On the Application of a Fast Polynomial Transform and the Chinese Remainder Theorem to Compute a Two-Dimensional Convolution

T. K. Truong and R. Lipes
Communications Systems Research Section

I. S. Reed
University of Southern California

C. Wu
Radar Science and Engineering Section

In this article, a fast algorithm is developed to compute two-dimensional convolutions of an array of $d_1 \cdot d_2$ complex number points, where $d_2 = 2^m$ and $d_1 = 2^{m-r+1}$ for some $1 \leq r \leq m$. This new algorithm requires fewer multiplications and about the same number of additions as the conventional FFT method for computing the two-dimensional convolution. It also has the advantage that the operation of transposing the matrix of data can be avoided.

I. Introduction

Two-dimensional convolutions of two sequences of complex number points can be applied to many areas, in particular to the synthetic aperture radar (SAR) (Refs. 1, 2). In SAR, a two-dimensional cross correlation of the raw echo data of complex numbers with the response function of a point target is required to produce images. When the two-dimensional filter does not change rapidly with the range, one can divide the entire range of echo data into several subintervals. Within each subinterval, one can use a constant filter function. This is accomplished usually by using the conventional fast Fourier transform (FFT). However, the FFT algorithm generally requires a large number of floating-point complex additions and multiplications. Also, the transpose of a matrix is usually required in the computation of such a two-dimensional convolution.

Recently, Rader (Ref. 3) proposed that a number-theoretic transform (NTT) could be used to accomplish two-dimensional filtering. It was shown (Ref. 4) that an improvement, both in accuracy and speed, of two-dimensional convolutions could be achieved by transforms over a finite field $GF(q)$, where q is a prime of the form $45 \times 2^{29} + 1$. However, to compute a two-dimensional convolution of two long sequences of an integer number of points, such a transform over a finite field did not allow for a wide variety of dynamic ranges.

More recently, Nussbaumer and Quandalle (Ref. 5) showed that a type of polynomial transform over the complex numbers could be used to efficiently compute two-dimensional convolutions. A principal advantage of this method over the above mentioned techniques is that the need for computing the transpose of matrix can be avoided. Furthermore, this new method offers a wider variety of dynamic ranges. It was shown recently by Arambepola and Rayner (Ref. 6) that the ideas of Nussbaumer and Quandalle can be generalized to a radix-2 polynomial transform analogous to the conventional radix-2 FET. But they do not make use of the Chinese remainder theorem as it is shown in this article to further reduce the complexity of the algorithm.

In this article, it is shown that a combination of a fast polynomial transform (FPT) and the Chinese remainder theorem (CRT) can be used to very efficiently compute a two-dimensional convolution of a $d_1 \cdot d_2$ complex number array, where $d_2 = 2^m$ and $d_1 = 2^{m-r+1}$ for $1 \leqslant r \leqslant m$. Such a new algorithm requires considerably fewer multiplications and about the same number of additions as the conventional algorithm for the two-dimensional case. Therefore, it has the potential for important application in SAR.

II. The Computation of Two-Dimensional Convolutions

The following algorithm for a two-dimensional digital convolution is based on an important identity. Let d_i be a power of 2 for $i = 1, 2$ and let C be the field of complex numbers. Also let a_{t_1,t_2} and b_{t_1,t_2} be two $d_1 \cdot d_2$ arrays, where $0 \leqslant t_i \leqslant d_i - 1$ for $i = 1, 2$. Then the two-dimensional cyclic convolution of a_{t_1,t_2} and b_{t_1,t_2}, where $a_{t_1,t_2}, b_{t_1,t_2} \in C$, is defined by

$$c_{n_1,n_2} = \sum_{t_1=0}^{d_1-1} \sum_{t_2=0}^{d_2-1} a_{t_1,t_2} b_{(n_1-t_1),(n_2-t_2)}, \quad 0 \leqslant n_i \leqslant d_i \text{ for } i = 1, 2 \tag{1}$$

where $(n_i - t_i)$ denotes the residue of $n_i - t_i$ modulo d_i for $i = 1, 2$.

Define

$$\left.\begin{aligned} A_{t_1}(Z) &= \sum_{t_2=0}^{d_2-1} a_{t_1,t_2} Z^{t_2} \quad B_{t_1}(Z) = \sum_{t_2=0}^{d_2-1} b_{t_1,t_2} Z^{t_2}; \\ C_{n_1}(Z) &= \sum_{n_2=0}^{d_2-1} c_{n_1,n_2} Z^{n_2}, \quad \text{for } 0 \leqslant n_1, t_1 \leqslant d_1 - 1 \\ &\qquad\qquad\qquad\qquad 0 \leqslant n_2, t_2 \leqslant d_2 - 1 \end{aligned}\right\} \tag{2}$$

Nussbaumer and Quandalle (Ref. 5) expressed the two-dimensional convolution in Eq. (1) as a one-dimensional convolution of polynomials, i.e.,

$$\left.\begin{aligned} C_{n_1}(Z) &= \sum_{t_1=0}^{d_1-1} \left(\sum_{t_2=0}^{d_2-1} a_{t_1,t_2} Z^{t_2} \right) B_{(n_1-t_1)}(Z) \\ &\equiv \sum_{t_1=0}^{d_1-1} A_{t_1}(Z) B_{(n_1-t_1)}(Z) \bmod \left(Z^{d_2} - 1\right) \quad \text{for } 0 \leqslant n_1 \leqslant d_1 - 1 \end{aligned}\right\} \tag{3}$$

where $C_{n_1}(Z), A_{t_1}(Z), B_{t_1}(Z)$ are defined in Eq. (2) and $(n_1 - t_1)$ denotes the residue of $n_1 - t_1$ modulo d_1.

If $d_2 = 2^m$ and $d_1 = 2^{m-r+1}$ for $1 \leqslant r \leqslant m$, then one can factor $Z^{d_2} - 1$ into pairwise relatively prime factors as follows,

$$Z^{d_2} - 1 = \left(Z^{d_2/2} + 1\right)\left(Z^{d_2/2^2} + 1\right)\cdots\left(Z^{d_2/2^{r-1}} + 1\right)\left(Z^{d_2/2^r} + 1\right)\left(Z^{d_2/2^r} - 1\right)$$

Thus, Eq. (3) is equivalent to

$$C_{n_1}(Z) = \sum_{t_2=0}^{d_1-1} A_{t_1}(Z)B_{(n_1-t_1)}(Z) \bmod \left(Z^{d_2/2} + 1\right)\left(Z^{d_2/2^2} + 1\right)\cdots\left(Z^{d_2/2^r} + 1\right)\left(Z^{d_2/2^r} - 1\right) \tag{4}$$

Since $(Z^{d_2/2} + 1), \cdots, (Z^{d_2/2^r} - 1)$ are pairwise relatively prime, by the Chinese remainder theorem (CRT) for polynomials (Ref. 7), the polynomial congruences

$$C^i_{n_1}(Z) \equiv C_{n_1}(Z) \bmod \left(Z^{d_2/2^i} + 1\right) \quad \text{for } i = 1, 2, \cdots, r \tag{5a}$$

and

$$C^*_{n_1}(Z) \equiv C_{n_1}(Z) \bmod \left(Z^{d_2/2^r} - 1\right) \tag{5b}$$

have a unique solution $C_{n_1}(Z)$ given by

$$C_{n_1}(Z) \equiv \sum_{i=1}^{r} C^i_{n_1}(Z)\left(-\frac{1}{2^i}\right)\frac{Z^{d_2} - 1}{Z^{d_2/2^i} + 1} + C^*_{n_1}(Z)\left(\frac{1}{2^r}\right)\frac{Z^{d_2} - 1}{Z^{d_2/2^r} - 1} \bmod Z^{d_2} - 1 \tag{6}$$

Note that the arithmetic needed to compute Eq. (6) requires only cyclic shifts and additions. The number of real additions needed to compute Eq. (6) is $2rd_2d_1$.

