H. J. Nussbaumer
P. Quandalle

Computation of Convolutions and Discrete Fourier Transforms by Polynomial Transforms

Abstract: Discrete transforms are introduced and are defined in a ring of polynomials. These polynomial transforms are shown to have the convolution property and can be computed in ordinary arithmetic, without multiplications. Polynomial transforms are particularly well suited for computing discrete two-dimensional convolutions with a minimum number of operations. Efficient algorithms for computing one-dimensional convolutions and Discrete Fourier Transforms are then derived from polynomial transforms.

Introduction

The calculation of two-dimensional digital convolutions has many applications, particularly in image processing. The main problem associated with these convolutions relates to the huge processing load required for their computation.

Direct calculation of a two-dimensional convolution of dimension $N \times N$ corresponds to N^4 multiplications and $N^2(N^2 - 1)$ additions. A substantial reduction in the number of operations can be achieved if direct computation is replaced by a Fast Fourier Transform (FFT) [1] or a Number Theoretic Transform (NTT) [2-6] approach. These two techniques, however, do not provide an optimum solution for evaluating two-dimensional convolutions. Computation by means of FFTs introduces a significant amount of roundoff errors, requires some means of processing trigonometric functions, and corresponds to still a large amount of multiplying. NTTs can be calculated without multiplications, and allow the computation of convolutions without quantization errors. However, such transforms suffer severe word length and transform length limitations. Moreover, implementation of modular arithmetic in general purpose computers is sometimes inefficient.

Recently, Agarwal and Cooley [7, 8] introduced new algorithms for digital convolutions. Their method is based upon a nesting of several short convolutions having lengths $N_1, N_2, N_3, \cdots$, which are relatively prime, and yields a total number of multiplications equal to $M_1M_2M_3, \cdots$, where $M_1, M_2, M_3, \cdots$ are the number of multiplications required to calculate the short convolutions of lengths $N_1, N_2, N_3, \cdots$, and are such that $M_i \approx 2N_i$. This technique, applicable to one-dimensional and multidimensional convolutions, is attractive because it appears to be computationally more efficient than the FFT for lengths up to 400 and because it does not place any restrictions on the arithmetic nor require any manipulation of trigonometric functions.

In this paper, we extend earlier work by Nussbaumer [9] on polynomial transforms to cover the various polynomial transforms that can be calculated without multiplications. We show that these transforms have the convolution property and yield efficient algorithms for computing two-dimensional convolutions in ordinary arithmetic. These algorithms correspond to a number of multiplications equal to $M_1N_2N_3 \cdots$ rather than $M_1M_2M_3 \cdots$ with the Agarwal-Cooley method and are therefore particularly efficient for large convolutions. These algorithms are then extended to one-dimensional convolutions.

In this case, since two-dimensional convolutions computed by polynomial transforms are usually such that both dimensions have a common factor, the efficient two-dimensional to one-dimensional mapping introduced by Agarwal and Cooley [7, 8] is not applicable, and the computation of one-dimensional convolutions is done at the expense of some reduction in computing efficiency by using the approach proposed by Agarwal and Burrus in [10].

We show also that polynomial transforms can be used for computing Discrete Fourier Transforms (DFTs) and allow, in some cases, a significant reduction in number of operations when compared to the Winograd Fourier Transform Algorithm (WFTA) [11-13].

Polynomial transforms

Let $y_{u,l}$ be a two-dimensional circular convolution of dimension $q \times q$ with

$$y_{u,l} = \sum_{m=0}^{q-1} \sum_{n=0}^{q-1} h_{n,m} x_{u-n,l-m}. \tag{1}$$

In polynomial notations, $y_{u,l}$ can be obtained from a set of q polynomials $Y_l(Z)$, by taking the coefficient of Z^u in the polynomial $Y_l(Z)$ with

$$Y_l(Z) = \sum_{u=0}^{q-1} y_{u,l} Z^u \qquad l = 0, 1, \cdots, q-1. \tag{2}$$

Under these conditions, the two-dimensional convolution (1) can be viewed as a one-dimensional polynomial convolution with

$$Y_l(Z) \equiv \sum_{m=0}^{q-1} H_m(Z) X_{l-m}(Z) \qquad \text{mod } (Z^q - 1); \tag{3}$$

$$H_m(Z) = \sum_{n=0}^{q-1} h_{n,m} Z^n \qquad m = 0, 1, \cdots, q-1; \tag{4}$$

$$X_r(Z) = \sum_{s=0}^{q-1} x_{s,r} Z^s \qquad r = 0, 1, \cdots, q-1. \tag{5}$$

We assume first that q is an odd prime. In this case, $Z^q - 1$ factors into a product of two irreducible cyclotomic polynomials [14]:

$$\begin{aligned} Z^q - 1 &= (Z-1)(Z^{q-1} + Z^{q-2} + \cdots + 1) \\ &= (Z-1)M(Z), \end{aligned} \tag{6}$$

with $M(Z) = Z^{q-1} + \cdots + 1$. $Y_l(Z)$ can be recovered from $Y_{1,l}(Z) \equiv Y_l(Z)$ modulo $M(Z)$ and $Y_{2,l} \equiv Y_l(Z)$ modulo $(Z-1)$ by using the Chinese remainder theorem extended to the ring of polynomials, with

