n. The asymptotic complexity is easily shown to be $O(n \log n)$ arithmetic operations whose bit complexity is essentially that of multiplying two ($\log n$) bit integers.

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

1. E. R. BERLEKAMP, *Algebraic Coding Theory,* McGraw-Hill, New York, 1968. MR 38 #6873.

2. F. J. MacWILLIAMS & N. J. A. SLOANE, *Theory of Error-Correcting Codes,* American Elsevier, New York. (To appear.)

3. J. M. POLLARD, "The fast Fourier transform in a finite field," *Math. Comp.,* v. 25, 1971, pp. 365–374. MR 46 #1120.

4. A. V. AHO, J. E. HOPCROFT & J. D. ULLMAN, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, Mass., 1974.

5. A. BORODIN & I. MUNRO, *The Computational Complexity of Algebraic and Numerical Problems,* American Elsevier, New York, 1975.

6. C. M. RADER, "Discrete convolutions via Mersenne transforms," *IEEE Trans. Computers,* v. C-21, No. 12, 1972, pp. 1269–1273.

7. R. C. AGARWAL & C. S. BURRUS, "Number theoretic transforms to implement fast digital convolution," *Proc. IEEE,* v. 63, No. 4, 1975, pp. 550–560.

8. I. S. REED & T. K. TRUONG, "The use of finite fields to compute convolutions," *IEEE Trans. Information Theory,* v. IT-21, No. 2, 1975, pp. 208–213.

9. A. SCHÖNHAGE & V. STRASSEN, "Schnelle Multiplikation grosser Zahlen," *Computing,* v. 7, 1971, pp. 281–292. MR **45** #1431.

10. L. I. BLUESTEIN, "A linear filtering approach to the computation of discrete Fourier transform," *IEEE Trans. Audio & Electroacoustics,* v. AU-18, 1970, pp. 451–455.

11. C. M. RADER, "Discrete Fourier transforms when the number of data samples is prime," *Proc. IEEE* (Letters), v. 56, No. 6, 1968, pp. 1107–1108.

12. J. W. COOLEY & J. W. TUKEY, "An algorithm for the machine calculation of complex Fourier series," *Math. Comp.,* v. 19, 1965, pp. 297–301. MR **31** #2843.

13. S. W. GOLOMB, "Properties of the sequence $3 \cdot 2^n + 1$," *Math. Comp.,* v. 30, 1976, pp. 657–663.

14. D. E. KNUTH, *The Art of Computer Programming.* Vol. 2: *Seminumerical Algorithms,* Addison-Wesley, Reading, Mass., 1971. MR **44** #3531.

15. R. C. AGARWAL & C. S. BURRUS, "Fast one-dimensional digital convolution by multidimensional techniques," *IEEE Trans. Acoustics, Speech and Signal Processing,* v. ASSP-22, No. 1, 1974, pp. 1–10.