The derivation of $C^i_{n_1}(Z)$ in Eq. (5a) proceeds in the following manner:

$$\begin{aligned} C^i_{n_1}(Z) &\equiv C_{n_1}(Z) \bmod \left(Z^{d_2/2^i} + 1\right) \\ &\equiv \sum_{t_1=0}^{d_1-1} A_{t_1}(Z)B_{(n_1-t_1)}(Z) \bmod \left(Z^{d_2/2^i} + 1\right) \\ &\equiv \sum_{t_1=0}^{d_1-1} A^i_{t_1}(Z)B^i_{(n_1-t_1)}(Z) \bmod \left(Z^{d_2/2^i} + 1\right) \end{aligned} \tag{7}$$

where $A^i_{t_1}(Z)$ is defined by

$$A^i_{t_1}(Z) = \sum_{t_2=0}^{(d_2/2^i)-1} \left(a_{t_1,t_2} - a_{t_1,t_2+d_2/2^i} + a_{t_1,t_2+2d_2/2^i} \cdots\right.$$

$$\left. - a_{t_1,t_2+3d_2/2^i} + \cdots - a_{t_1,t_2(2^i-1)d_2/2^i}\right) Z^{t_2}$$

$$= \sum_{t_2=0}^{(d_2/2^i)-1} a^i_{t_1,t_2} Z^{t_2} \quad \text{for } 1 \leqslant i \leqslant r$$

and $B^i_{t_1}(Z)$ is obtained from the expression $A^i_{t_1}(Z)$ on replacing each $a_{i,j}$ by $b_{i,j}$, that is,

$$B^i_{t_1}(Z) = \sum_{t_2=0}^{(d_2/2^i)-1} b^i_{t_1,t_2} Z^{t_2} \quad \text{for } 1 \leqslant i \leqslant r$$

Note that $A^i_{t_1}(Z)$ and $B^i_{t_1}(Z)$ are, respectively, $A_{t_1}(Z)$ and $B_{t_1}(Z)$ reduced modulo $(Z^{d_2/2^i} + 1)$.

It was demonstrated in (Ref. 6) that a fast polynomial transform can be used to compute Eq. (7). Let us show this more carefully. Since $d_2 = 2^m$ and $d_1 = 2^{m-r+1}$ for $1 \leqslant r \leqslant m$, then $d_2/2^{i-1} = 2^{m-i+1} = 2^{r-i} \cdot 2^{m-r+1} = 2^{r-i} d_1$ for $i = 1, 2, \cdots, r$. Note the identity

$$Z^{d_2/2^{i-1}} \equiv \left(Z^{d_2/2^i}\right)^2 \equiv (-1)^2 \equiv 1 \bmod \left(Z^{d_2/2^i} + 1\right) \tag{8}$$

Also let $j = 2^{r-i}$ and define the d_1-point jth power polynomial transform of $A^i_{t_1}(Z)$ and $B^i_{t_2}(Z)$ by

$$\tilde{A}^i_k(Z) \equiv \sum_{t_1=0}^{d_1-1} A^i_{t_1}(Z)(\tilde{Z})^{t_1 k} \equiv \sum_{\ell=0}^{(d_2/2^i)-1} \tilde{a}^i_{k,\ell} Z^{\ell} \bmod \left(Z^{d_2/2^i} + 1\right) \quad \text{for } 0 \leqslant k \leqslant d_1 - 1 \tag{9a}$$

and

$$\tilde{B}^i_k(Z) \equiv \sum_{t_1=0}^{d_1-1} B^i_{t_1}(Z)(\tilde{Z})^{t_1 k} \equiv \sum_{\ell=0}^{(d_2/2^i)-1} \tilde{b}^i_{k,\ell} Z^{\ell} \bmod \left(Z^{d_2/2^i} + 1\right) \quad \text{for } 0 \leqslant k \leqslant d_1 - 1, \tag{9b}$$

respectively, where $\tilde{Z} = Z^j$ is the jth power of Z. Note by Eq. (8) that

$$\tilde{Z}^{d_1} \equiv Z^{jd_1} \equiv Z^{2^{r-i}d_1} \equiv Z^{d_2/2^{i-1}} \equiv 1 \bmod \left(Z^{d_2/2^i} + 1\right), \quad \text{for } i = 1, 2, \cdots, r$$

The product of the transforms $\widetilde{A}^i_k(Z)$ and $\widetilde{B}^i_k(Z)$ is given by

$$\widetilde{C}_k(Z) \equiv \widetilde{A}^i_k(Z) \cdot \widetilde{B}^i_k(Z) \equiv \sum_{\ell=0}^{(d_2/2^i)-1} \widetilde{c}^i_{k,\ell} Z^\ell \bmod \left(Z^{d_2/2^i}+1\right) \quad \text{for } 0 \leqslant k \leqslant d_1 - 1, \quad 1 \leqslant i \leqslant r \tag{10}$$

One needs now to compute the inverse transform of $\widetilde{C}^i_k(Z)$. That is, to compute

$$C^i_{n_1}(Z) \equiv \frac{1}{d_1} \sum_{k=0}^{d_1-1} \widetilde{C}^i_k(Z) \widetilde{Z}^{-kn_1}$$

$$\equiv d_1^{-1} \sum_{k=0}^{d_1-1} \left(\sum_{s=0}^{d_1-1} A^i_s(Z) Z^{sk}\right) \cdot \left(\sum_{r=0}^{d_1-1} B^i_r(Z) Z^{rk}\right) \widetilde{Z}^{-kn_1}$$

$$\equiv \sum_{s=0}^{d_1-1} \sum_{r=0}^{d_1-1} \widetilde{A}^i_s(Z) \widetilde{B}^i_r(Z) \frac{1}{d_1} \sum_{k=0}^{d_1-1} \widetilde{Z}^{(s+r-n_1)k} \bmod \left(Z^{d_2/2^i}+1\right) \tag{11}$$