$$Y_l(Z) \equiv Y_{1,l}(Z) S_1(Z) + Y_{2,l} S_2(Z) \qquad \text{mod } (Z^q - 1) \tag{7}$$

$$S_1(Z) \equiv 1 \quad S_2(Z) \equiv 0 \qquad \text{mod } M(Z) \tag{8}$$

$$S_1(Z) \equiv 0 \quad S_2(Z) \equiv 1 \qquad \text{mod } (Z-1) \tag{9}$$

and

$$S_1(Z) = (q - M(Z))/q; \tag{10}$$

$$S_2(Z) = M(Z)/q. \tag{11}$$

Since $Y_{2,l}$ is defined modulo $(Z-1)$, its calculation reduces to that of a single one-dimensional scalar convolution, with

$$Y_{2,l} = \sum_{m=0}^{q-1} H_{2,m} X_{2,l-m} \qquad l = 0, 1, \cdots, q-1; \tag{12}$$

$$H_{2,m} = \sum_{n=0}^{q-1} h_{n,m} \qquad X_{2,r} = \sum_{s=0}^{q-1} x_{s,r}. \tag{13}$$

Under these conditions, the bulk of the computation of $Y_l(Z)$ corresponds to the calculation of $Y_{1,l}(Z)$.

In order to simplify the computation of $Y_{1,l}(Z)$ we introduce a transform $\bar{H}_k(Z)$ defined modulo $M(Z)$, with

$$\bar{H}_k(Z) \equiv \sum_{m=0}^{q-1} H_{1,m}(Z) Z^{mk} \quad \text{mod } M(Z)$$

$$H_{1,m}(Z) \equiv H_m(Z) \quad \text{mod } M(Z);$$

$$k = 0, 1, \cdots, q-1. \tag{14}$$

We define similarly an inverse transform by

$$H_{1,l}(Z) \equiv \frac{1}{q} \sum_{k=0}^{q-1} \bar{H}_k(Z) Z^{-lk} \quad \text{mod } M(Z)$$

$$l = 0, 1, \cdots, q-1. \tag{15}$$

The computation of these polynomial transforms reduces to multiplications by powers of Z and additions. Since $Z^q \equiv 1$ modulo $M(Z)$, the multiplications of polynomials $H_{1,m}(Z)$ by powers of Z correspond to simple rotations of the words $h_{n,m}$ within a q-word polynomial followed by a reduction modulo $M(Z)$. Polynomial additions are performed by adding separately the words corresponding to each coefficient of Z. Thus, polynomial transforms can be computed with simple additions, with the advantage over NTTs that operations can be implemented in ordinary arithmetic, without any restrictions on word lengths or truncation.

We show now that $Y_{1,l}(Z)$ can be evaluated efficiently by polynomial transforms. To see this, we compute the transforms $\bar{H}_k(Z)$ and $\bar{X}_k(Z)$ of $H_m(Z)$ and $X_r(Z)$ by (14), we multiply modulo $M(Z)$ the results term by term and we calculate the inverse transform $P_l(Z)$ of the polynomial sequence $\bar{H}_k(Z)\bar{X}_k(Z)$:

$$P_l(Z) \equiv \sum_{m=0}^{q-1} \sum_{r=0}^{q-1} H_{1,m}(Z) X_{1,r}(Z) \frac{1}{q} \sum_{k=0}^{q-1} Z^{(m+r-l)k}$$

$$\text{mod } M(Z). \tag{16}$$

Let $t = m + r - l$ and $S = \Sigma_{k=0}^{q-1} Z^{tk}$. For $t \equiv 0$ modulo q, $S \equiv q$. For $t \not\equiv 0$ modulo q, the set of exponents tk modulo q is merely a permutation of the integers $0, 1, \cdots, q-1$, so that $S \equiv \Sigma_{k=0}^{q-1} Z^k = M(Z) \equiv 0$ modulo $M(Z)$. Therefore,

$$P_l(Z) \equiv Y_{1,l}(Z) \equiv \sum_{m=0}^{q-1} H_{1,m}(Z) X_{1,l-m}(Z) \text{ mod } M(Z). \tag{17}$$

Thus, the computation of $Y_{1,l}(Z)$ reduces to the calculation of three polynomial transforms and to q polynomial multiplications $\bar{H}_k(Z)\bar{X}_k(Z)$ defined modulo $M(Z)$. In most filtering applications, one of the input sequences is constant and its transform can be precomputed and premultiplied by $1/q$. In this case, only two polynomial transforms are required and the computation process reduces to that shown in Fig. 1.