Now let $t = s + r - n_1$ and note that

$$S \equiv \frac{1}{d_1} \sum_{k=0}^{d_1-1} \widetilde{Z}^{tk} \equiv \frac{1}{d_1} \frac{\widetilde{Z}^{td_1} - 1}{\widetilde{Z}^t - 1}$$

$$\equiv \frac{1}{d_1} \frac{\left(\widetilde{Z}^{d_1}\right)^t - 1}{\widetilde{Z}^t - 1} \equiv \frac{1}{d_1} \frac{\left(Z^{d_2/2^{i-1}}\right)^t - 1}{\widetilde{Z}^t - 1} \equiv 0 \bmod \left(Z^{d_2/2^i}+1\right) \quad \text{for } t \not\equiv 0 \bmod d_1$$

It is seen that $S = 1$ for $t \equiv 0 \bmod d_1$ and $S = 0$ for $t \not\equiv 0 \bmod d_1$. Hence Eq. (11) yields the desired result, namely,

$$C^i_{n_1}(Z) \equiv \sum_{t=0}^{d_1-1} A^i_{t_1}(Z) B^i_{(n_1-t_1)}(Z) \bmod \left(Z^{d_2/2^i}+1\right) \quad \text{for } 0 \leqslant n_1 \leqslant d_1 - 1, 1 \leqslant i \leqslant r \tag{12}$$

where $(n_1 - t_1)$ denotes the residue of $(n_1 - t_1)$ modulo d_1.

Since $d_2 = 2^m$ and $d_1 = 2^{m-r+1}$ for $1 \leqslant r \leqslant m$ the analogue of the conventional FFT algorithm can be utilized to realize the needed polynomial transforms Eqs. (9a) and (9b) and its inverse defined by Eq. (11). That is, in Eq. (9a) let the input data to the usual FFT be replaced by a sequence of polynomials over C, and let w, the usual d_1-*th* complex root of unity, be replaced by $\widetilde{Z}$, satisfying $\widetilde{Z}^{d_1} = 1$, and finally let the arithmetic operations on the complex number field be replaced by arithmetic operations in the field of polynomials. Then the polynomial transforms in Eqs. (9a), (9b), and (11) can be computed by the conventional FFT algorithm. The polynomial transform, obtained in this manner, is called the fast polynomial transform (FPT). The number of operations needed to perform the polynomial transform defined in Eq. (9a) requires only $(d_2/2) \cdot \log d_1$ cyclic shifts of polynomials and $2(d_2/2)\, d_1 \cdot \log d_1$ real additions.

To compute $\widetilde{C}_k^i(Z)$, given in Eq. (10) for $0 \leqslant k \leqslant d_1 - 1$, one needs to compute a polynomial product mod $Z^{d_2/2^i} + 1$. To do this directly takes a lengthy calculation. Using an idea, based on the work of Arambepola and Rayner (Ref. 6), this equation can be transformed into

$$\widetilde{C}_k^i(cu) = \widetilde{A}_k^i(cu) \cdot \widetilde{B}_k^i(cu) \bmod \left(u^{d_2/2^i} - 1\right) \tag{13}$$

by changing variables from Z to u with the mapping $Z = cu$, where c is *a* $d_2/2$-*th* primitive root of -1, i.e. $c^{d_2/2^i} = -1$. One can verify (Ref. 8) that the above polynomial product mod $u^{d_2/2^i} - 1$ can be obtained as a cyclic convolution of the two $d_2/2^i$-point coefficients of the polynomials $\widetilde{A}_k^i(cu)$ and $\widetilde{B}_k^i(cu)$. This is vastly simpler than finding the product modulo $Z^{d_2/2^i} + 1$.

In Eq. (13), one observes that

$$\begin{aligned} \widetilde{A}_k^i(cu) &= \sum_{\ell=0}^{(d_2/2^i)-1} \widetilde{a}_{k,\ell}^i (cu)^\ell = \sum_{\ell=0}^{(d_2/2^i)-1} \hat{a}_{k,\ell}^i u^\ell \\ \widetilde{B}_k^i(cu) &= \sum_{\ell=0}^{(d_2/2^i)-1} \widetilde{b}_{k,\ell}^i (cu)^\ell = \sum_{\ell=0}^{(d_2/2^i)-1} \hat{b}_{k,\ell}^i u^\ell \end{aligned} \tag{14}$$

and

$$\widetilde{C}_k^i(cu) = \sum_{\ell=0}^{(d_2/2^i)-1} \widetilde{c}_{k,\ell}^i (cu)^\ell = \sum_{\ell=0}^{(d_2/2^i)-1} \hat{c}_{k,\ell}^i u^\ell$$

where $\hat{a}_{k,\ell}^i = c^\ell \widetilde{a}_{k,\ell}^i$, $\hat{b}_{k,\ell}^i = c^\ell \widetilde{b}_{k,\ell}^i$, and $\hat{c}_{k,\ell}^i = c^\ell \widetilde{c}_{k,\ell}^i$ for $0 \leqslant \ell \leqslant (d_2/2^i) - 1$. If one defines

$$\begin{aligned} \hat{A}_k^i(u) &= \sum_{\ell=0}^{(d_2/2^i)-1} \hat{a}_{k,\ell}^i u^\ell, \quad \hat{B}_k^i(u) = \sum_{\ell=0}^{(d_2/2^i)-1} \hat{b}_{k,\ell}^i u^\ell; \\ \hat{C}_k^i(u) &= \sum_{\ell=0}^{(d_2/2^i)-1} \hat{c}_{k,\ell}^i u^\ell \end{aligned} \tag{15}$$

Then Eq. (13) becomes

$$\hat{C}_k^i(u) = \hat{A}_k^i(u)\, \hat{B}_k^i(u) \bmod \left(u^{d_2/2^i} - 1\right) \tag{16}$$

The polynomial multiplication in Eq. (16) can be computed by the cyclic convolution of the two $d_2/2^i$-point sequences, $(\hat{a}^i_{k,0}, \hat{a}^i_{k,1}, \cdots, \hat{a}^i_{k,d_2/2^i-1})$ and $(\hat{b}^i_{k,0}, \hat{b}^i_{k,1}, \cdots, \hat{b}^i_{k,d_2/2^i-1})$. This cyclic convolution can be written as

$$\hat{c}^i_{k,\ell} = \sum_{n=0}^{(d_2/2^i)-1} \hat{a}^i_{k,n}\, \hat{b}^i_{k,(\ell-n)} \quad \text{for } 0 \leqslant k \leqslant d_1 - 1 \quad 0 \leqslant \ell \leqslant d_2/2^i - 1 \tag{17}$$

where $(\ell - n)$ denotes the residue of $(\ell - n)$ modulo $d_2/2^i$ and $\hat{a}^i_{n,k}$ and $\hat{b}^i_{n,k}$ are the coefficients of the $\hat{A}^i_k(Z)$ and $\hat{B}^i_k(Z)$, respectively. The convolution $\hat{c}^i_{k,\ell}$ in Eq, (17) can be computed by using the conventional FFT (Ref. 9). Thus, the desired quantities in Eq. (15), i.e., $\tilde{c}^i_{k,\ell}$ for $0 \leqslant k \leqslant d_1 - 1$, $0 \leqslant \ell \leqslant d_2/2^i - 1$ are obtained from $\tilde{c}_{k,\ell} = c^{-\ell}\hat{c}^i_{k,\ell}$. The number of real multiplications and real additions needed to compute Eq. (5a) are $d_1(d_2/2^{i-2}) \cdot \log(d_2/2^i) + d_1d_2/2^{i-2} + 8d_1(d_2/2^i - 1)$ and $2d_1d_2 + (d_2/2^{i-2})d_1 \cdot \log d_1 + 6d_1(d_2/2^i) \cdot \log(d_2/2^i)$, for $i = 1, 2, \cdots, r$, respectively.

The flow chart for computing $C^i_n(Z)$ is given in Fig. 1.

Consider now the computation of $C^*_{n_1}(Z)$ in Eq. (5b). That is,

$$C^*_{n_1}(Z) = \sum_{t_1=0}^{d_1-1} A^*_{t_1}(Z)\, B^*_{(n_1-t_1)}(Z) \bmod \left(Z^{d_2/2^r} - 1\right) \tag{18}$$

where

$$A^*_{t_1}(Z) = \sum_{t_2=0}^{(d_2/2^r)-1} \Big(a_{t_1,t_2} + a_{t_1,t_2+d_2/2^r} + a_{t_1,t_2+2d_2/2^r}$$

$$+ a_{t_1,t_2+3d_2/2^r} + \ldots + a_{t_1,t_2+(2^r-1)d_2/2^r}\Big) Z^{t_2}$$

$$= \sum_{t_2=0}^{(d_2/2^r)-1} a^*_{t_1,t_2} Z^{t_2}$$

and $B^*_{t_1}(Z)$ is obtained from the expression $A^*_{t_1}(Z)$ on replacing each $a_{i,j}$ by $b_{i,j}$, that is,

$$B^*_{t_1}(Z) = \sum_{t_2=0}^{(d_2/2^r)-1} b^*_{t_1,t_2} Z^{t_2}$$

Evidently, $C^*_{n_1}(Z)$ in Eq. (18) is in the form of

$$C^*_{n_1}(Z) = \sum_{n_2=0}^{(d_2/2^r)-1} c^*_{n_1,n_2} Z^{t_2}$$

Note here that $A^*_{t_1}(Z)$ and $B^*_{t_1}(Z)$ are, respectively, $A_{t_1}(Z)$ and $B_{t_1}(Z)$ reduced modulo $Z^{d_2/2^i} + 1$.

In order to use the FPT technique to obtain Eq. (18), again use the idea of Arambepola and Rayner (Ref. 6). To compute Eq. (18), let $Z = cu$, where c is a $d_2/2^r$-th root of -1. By an argument similar to that used to obtain Eq. (12), (18) becomes

$$C^*_{n_1}(cu) = \sum_{t_1=0}^{d_1-1} A^*_{t_1}(cu)\, B^*_{(n_1-t_1)}(cu) \bmod \left(u^{d_2/2^r} + 1\right) \quad \text{for } 0 \leqslant n_1 \leqslant d_1 - 1 \tag{19}$$

In Eq. (19), one observes that

$$\begin{aligned} A^*_{t_1}(cu) &= \sum_{t_2=0}^{(d_2/2^r)-1} a^*_{t_1,t_2}(cu)^{t_2} = \sum_{t_2=0}^{(d_2/2^r)-1} a'_{t_1,t_2}\, u^{t_2} \\ B^*_{t_1}(cu) &= \sum_{t_2=0}^{(d_2/2^r)-1} b^*_{t_1,t_2}(cu)^{t_2} = \sum_{t_2=0}^{(d_2/2^r)-1} b'_{t_1,t_2}\, u^{t_2} \end{aligned} \tag{20}$$

and

$$C^*_{n_1}(cu) = \sum_{n_2=0}^{(d_2/2^r)-1} c^*_{n_1,n_2}(cu)^{n_2} = \sum_{n_2=0}^{(d_2/2^r)-1} c'_{n_1,n_2}\, u^{n_2}$$

where $c'_{n_1,n_2} = c^{n_2} c^*_{n_1,n_2}$, $a'_{t_1,t_2} = c^{t_2} a^*_{t_1,t_2}$, and $b'_{t_1,t_2} = c^{t_2} b^*_{t_1,t_2}$ for $0 \leqslant n_1, t_1 \leqslant d_1 - 1$, $0 \leqslant n_2, t_2 \leqslant d_2/2^r - 1$

In Eq. (20), define $A'_{t_1}(u) = A^*_{t_1}(cu)$, $B'_{t_1}(u) = B^*_{t_1}(cu)$, and $C'_{t_1}(u) = C^*_{n_1}(cu)$. Then Eq. (19) becomes

$$C'_{n_1}(u) = \sum_{t_1=0}^{d_1-1} A'_{t_1}(u)\, B'_{(n_1-t_1)}(u) \bmod \left(u^{d_2/2^r} + 1\right) \quad \text{for } 0 \leqslant n_1 \leqslant d_1 - 1$$

Using the same procedure used in the computation of Eq. (7), one obtains $C'_{n_1}(u)$. Hence, $c^*_{n_1,n_2}$ in Eq. (18) is obtained from the substitutions, $c^{-n_2} c'_{n_1,n_2}$. The number of real multiplications and real additions needed to compute Eq. (19) are $d_1 d_2/2^{r-2} + d_1 d_2/2^{r-2} \log(d_2/2^r) + 16\, d_1(d_2/2^r - 1)$ and $2d_1 d_2 + d_1(d_2/2^{r-2}) \log d_1 + 3\, d_1(d_2/2^{r-1}) \log(d_2/2^r)$, respectively. Hence the total number of real multiplications and real additions needed to compute Eq. (4) are $8\, d_1 [(d_2/2^{r+1})(2^r(\log d_2 + 3) + 4) - (d_2 + r + 2)]$ and $2 \cdot d_1 [d_2(2r-4) + d_2(2 \cdot \log d_1 - 1 + 3 \cdot \log d_2) + 6 \cdot d_2/2^r]$, respectively. The flowchart of this new algorithm is shown in Fig. 2.

In the introduction it was stated that transposition of the data matrix, usually required in two-dimensional convolution, can be avoided in this new algorithm. Assume the data matrix for a typical array such as a_{t_1,t_2}, $0 \leqslant t_i \leqslant d_i - 1$ for $i = 1,2$, is arranged with t_1 indexing the row, t_2 indexing the column. Then the polynomial $A_{t_1}(Z)$ of Eq. (2) is the t_1 th row of this matrix. The FPT of $A_{t_1}(Z)$ can be implemented by a decimation-in-time algorithm analogous to the well-known conventional FFT one. An example is given in the Appendix. This requires that only two polynomials (or rows) need be available for processing at anytime and the resulting two polynomials can replace the input ones. Consequently, no additional storage is required for the FPT, and the data array need only be accessed by rows with replacement after processing, thus obviating the accessing of individual columns usually done in two dimensional convolution.