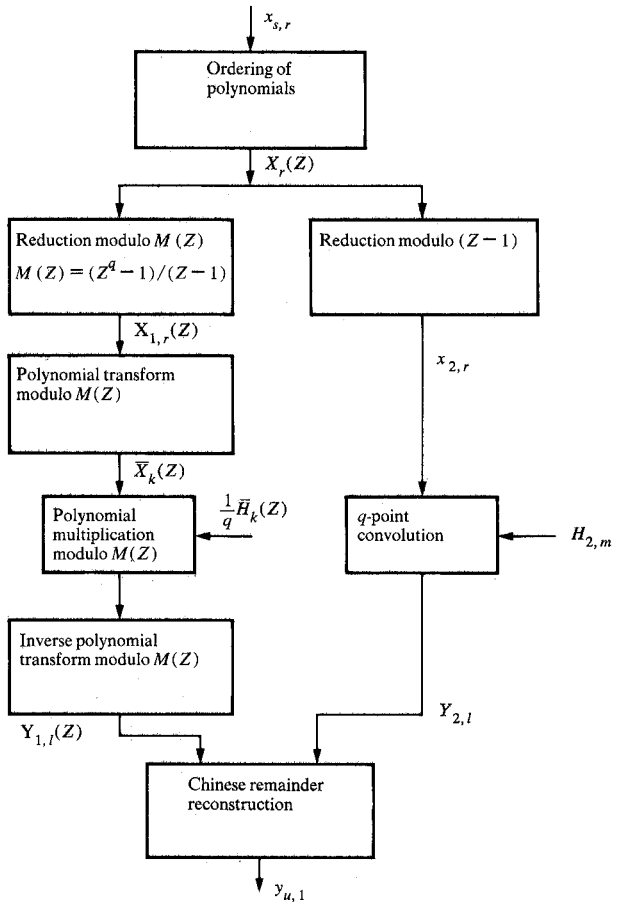

Figure 1 Computation of two-dimensional convolutions by polynomial transforms with q prime.

The organization of the calculations can be simplified at the expense of a slightly larger number of operations by calculating the polynomial transforms and the polynomial multiplications modulo $(Z^q - 1)$ instead of modulo $M(Z)$ with a final and unique operation modulo $M(Z)$ prior to Chinese remainder reconstruction. We call these transforms pseudo-polynomial transforms by analogy with the pseudo-Mersenne Transforms introduced by Nussbaumer in [5].

Computation of two-dimensional convolutions by polynomial transforms

We have seen above that a fixed taps $q \times q$-point convolution could be computed with two reductions modulo $(Z - 1)$ and modulo $M(Z)$, two polynomial transforms, one Chinese remainder operation, one q-point convolution, and q polynomial multiplications modulo $M(Z)$. In all of these operations, the only multiplications are those required for computing the q-point convolution and the q polynomial multiplications modulo $M(Z)$.

In the case of computation by pseudo-polynomial transforms, the number of multiplications corresponds to $(q + 1)$ convolutions of q terms.

These short convolutions and polynomial multiplications can be computed by FFTs or NTTs. The problems associated with the use of these transforms such as round-off errors for FFT or excessive word length for NTT are then limited to only a part of the total computation process.

It is, however, often more advantageous to perform these computations by using short convolution algorithms. If M_1 and M_2 are respectively the number of multiplications required to compute a convolution of dimension q and a polynomial product modulo $M(Z)$, the total number of multiplications corresponding to a convolution of dimension $q \times q$ becomes $(q + 1)M_1$ with pseudo-polynomial transforms and $qM_2 + M_1$ with polynomial transforms. Thus, the algorithm derived from polynomial transforms yields about N_1M_1 multiplications for a convolution of dimension $N_1 \times N_1$ as against M_1M_1 for the Agarwal-Cooley approach [7, 8].

Winograd [11, 12] has shown that convolutions of dimension q and polynomial products modulo $M(Z)$, with q prime, can be computed in the field of rational numbers with a minimum number of multiplications equal to $2(q - 1)$ and $2q - 3$ respectively. Thus, for small q, the polynomial transform method can be expected to require about half as many multiplications as the Agarwal-Cooley approach. We shall see later that much larger savings can be achieved when composite algorithms are used.

When computing two-dimensional convolutions by polynomial transforms, it would seem that the Chinese remainder reconstruction (7)–(10) requires a large number of additions. This need not be the case, however, provided the computation is properly organized: Since $S_1(Z) \equiv 0$ modulo $(Z - 1)$, one can define $T(Z)$ so that $S_1(Z) \equiv [q - M(Z)]/q \equiv T(Z)(Z - 1)/q$, and

$$Y_l(Z) \equiv [T(Z)Y_{1,l}(Z)(Z - 1) + M(Z)Y_{2,l}]/q \quad \text{mod } (Z^q - 1). \qquad (18)$$

$T(Z)$ is given by $T(Z) = [-Z^{q-2} + \cdots + (3 - q)Z^2 + (2 - q)Z + 1 - q]$. When the filter has fixed taps, $\bar{H}_k(Z)/q$ and $H_{2,m}$ are computed only once. Thus, $\bar{H}_k(Z)/q$ can be premultiplied by $T(Z)/q$ and $H_{2,m}$ can be premultiplied by $1/q$. Under these conditions, the Chinese remainder reconstruction does not require any multiplication by q or $1/q$ and reduces to multiplying $Y_{2,l}/q$ by $(Z^{q-1} + \cdots + 1)$ and $Y_{1,l}(Z)T(Z)/q$ by $(Z - 1)$. These two operations can be done with only $2q(q - 1)$ additions.

We give in Table 1, line 1, the total number of operations for convolutions of dimension $q \times q$ computed by polynomial transforms. The number of additions and multiplications is given in Table 2, columns 2–5. In this table,

Table 1 Polynomial transforms for the computation of convolutions.

q	*Transform ring* $M(Z)$	*Transform*		*Dimension of convolutions*	*No. of additions*	*Generalized multiplications*
		Length	*Root*			
prime	$(Z^q - 1)/(Z - 1)$	q	Z	$q \times q$	$2q^3 + 2q^2 - 10q + 8$	$-q$ products $M(Z)$ -1 convolution q
prime	$(Z^q - 1)/(Z - 1)$	$2q$	$-Z$	$2q \times q$	$4q^3 + 8q^2 - 24q + 16$	$-2q$ products $M(Z)$ -1 convolution $2q$
prime	$(Z^{2q} - 1)/(Z^2 - 1)$	$2q$	$-Z^{q+1}$	$2q \times 2q$	$8q^3 + 16q^2 - 48q + 32$	$-2q$ products $M(Z)$ -1 convolution $2 \times 2q$
q_1^2	$(Z^{q_1^2} - 1)/(Z^{q_1} - 1)$	q_1	Z^{q_1}	$q_1 \times q_1^2$	$2q_1^4 + 4q_1^3 - 8q_1^2 - 2q_1 + 8$	$-q_1$ products $M(Z)$ $-q_1$ products $(Z^{q_1} - 1)/(Z - 1)$ -1 convolution q_1
q_1^2	$(Z^{q_1^2} - 1)/(Z^{q_1} - 1)$	q_1^2	Z	$q_1^2 \times q_1^2$	$4q_1^5 + 2q_1^4 - 10q_1^3 + 2q_1^2 + 12$	$-q_1(q_1 + 1)$ products $M(Z)$ $-q_1$ products $(Z^{q_1} - 1)/(Z - 1)$ -1 convolution q_1
q_1^2	$(Z^{2q_1^2} - 1)/(Z^{2q_1} - 1)$	$2q_1^2$	$-Z$	$2q_1^2 \times 2q_1^2$	$16q_1^5 + 16q_1^4 - 48q_1^3 - 8q_1^2 + 40q_1 + 16$	$-2q_1(q_1 + 1)$ products $M(Z)$ $-2q_1$ products $(Z^{2q_1} - 1)/(Z^2 - 1)$ -1 convolution $2 \times 2q_1$
q_1q_2	$(Z^{q_1q_2} - 1)/(Z^{q_2} - 1)$	q_1	Z^{q_2}	$q_1 \times q_1q_2$	$q_2(2q_1^3 + 2q_1^2 - 10q_1 + 8)$	$-q_1$ products $M(Z)$ -1 convolution q_1q_2
2^t	$Z^{2^t} + 1$	2^{t+1}	Z	$2^{t+1} \times 2^t$ *	$(t + 1)2^{2(t+1)}$	2^{t+1} products $M(Z)$

we have used short convolution algorithms similar to those proposed by Agarwal and Cooley [7, 8] and derived the corresponding algorithms for polynomial multiplication modulo $(Z^q - 1)/(Z - 1)$. These last algorithms are such that $M_2 = M_1 - 1$ and are given in Appendix A for $q = 3, 5, 7$. We give also for comparison in columns 6–7 the number of operations for the Agarwal-Cooley approach. It can be seen that the number of multiplications is approximately half that in the Agarwal-Cooley method while the number of additions is somewhat smaller. An example, illustrating the computation of a convolution of dimension 3×3 by polynomial transforms, is presented in Appendix B.

Generalized polynomial transforms

For the sake of concreteness we have, up to now, restricted our discussion to polynomial transforms having q terms with q prime and defined modulo $(Z^q - 1)/(Z - 1)$. It is apparent from Eqs. (14)–(17) that any polynomial transform of dimension N, having a root aZ^d and defined modulo a polynomial $M(Z)$, has the circular convolution property provided $(aZ^d)^N \equiv 1$ modulo $M(Z)$ and $S \equiv 0$ modulo $M(Z)$ for $t \not\equiv 0$ modulo N, $S \equiv N$ modulo $M(Z)$ for $t \equiv 0$ modulo N, with $S = \sum_{k=0}^{N-1} (aZ^d)^{tk}$. When polynomial transforms are used for computing two-dimensional convolutions, $M(Z)$ must be a factor of $Z^q - 1$, relatively prime with the other factors of $Z^q - 1$. The transform root aZ^d should also be as simple as possible in order to minimize the complexity of transform computation. In spite of these limitations, we show now that there is a very large class of polynomial transforms having the convolution property.

We consider first a polynomial transform having N_1q terms, with q an odd prime, $M(Z) = (Z^q - 1)/(Z - 1)$, and

$$\bar{H}_k(Z) \equiv \sum_{m=0}^{N_1q-1} H_m(Z)(WZ)^{mk} \bmod M(Z)$$

$$W = e^{-j2\Pi/N_1},\ j = \sqrt{-1};$$

$$k = 0, 1, \cdots, N_1q - 1. \qquad (19)$$

Eq. (16) then becomes, with $t = m + r - l$,

$$P_l(Z) \equiv \sum_{m=0}^{N_1q-1} \sum_{r=0}^{N_1q-1} H_{1,m}(Z)X_{1,r}(Z) \frac{1}{N_1q} \sum_{k=0}^{N_1q-1} (WZ)^{tk} \bmod M(Z). \qquad (20)$$

If N_1 and q are relatively prime, k can be replaced by $qv + N_1k$, with $v = 0, 1, \cdots, N_1 - 1$, $k = 0, 1, \cdots, q - 1$, and the sum

$$S = \sum_{k=0}^{N_1q-1} (WZ)^{tk}$$

Table 2 Number of operations for two-dimensional convolutions computed by polynomial transforms.