In the Appendix, it is shown by an example how fast polynomial transforms can be combined with FFTs to yield a new fast algorithm for computing a two-dimensional convolution. The number of operations needed for this new algorithm to perform the two-dimensional convolutions of a $d_1 \cdot d_2$ array, where $d_2 = 2^m$ and $d_1 = 2^{m-r+1}$ for $1 \leqslant r \leqslant m$, is given in Table 1. In this table, the FPT-FFT algorithm and conventional FFT algorithm for computing the two-dimensional convolutions are compared by giving the number of operations to perform these algorithms.

Table 1. Complexity of new algorithm for two-dimensional convolutions

	FPT-FFT-algorithm			Radix-2 FFT algorithm	
Convolution size $d = d_1 \times d_2 = 2^{m-r+1} \times 2^m$ for $1 \leqslant r \leqslant m$	No. of factors of $z^{d_2}-1=r+1$	No. of real multiplication	No. of real addition	No. of real multiplication $4\, d_1 d_2\, (\log d_2 + \log d_1 + 1)$	No. of real addition $6\, d_1 d_2\, (\log d_2 + \log d_1) + 2\, d_1 d_2$
$2^7 \times 2^9$	4	2,747,392	5,537,792	4,456,448	6,422,528
$2^7 \times 2^{10}$	5	5,892,096	12,288,000	9,437,184	13,631,488
$2^{11} \times 2^{11}$	2	234,831,872	457,179,136	385,875,968	562,036,736
$2^{10} \times 2^{11}$	3	109,019,136	222,298,112	184,549,376	268,435,456
$2^8 \times 2^{12}$	6	55,035,904	118,882,304	88,080,384	127,926,272
$2^8 \times 2^{13}$	7	117,948,416	258,342,912	184,549,376	268,435,456
$2^7 \times 2^{13}$	8	58,842,112	129,073,152	88,080,384	127,926,272

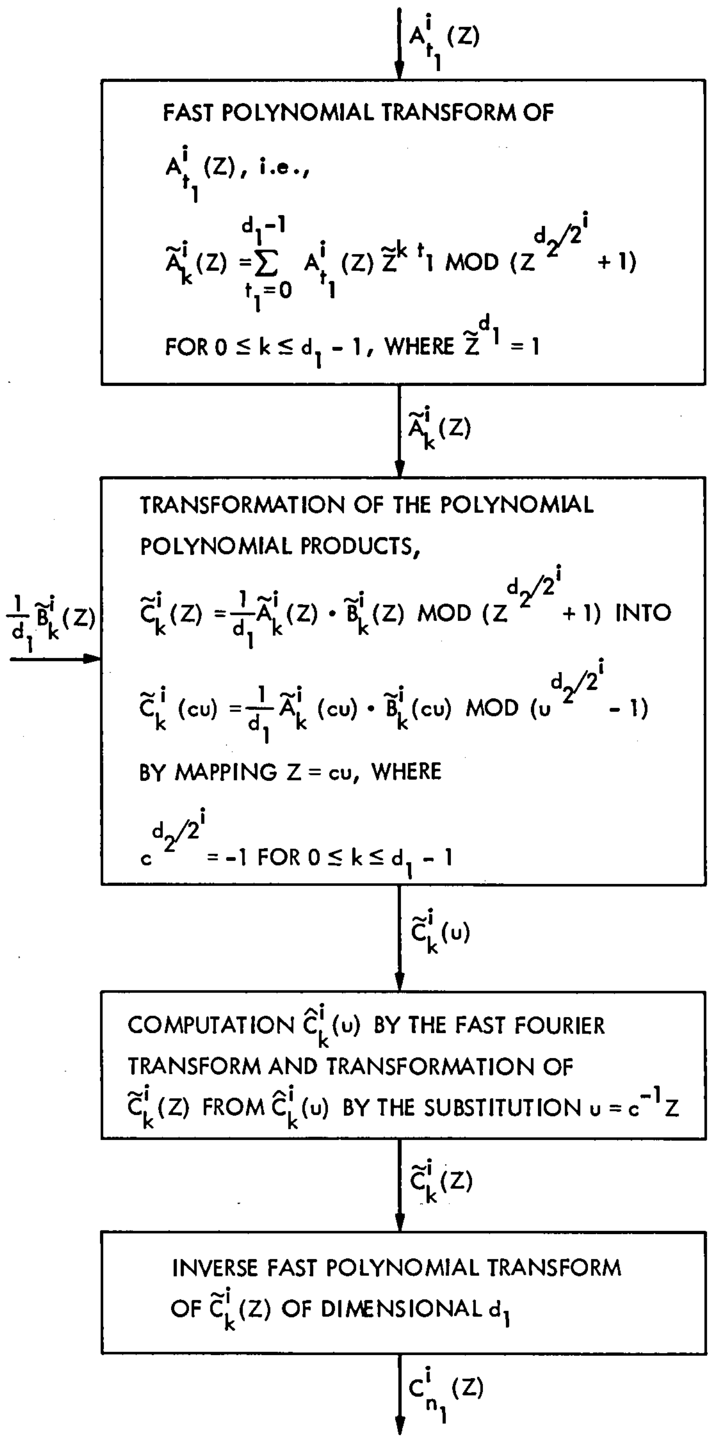
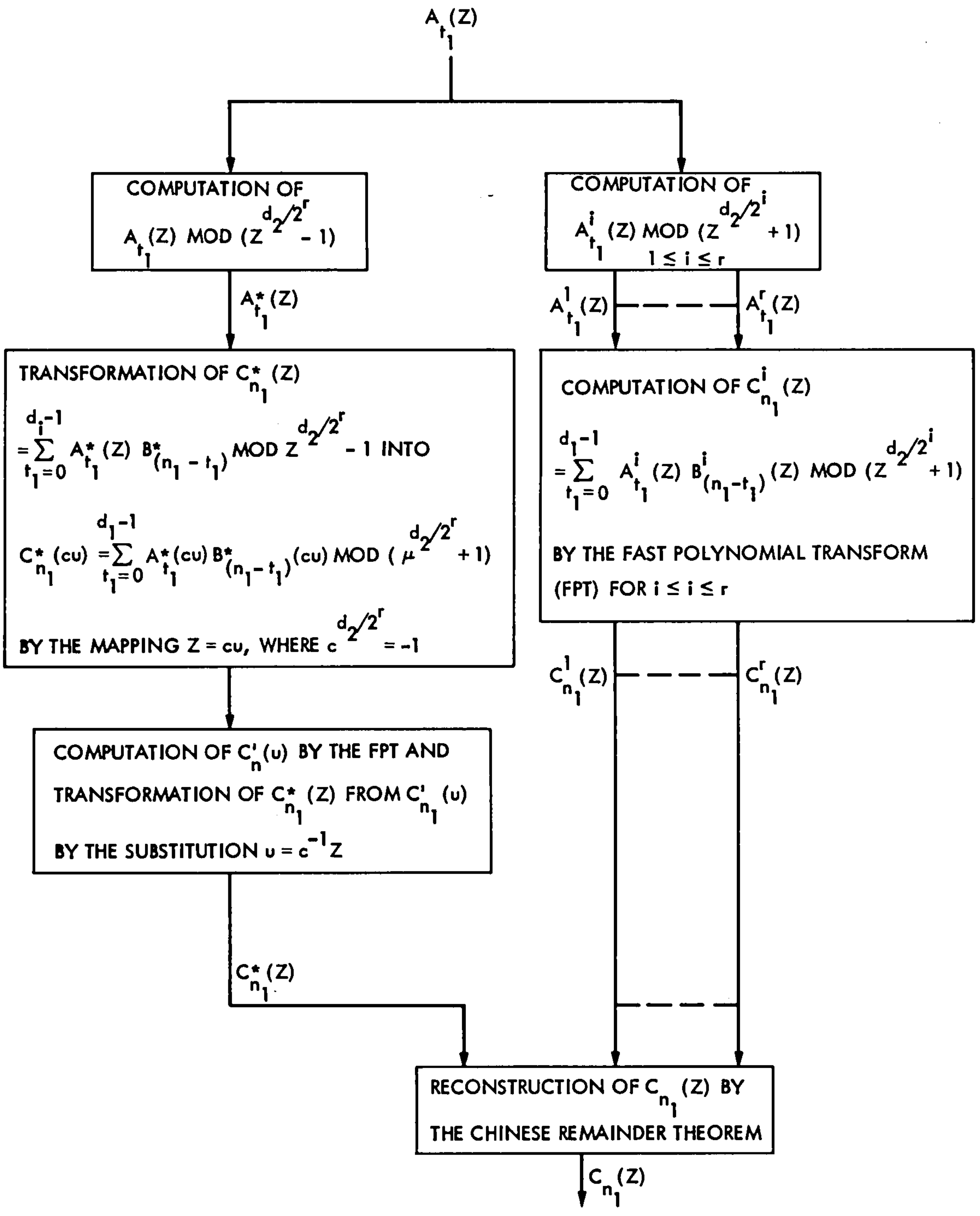