	Polynomial transform approach				*Agarwal-Cooley method*	
Convolution size	*Total no. of mults.*	*Total no. of adds.*	*Mults. per point*	*Adds. per point*	*Mults. per point*	*Adds. per point*
3 × 3 (9)	13	70	1.44	7.78	1.78	8.56
4 × 4 (16)	22	122	1.37	7.62	1.56	8.44
6 × 3 (18)	26	176	1.44	9.78	1.78	10.56
5 × 5 (25)	55	394	2.20	15.76	4.00	18.60
3 × 9 (27)	58	373	2.15	13.81	2.81	15.67
6 × 6 (36)	52	436	1.44	12.11	1.78	12.56
7 × 7 (49)	121	1177	2.47	24.02	5.22	32.86
5 × 10 (50)	110	888	2.20	17.76	4.00	20.60
8 × 8 (64)	130	750	2.03	11.72	3.06	15.81
9 × 9 (81)	193	1526	2.38	18.84	4.46	28.00
10 × 10 (100)	220	2036	2.20	20.36	4.00	22.60
14 × 14 (196)	484	5632	2.47	28.73	5.22	36.86
18 × 18 (324)	772	7596	2.38	23.44	4.46	32.00
30 × 15 (450)	1430	14644	3.18	32.54	7.11	43.62
30 × 30 (900)	2860	31088	3.18	34.54	7.11	45.62
35 × 35 (1225)	6655	77099	5.43	62.98	20.90	130.03
42 × 42 (1764)	6292	81980	3.57	46.47	9.29	70.97
70 × 35 (2450)	13310	159098	5.43	64.98	20.90	132.03
60 × 60 (3600)	15730	178634	4.37	49.62	11.11	73.47

becomes

$$S \equiv \underbrace{\sum_{r=0}^{N_1-1}}\ W^{tr} \underbrace{\sum_{k=0}^{q-1}}\ Z^{tk}.$$

Thus, $S \not\equiv 0$ only for $t \equiv 0$ modulo $N_1 q$ and (20) reduces to a polynomial convolution of dimension $N_1 q$. Two-dimensional convolutions of dimension $N_1 q \times q$ can thus be computed by using the polynomial transform (19) and computing an auxiliary one-dimensional convolution of dimension $N_1 q$. The main cases of interest are, of course, $N_1 = 2$, $N_1 = 4$, which correspond to convolutions of dimensions $2q \times q$, $4q \times q$ and to transforms with roots $-Z$ and jZ, which can be calculated without multiplications.

If we restrict ourselves to polynomial transforms that can be computed without multiplications, it can be shown by similar considerations that polynomial transforms having the convolution property can be defined modulo $(Z^{2q} - 1)/(Z^2 - 1)$, modulo $(Z^{q_1 q_2} - 1)/(Z^{q_2} - 1)$ with q_1 odd prime and q_2 relatively prime with q_1, and $(Z^{q_1^2} - 1)/(Z^{q_1} - 1)$ with q_1 prime.

The case of convolutions of dimension $2q_1^2 \times 2q_1^2$ computed by polynomial transforms defined modulo $(Z^{2q_1^2} - 1)/(Z^{2q_1} - 1)$ is particularly interesting because the transforms can be computed with a reduced number of additions by means of a three-stage FFT-type algorithm. In this case, the generalized multiplications correspond to $2q_1^2$ polynomial products modulo $(Z^{2q_1^2} - 1)/(Z^{2q_1} - 1)$ plus the auxiliary computation of a convolution of dimension $2q_1^2 \times 2q_1$. This last convolution can in turn be computed by polynomial transforms with $2q_1$ polynomial products modulo $(Z^{2q_1^2} - 1)/(Z^{2q_1} - 1)$ plus an auxiliary convolution of dimension $2q_1 \times 2q_1$. This last convolution can also be computed by a polynomial transform with $2q_1$ polynomial products modulo $(Z^{2q_1} - 1)/(Z^2 - 1)$ plus one auxiliary convolution of dimension $2 \times 2q_1$.

Generalized convolutions defined modulo $Z^q + 1$ in one or both dimensions can also be computed by polynomial transforms. The main cases of interest here are the generalized convolutions of dimension $2q \times q^*$, with q prime, computed by transforms defined modulo $(Z^q + 1)/(Z + 1)$ with root Z and the generalized convolutions of dimensions $2^{t+1} \times 2^t *$ and $2^t * \times 2^t *$ computed by transforms defined modulo $(Z^{2^t} + 1)$. [We use here the symbol * to indicate a generalized convolution modulo $(Z^q + 1)$ as opposed to conventional circular convolutions modulo $(Z^q - 1)$]. It should be noted that for $q = 2^t$, the corresponding transforms have the generalized convolution property modulo $(Z^{2^t} + 1)$ and no auxiliary computation is required. These transforms are also optimum from the computational complexity standpoint since they can be computed with $t + 1$-stage radix-2 FFT-type algorithms. Moreover, when q is even, $Z^q \equiv -1$, and $Z^{q/2} \triangleq j$, with $j = \sqrt{-1}$. It is then possible to take advantage of this property to compute complex multiplications with only two real multiplications by using an approach similar to that proposed by Nussbaumer for Fermat number transforms [15].

We summarize in Table 1 the main existence conditions for polynomial transforms that can be computed without multiplications. It can be seen that these cases correspond to two-dimensional convolutions such that both dimensions have a common factor. We give, in Table 2, the number of operations for two-dimensional convolutions computed by polynomial transforms. It can be seen that the computational savings can be very significant for large convolution sizes. Except for $q = 7$ and $q = 9$, all algorithms for short convolutions and polynomial products are derived from those given by Agarwal and Cooley [7, 8]. For $q = 7$ and $q = 9$, we have used more efficient algorithms in which the polynomial products modulo $(Z^9 - 1)/(Z^3 - 1)$ and modulo $(Z^7 - 1)/(Z - 1)$ are computed in a way similar to that of the polynomial product modulo $(Z^3 - 1)/(Z - 1)$. This algorithm corresponds to 15 multiplications and 51 additions for the polynomial product modulo $(Z^9 - 1)/(Z^3 - 1)$ and to 19 multiplications and 81 additions for the nine-term convolution.

It should now be apparent that NTTs and polynomial transforms bear a strong relationship. In fact, NTTs are particular cases of polynomial transforms in which the q-bit words are to be viewed as polynomials: If we take for instance the polynomial transforms defined modulo $Z^{2^t} + 1$, using 2^{t+1} input polynomials corresponding to 2^{t+1} words of 2^t bits, we obtain the Fermat Number Transform. The existing body of knowledge on NTTs can thus be used as a useful tool to define new polynomial transforms. The main advantage of polynomial transforms over NTTs stems from the fact that all operations are performed modulo a polynomial instead of modulo an integer so that there are no restrictions on arithmetic, word length, or truncation.

Composite algorithms

Suppose now that we want to compute a two-dimensional convolution of dimension $q_1q_2 \times q_1q_2$, where q_1 and q_2 are relatively prime. Using the Chinese remainder theorem, we can view this convolution as a convolution of dimension $(q_1 \times q_1) \times (q_2 \times q_2)$. Under these conditions, this convolution can be computed by a nesting algorithm similar to that proposed in [7, 8]. If M_1, A_1 and M_2, A_2 are the number of multiplications and additions required to compute the convolutions of dimensions $q_1 \times q_1$ and $q_2 \times q_2$, respectively, the total number of multiplications M and of additions A then becomes

$$M = M_1M_2; \tag{21}$$

$$A = A_1q_2^2 + M_1A_2. \tag{22}$$

This computation mechanism can be used recursively to cover the case of more than two factors and is particularly useful when used in conjunction with polynomial transforms, since it allows us to derive efficient algorithms for computing large convolutions from a limited set of small values of q.

We give in the lower part of Table 2 the number of operations for large convolutions computed by polynomial transforms and nesting. It can be seen that for large transforms, the number of additions and multiplications is drastically reduced with respect to the Agarwal-Cooley method. Actual processing load savings depend upon the programming of the algorithm, and particular attention must be paid to implementing efficiently ancillary operations. In this respect, it should be noted that several factors weigh in favor of polynomial transforms. When compared with the FFT algorithm, the polynomial transform approach can be implemented with real arithmetic for the computation of real convolutions and requires no manipulations of trigonometric functions. If the h sequence is fixed, the polynomial transform of h can be precomputed and stored in a read-only memory. In this case, the number of memory locations is reduced with respect to the Agarwal-Cooley method by the same factor as the number of multiplications.

It should also be noted that when a two-dimensional convolution of fixed size must be computed repeatedly, the programming can be divided into two steps, a generation step and an execution step. The programs can then be designed in such a way that most of the bookkeeping operations concerning polynomial manipulations take place within the generation step and therefore do not penalize significantly the execution time.

Computation of one-dimensional convolutions by polynomial transforms

Polynomial transforms permit the computation of two-dimensional transforms such that the lengths in both dimensions are not usually prime. Thus, the efficient two-dimensional to one-dimensional mapping introduced by Agarwal-Cooley [7, 8] is not applicable. It is still possible, however, at the expense of a reduction in computing efficiency, to compute one-dimensional convolutions by a method derived from the overlap-add, overlap-save approaches [10, 16]. We specify here the processing load for one-dimensional convolutions computed by polynomial transforms. To this effect, we consider an N-term convolution:

$$y_e = \sum_{i=0}^{N-1} h_i x_{(e-i) \bmod N} \qquad e = 0, 1, \cdots, N-1. \tag{23}$$

If $N = qN_1$, we can change the indices e and i with

$$e = N_1u + l; \qquad i = N_1n + m$$
$$u, n = 0, 1, \cdots, q-1;$$
$$l, m = 0, 1, \cdots, N_1 - 1; \tag{24}$$

Table 3 Number of operations for one-dimensional convolutions computed by polynomial transforms.