$A^i_{t_1}(Z)$
FAST POLYNOMIAL TRANSFORM OF $A^i_{t_1}(Z)$, i.e.,
$\tilde{A}^i_k(Z) = \sum_{t_1=0}^{d_1-1} A^i_{t_1}(Z)\,\tilde{Z}^{k\,t_1} \text{ MOD } (Z^{d_2/2^i}+1)$
FOR $0 \le k \le d_1 - 1$, WHERE $\tilde{Z}^{d_1} = 1$
$\tilde{A}^i_k(Z)$
$\frac{1}{d_1}\tilde{B}^i_k(Z)$
TRANSFORMATION OF THE POLYNOMIAL POLYNOMIAL PRODUCTS,
$\tilde{C}^i_k(Z) = \frac{1}{d_1}\tilde{A}^i_k(Z) \cdot \tilde{B}^i_k(Z) \text{ MOD } (Z^{d_2/2^i}+1)$ INTO
$\tilde{C}^i_k(cu) = \frac{1}{d_1}\tilde{A}^i_k(cu) \cdot \tilde{B}^i_k(cu) \text{ MOD } (u^{d_2/2^i}-1)$
BY MAPPING $Z = cu$, WHERE
$c^{d_2/2^i} = -1$ FOR $0 \le k \le d_1 - 1$
$\tilde{C}^i_k(u)$
COMPUTATION $\hat{C}^i_k(u)$ BY THE FAST FOURIER TRANSFORM AND TRANSFORMATION OF $\tilde{C}^i_k(Z)$ FROM $\hat{C}^i_k(u)$ BY THE SUBSTITUTION $u = c^{-1}Z$
$\tilde{C}^i_k(Z)$
INVERSE FAST POLYNOMIAL TRANSFORM OF $\tilde{C}^i_k(Z)$ OF DIMENSIONAL d_1
$C^i_{n_1}(Z)$

Fig. 1. Computation of the cyclic convolution of two Z-polynomials mod $Z^{d_2/2^i} + 1$, i.e., $C^i_{n_1}(Z)$ given in Fig. 2 by fast polynomial transforms



$A_{t_1}(Z)$
COMPUTATION OF $A_{t_1}(Z)$ MOD $(Z^{d_2/2^r} - 1)$
COMPUTATION OF $A^i_{t_1}(Z)$ MOD $(Z^{d_2/2^i} + 1)$ $1 \le i \le r$
$A^*_{t_1}(Z)$
$A^1_{t_1}(Z)$
$A^r_{t_1}(Z)$
TRANSFORMATION OF $C^*_{n_1}(Z)$ $= \sum_{t_1=0}^{d_1-1} A^*_{t_1}(Z)\, B^*_{(n_1-t_1)}$ MOD $Z^{d_2/2^r} - 1$ INTO
$C^*_{n_1}(cu) = \sum_{t_1=0}^{d_1-1} A^*_{t_1}(cu)\, B^*_{(n_1-t_1)}(cu)$ MOD $(\mu^{d_2/2^r} + 1)$
BY THE MAPPING $Z = cu$, WHERE $c^{d_2/2^r} = -1$
COMPUTATION OF $C^i_{n_1}(Z)$ $= \sum_{t_1=0}^{d_1-1} A^i_{t_1}(Z)\, B^i_{(n_1-t_1)}(Z)$ MOD $(Z^{d_2/2^i} + 1)$
BY THE FAST POLYNOMIAL TRANSFORM (FPT) FOR $i \le i \le r$
$C^1_{n_1}(Z)$
$C^r_{n_1}(Z)$
COMPUTATION OF $C'_n(u)$ BY THE FPT AND TRANSFORMATION OF $C^*_{n_1}(Z)$ FROM $C'_{n_1}(u)$ BY THE SUBSTITUTION $u = c^{-1}Z$
$C^*_{n_1}(Z)$
RECONSTRUCTION OF $C_{n_1}(Z)$ BY THE CHINESE REMAINDER THEOREM
$C_{n_1}(Z)$

Fig. 2. Computation of a two-dimensional convolution of dimensional $d_1 x d_2$, where $d_2 = 2^m$ and $d_1 = 2^{m-r+1}$ for $1 \le r \le m$