Polynomial transform approach			*Agarwal-Cooley [7, 8]*		
Convolution size	*Mults. per point*	*Adds. per point*	*Convolution size*	*Mults. per point*	*Adds. per point*
240	5.96	61.00	210	7.24	36.03
288	5.87	49.41	360	8.56	44.21
450	6.36	69.08	420	9.05	46.29
800	8.94	83.08	840	12.67	64.47
900	7.94	87.30	840	12.67	64.47
1440	11.74	85.79	1260	16.59	81.86
1800	8.74	99.24	2520	23.22	115.10
2400	11.92	115.85	2520	23.22	115.10

$$y_{N_1u+l} = \sum_{m=0}^{N_1-1} \sum_{n=0}^{q-1} h_{N_1n+m} x_{N_1(u-n)+l-m}. \tag{25}$$

This two-dimensional convolution is circular, of dimension q, for index n and aperiodic, of dimension N_1, for index m. Equation (25) can therefore be computed by circular convolutions of dimension $q \times q$ calculated by polynomial transforms provided $2N_1 \leq q + 1$. In this case, $q - N_1$ of the input polynomials $X_m(Z) = \sum_{n=0}^{q-1} x_{N_1n+m} Z^n$ are null and only N_1 of the output polynomials need to be computed.

When the multidimensional convolution is calculated by polynomial transforms used in conjunction with composite algorithms, accounting for the zero-valued samples in the evaluation of processing load may become difficult. It is sometimes preferable to compute the multidimensional convolution by a single polynomial transform of dimension $q = q_1 q_2, \cdots, q_d$ defined modulo $(Z^q - 1)/(Z^{q_1} - 1)(Z^{q_2^{-1}} + \cdots + 1) \cdots (Z^{q_d^{-1}} + \cdots 1)$, with $q_1, q_2, \cdots, q_d$ relatively prime. This yields about the same processing load as when computation is performed by a composite algorithm and polynomial transforms of dimensions $q_1, q_2, \cdots, q_d$, but the overall organization is much simpler to visualize. We give in Table 3 the number of operations for one-dimensional convolutions computed by polynomial transforms. In this table, we did not account for the zero-valued samples, so that the number of additions given here is an upper bound. It can be seen that for convolution sizes above 200, the number of multiplications is significantly lower than with the Agarwal-Cooley method.

Multidimensional polynomial transforms

Multidimensional polynomial transforms can be defined in a way similar to one-dimensional polynomial transforms. Since we will need these transforms for the computation of Discrete Fourier Transforms (DFTs), we make precise here the case of two-dimensional transforms (three-dimensional convolutions) in order to evaluate the corresponding processing load.

Assuming we want to compute a convolution of dimension $q \times q \times q$, with q prime, we redefine Eqs. (1) to (5) with

$$H_{m_1,m_2}(Z) = \sum_{n=0}^{q-1} h_{n,m_1,m_2} Z^n \qquad m_1, m_2 = 0, 1, \cdots, q-1; \tag{26}$$

$$X_{r_1,r_2}(Z) = \sum_{s=0}^{q-1} x_{s,r_1,r_2} Z^s \qquad r_1, r_2 = 0, 1, \cdots, q-1; \tag{27}$$

$$Y_{l_1,l_2}(Z) \equiv \sum_{m_1=0}^{q-1} \sum_{m_2=0}^{q-1} H_{m_1,m_2}(Z)\, X_{l_1-m_1,l_2-m_2}(Z) \mod (Z^q - 1); \tag{28}$$

$$Y_{l_1,l_2}(Z) = \sum_{u=0}^{q-1} y_{u,l_1,l_2} Z^u \qquad l_1, l_2 = 0, 1, \cdots, q-1; \tag{29}$$

$$y_{u,l_1,l_2} = \sum_{m_1=0}^{q-1} \sum_{m_2=0}^{q-1} \sum_{n=0}^{q-1} h_{n,m_1,m_2} x_{u-n,l_1-m_1,l_2-m_2} \qquad u = 0, 1, \cdots, q-1. \tag{30}$$

We define a two-dimensional transform modulo $M(Z) = (Z^q - 1)/(Z - 1)$ by

$$\tilde{H}_{k_1,k_2}(Z) \equiv \sum_{m_1=0}^{q-1} \sum_{m_2=0}^{q-1} H_{l,m_1,m_2}(Z) Z^{m_1k_1+m_2k_2} \mod M(Z) \qquad k_1, k_2 = 0, 1, \cdots, q-1;$$

$$H_{l,m_1,m_2}(Z) \equiv H_{m_1,m_2}(Z) \mod M(Z), \tag{31}$$

with the corresponding inverse transform. Multiplying

term by term $\tilde{H}_{k_1,k_2}(Z)$ by the transform $\tilde{X}_{k_1,k_2}(Z)$ of $X_{r_1,r_2}(Z)$ and taking the inverse transform yields

$$P_{l_1,l_2}(Z) \equiv \sum_{m_1=0}^{q-1} \sum_{m_2=0}^{q-1} \sum_{r_1=0}^{q-1} \sum_{r_2=0}^{q-1} H_{l,m_1,m_2}(Z) X_{l,r_1,r_2} \times (Z) \frac{1}{q^2} \sum_{k_1=0}^{q_1-1} Z^{(m_1+r_1-l_1)k_1} \sum_{k_2=0}^{q-1} Z^{(m_2+r_2-l_2)k_2} \quad \text{mod } M(Z). \tag{32}$$