Appendix

Example of a Two-Dimensional Convolution of Dimension Computed by the New Method

Example: Compute the two-dimensional cyclic convolution of two sequences a_{t_1,t_2} and b_{t_1,t_2}, i.e.,

$$c_{n_1,n_2} = \sum_{t_1=0}^{4-1} \sum_{t_2=0}^{4-1} a_{t_1,t_2}\, b_{(n_1-t_2),(n_2-t_2)} \quad \text{for } 0 \leqslant n_1,n_2 \leqslant 3 \tag{A-1}$$

where (x) denotes the residue of x modulo 4 and where $a_{0,0} = y_{0,0} = 1, a_{0,1} = b_{0,1} = 1, a_{0,2} = b_{0,2} = 0, a_{0,3} = b_{0,3} = 0,$ $a_{1,0} = b_{1,0} = 1, a_{1,1} = b_{1,1} = 1, a_{1,2} = b_{1,2} = 1, a_{1,3} = b_{1,3} = 0, a_{2,0} = b_{2,0} = 0, a_{2,1} = b_{2,1} = 0, a_{2,2} = b_{2,2} = 0, a_{2,3} =$ $b_{2,3} = 0, a_{3,0} = b_{3,0} = 0, a_{3,1} = b_{3,1} = 0, a_{3,2} = b_{3,2} = 0, a_{3,3} = b_{3,3} = 0.$

Define

$$A_{t_1}(Z) = \sum_{t_2=0}^{4-1} a_{t_1,t_2} Z^{t_2}, B_{t_1}(Z) = \sum_{t_2=0}^{4-1} b_{t_1,t_2} Z^{t_2};$$

$$C_{n_1}(Z) = \sum_{n_2=0}^{4-1} c_{n_1,n_2} Z^{n_2} \quad \text{for } 0 \leqslant n_1,t_1 \leqslant 3$$

That is, $A_0(Z) = B_0(Z) = 1 + Z, A_1(Z) = B_1(Z) = 1 + Z + Z^2, A_2(Z) = B_2(Z) = 0, A_3(Z) = B_3(Z) = 0$. From Eq. (4), (A-1) becomes

$$C_{n_1}(Z) = \sum_{t_1=0}^{4-1} A_{t_1}(Z)\; B_{(n_1-t_1)}(Z) \bmod (Z^2 - 1)(Z^2 + 1) \tag{A-2}$$

From Eqs. (5a) and (5b) one obtains

$$C^1_{n_1}(Z) = \sum_{t_1=0}^{4-1} A^1_{t_1}(Z) B^1_{(n_1-t_1)}(Z) \bmod (Z^2 + 1) \tag{A-3a}$$

where $A^1_0(Z) = B^1_0(Z) = 1 + Z, A^1_1(Z) = B^1_1(Z) = Z, A^1_2(Z) = B^1_2(Z) = 0, A^1_3(Z) = B^1_3(Z) = 0.$

and

$$C^*_n(Z) = \sum_{t_1=0}^{4-1} A^*_{t_1}(Z) B^*_{(n_1-t_1)}(Z) \bmod (Z^2 - 1) \tag{A-3b}$$

where $A^*_0(Z) = B^*_0(Z) = 1 + Z, A^*_1(Z) = B^*_1(Z) = 2 + Z, A^*_2(Z) = B^*_2(Z) = 0, A^*_3(Z) = B^*_3(Z) = 0.$

To compute Eq. (A-3a), one takes the Z polynomial transform of $A^1_{t_1}(Z)$ given by

$$\widetilde{A}_k(Z) = \sum_{t_1=0}^{4-1} A^1_{t_1}(Z) Z^{t_1 k} \bmod (Z^2 + 1)$$

The flowgraph of the decimation-in-time decomposition of a four-point polynomial transform computation is given in Fig. 1-A as $\widetilde{A}_0(Z) = 1 + 2Z$, $\widetilde{A}_1(Z) = Z$, $\widetilde{A}_2(Z) = 1$, and $\widetilde{A}_3(Z) = 2 + Z$. Similarly, the Z polynomial transform of $\widetilde{B}_k(Z)$ are $\widetilde{B}_0(Z) = 1 + 2Z$, $\widetilde{B}_1(Z) = Z$, $\widetilde{B}_2(Z) = 1$, and $\widetilde{B}_3(Z) = 2 + Z$.

Let

$$\widetilde{C}_k(Z) = \widetilde{A}_k(Z) \cdot \widetilde{B}_k(Z) \bmod (Z^2 + 1) \text{ for } k = 0, 1, 2, 3 \tag{A-4}$$

If one uses the mapping $Z = iu$, where $i^2 = -1$ then Eq. (A-4) becomes

$$\widetilde{C}_k(iu) = \widetilde{A}_k(iu) \cdot \widetilde{B}_k(iu) \bmod (u^2 - 1) \text{ for } 0 \leqslant k \leqslant 3 \tag{A-5}$$

For $t = 0$, Eq. (A-5) becomes

$$\widetilde{C}_0(iu) \equiv \widetilde{A}_0(iu) \cdot \widetilde{B}_0(iu) \equiv (1 + 2iu)(1 + 2iu) \bmod (u^2 - 1) \tag{A-6}$$

The FFT can be used to compute Eq. (A-6). That is, let $a_0 = 1, a_1 = 2i, b_0 = 1, b_1 = 2i$. The transform of a_n is

$$A_k = \sum_{k=0}^{2-1} a_n w^{nk} = \sum_{n=0}^{2-1} a_n(-1)^{nk} = 1 + 2i(-1)^k \quad \text{for } k = 0, 1 \tag{A-7}$$

The radix-2 FFT algorithm is used to compute Eq. (A-7). The results are $A_0 = 1 + 2i$ and $A_1 = 1 - 2i$. Similarly, one obtains $B_0 = 1 + 2i$ and $B_1 = 1 - 2i$. Let $C_0 = A_0 \cdot B_0 = -3 + 4i$, and $C_1 = A_1 \cdot B_1 = -3 - 4i$. The inverse Fourier transform of C_k is

$$\begin{aligned} c_n &= 2^{-1} \sum_{k=0}^{2-1} c_k w^{-nk} = 2^{-1} \sum_{k=0}^{2-1} c_k(-1)^{-nk} \\ &= 2^{-1} \left[(-3 + 4i) + (-3 - 4i)(-1)^{-n}\right] \quad \text{for } n = 0, 1 \end{aligned} \tag{A-8}$$

Again, the radix-2 FFT algorithm is used to compute Eq. (A-8). That is, $c_0 = -3$ and $c_1 = 4i$. Thus, $\widetilde{C}_0(iu) = -3 + 4iu \bmod (u^2 - 1)$. Hence, one obtains $\widetilde{C}_0(Z) = -3 + 4i^{-1}iZ = -3 + 4Z \bmod (Z^2 + 1)$.

Similarly one obtains $\widetilde{C}_1(Z) = -1$, $\widetilde{C}_2(Z) = 1$, and $\widetilde{C}_3(Z) = 3 + 4Z$. The inverse polynomial transform of $\widetilde{C}_k(Z)$ is given by

$$C^1_{n_1}(Z) = 4^{-1} \sum_{k=0}^{4-1} \widetilde{C}_k(Z) \cdot Z^{-n_1 k} \tag{A-9}$$

The FPT is used to compute Eq. (A-9). The results in Eq. (A-3) are $C^1_0(Z) = 2Z$, $C^1_1(Z) = -2 + 2Z$, $C^1_2(Z) = -1$, and $C^1_3(Z) = 0$.

To compute Eq. (A-3b), one needs to use the mapping $Z = iu$. Then Eq. (A-3b) becomes

$$C^*_{n_1}(iu) \equiv \sum_{t_1=0}^{4-1} A^*_{t_1}(iu) B^*_{(n_1-t_1)}(iu)$$

$$\equiv c^*_{n_1,0} + i c^*_{n_1,1} u \mod (u^2+1) \quad \text{for } 0 \leqslant n_1 \leqslant 3 \tag{A-10}$$

where $A^*_0(iu) = B^*_0(iu) = 1 + iu$, $A^*_1(iu) = B^*_1(iu) = 2 + iu$, $A^*_2(iu) = B^*_2(iu) = 0$, $A^*_3(iu) = B^*_3(iu) = 0$.

Let $A'_0(u) = B'_0(u) = 1 + iu$, $A'_1(u) = B'_1(u) = 2 + iu$, $A'_2(u) = B'_2(u) = 0$, $A'_3(u) = B'_3(u) = 0$ and $C'_{n_1}(u) = c'_{n_{1,0}} + c'_{n_{1,0}} u$ for $0 \leqslant n_1 \leqslant 3$, where $c'_{n_{1,0}} = c^*_{n_{1,0}}$ and $c'_{n_{1,1}} = ic^*_{n_{1,0}}$. Hence, Eq. (A-10) becomes

$$C'_{n_1}(u) = \sum_{t_1=0}^{4-1} A'_{t_1}(u) B'_{(n_1-t_1)}(u) \mod (u^2+1) \tag{A-11}$$

Equation (A-11) can be computed by the FPT. This yields $C'_0(u) = 2 + 2iu$, $C'_1(u) = 6 + 6\,iu$, $C'_2(u) = 5 + 4\,iu$, and $C'_3(u) = 0$. Hence

$$C^*_0(Z) = C'_0(i^{-1}Z) = 2 + 2Z,\ C^*_1(Z) = C'_1(i^{-1}Z) = 6 + 6Z\,;$$

$$C^*_2(Z) = C'_2(i^{-1}Z) = 5 + 2Z,\ C^*_3(Z) = C'_3(i^{-1}Z) = 0 \tag{A-12}$$

From Eq. (6), one obtains

$$C_{n_1}(Z) = \frac{1}{2}\left[C^*_{n_1}(Z)(Z^2+1) + C^1_{n_1}(Z)(1 - Z^2)\right] \mod (Z^4 - 1)$$

$$\text{for } 0 \leqslant n_1 \leqslant 3$$

where $C^1_{n_1}(Z)$ and $C^*_{n_1}(Z)$ are defined in Eqs. (A-3a) and (A-3b), respectively. Thus, the desired results are $C_0(Z) = 1 + 2Z + Z^2$, $C_1(Z) = 2 + 4Z + 4Z^2 + 2Z^3$, $C_2(Z) = 2 + 2Z + 3Z^2 + 2Z^3$, and $C_3(Z) = 0$. Hence $c_{0,0} = 1, c_{0,1} = 2, c_{0,2} = 1, c_{0,3} = 0, c_{1,0} = 2, c_{1,1} = 4, c_{1,2} = 4, c_{1,3} = 2, c_{2,0} = 2, c_{2,1} = 2, c_{2,2} = 3, c_{2,3} = 2, c_{3,0} = 0, c_{3,1} = 0, c_{3,2} = 0$, and $c_{3,3} = 0$.

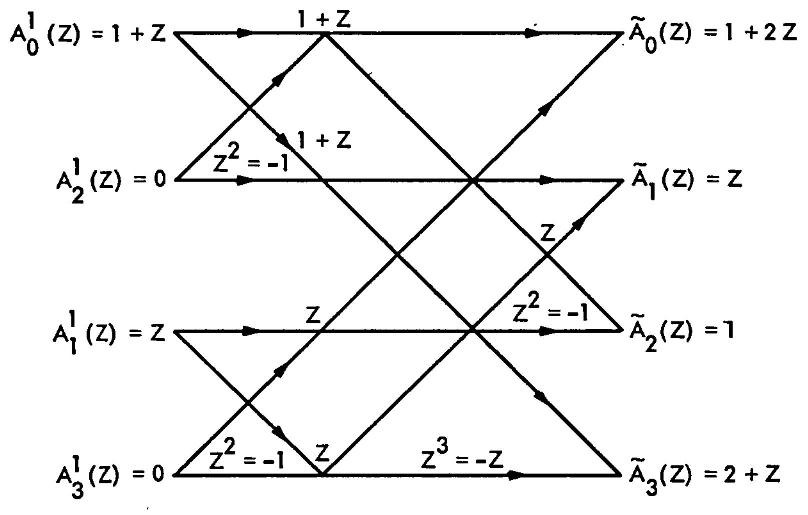


Fig. 1A. Example of four-point fast polynomial transform

References

1. Wu, C., "A Digital System to Produce Imagery from SAR Data," *AIAA Systems Design Driven by Sensors*, Pasadena, California, October 18-20, 1976.

2. Tomiyasu, K., "Tutorial Review of Synthetic – Aperture Radar (SAR) With Applications to Imaging of the Ocean Surface," *Proceedings of the IEEE*, Vol. 66, No. 5, May 1978, pp. 563-583.

3. Rader, C. M., "On the Application of the Number Theoretic Methods of High-Speed Convolution to Two-Dimensional Filtering," *IEEE Trans. Acoust. Speed Signal Processing ASSP-23*, 575 (1975), pp. 575.

4. Reed, I. S., Truong, T. K., Kwoh, Y. S., and Hall, E. L., "Image Processing by Transforms Over a Finite Field," *IEEE Transactions on Computers,* Vol. C-26, No. 9, September 1977, pp. 874-881.

5. Nussbaumer, H. J., and Quandalle, P., "Computation of Convolutions and Discrete Fourier Transforms by Polynomial Transforms," *IBM J. Res. Develop.*, Vol. 22, No. 2, Mar. 1978, pp. 134-144.

6. Arambepola, B., and Rayner, P. J. W., "Efficient Transforms for Multidimensional Convolutions," *Electronics Letters*, 15 March 1979, Vol. 15, No. 6, pp. 189-190.

7. Berlekamp, E. R., *Algebraic Coding Theory*, McGraw-Hill Book Company, New York, 1968.

8. Pollard, J. M., "The Fast Fourier Transform in a Finite Field," *Math Comput.*, Vol. 25, No. 114, April 1971, pp. 365-374.

9. Cooley, J. W., and Tukey, J. W., "An Algorithm for the Machine Calculation of Complex Fourier Series," *Math Comput.*, Vol. 19, April 1965, pp. 297-301.