Since $Z^q \equiv 1$ modulo $M(Z)$, $P_{l_1,l_2}(Z)$ reduces to

$$P_{l_1,l_2}(Z) \equiv Y_{1,l_1,l_2}(Z) = \sum_{m_1=0}^{q-1} \sum_{m_2=0}^{q-1} H_{1,m_1,m_2}(Z) X_{1,l_1-m_1,l_2-m_2}(Z) \quad \text{mod } M(Z). \tag{33}$$

We compute an auxiliary convolution Y_{2,l_1,l_2} modulo $(Z-1)$ with

$$Y_{2,l_1,l_2} = \sum_{m_1=0}^{q-1} \sum_{m_2=0}^{q-1} H_{2,m_1,m_2} X_{2,l_1-m_1,l_2-m_2}; \tag{34}$$

$$H_{2,m_1,m_2} = \sum_{n=0}^{q-1} h_{n_1,m_1,m_2}; \tag{35}$$

$$X_{2,r_1,r_2} = \sum_{s=0}^{q-1} x_{s,r_1,r_2}. \tag{36}$$

The final convolution product is reconstructed from $Y_{1,l_1,l_2}(Z)$ and Y_{2,l_1,l_2} by using the Chinese remainder theorem

$$Y_{l_1,l_2}(Z) \equiv Y_{1,l_1,l_2}(Z) S_1(Z) + Y_{2,l_1,l_2} S_2(Z) \quad \text{mod } (Z^q - 1), \tag{37}$$

with $S_1(Z)$ and $S_2(Z)$ defined by Eqs. (8) to (11).

Under these conditions, the two-dimensional polynomial transform can be computed with $2q(q^3 - q^2 - 3q + 4)$ additions and the total number of operations for a three-dimensional convolution of dimension $q \times q \times q$ reduces to $4q^4 + 2q^3 - 14q^2 + 6q + 8$ additions, $q(q+1)$ polynomial products modulo $(Z^q - 1)/(Z - 1)$, and one convolution of dimension q.

Using the same approach gives, for a four-dimensional convolution of dimension $q \times q \times q \times q$, a total processing load of $6q^5 + 2q^4 - 20q^3 + 10q^2 + 6q + 8$ additions, $q^3 + q^2 + q$ polynomial multiplications modulo $(Z^q - 1)/(Z - 1)$ and one convolution of dimension q. We give in Table 4 the corresponding number of operations for several multidimensional convolutions used for computing DFTs.

Table 4 Number of operations for short multidimensional convolutions computed by polynomial transforms.

Convolution size	*Total no. of mults.*	*Total no. of adds.*
$3 \times 3 \times 3$	40	325
$3 \times 3 \times 3 \times 3$	121	1324
$6 \times 6 \times 6$	320	3896
$6 \times 6 \times 6 \times 6$	1936	31552

Computation of DFTs by polynomial transforms

Since the polynomial transforms provide an efficient way of computing convolutions, they can be used for the calculations of DFTs. In order to make the corresponding algorithms precise, we consider a DFT of dimension N:

$$A_k = \sum_{n=0}^{N-1} a_n W^{nk} \qquad W = e^{-j2\Pi/N}; \quad k = 0, 1, \cdots, N-1. \tag{38}$$

When $N = N_1$, with N_1 prime, this transform can be computed as a correlation by using Rader's algorithm [17] with

$$A_0 = \sum_{n=0}^{N_1-1} a_n \qquad n \equiv g^u \bmod N_1; \quad k \equiv g^v \bmod N_1; \tag{39}$$

$$A_{g^v} = \sum_{u=0}^{N_1-2} (a_{g^u} - a_0) W^{g^{(u+v)}} \qquad v = 0, 1, \cdots, N_1 - 2, \tag{40}$$

g being a primitive root modulo N_1. Thus, this DFT can be calculated with one multiplication by W^0 and one correlation of dimension $N_1 - 1$. When $N = N_1 N_2$, with N_1, N_2 relatively prime, the DFT of dimension N can be viewed as a DFT of dimension N_1 in which each multiplication is replaced by computing the DFT of N_2 terms [10–12, 18]:

$$A_{N_2 k_1 + N_1 k_2} = \sum_{n_1=0}^{N_1-1} W_{N_1}^{N_2^2 n_1 k_1} \sum_{n_2=0}^{N_2-1} a_{N_2 n_1 + N_1 n_2} W_{N_2}^{N_1^2 n_2 k_2}. \tag{41}$$

Thus, if N_1 and N_2 are primes, the DFT of dimension $N_1 N_2$ can be calculated with one transform of dimension N_2, one correlation of dimension $N_1 - 1$ and one two-dimensional correlation of dimension $(N_1 - 1) \times (N_2 - 1)$. We propose here to compute the transform of dimension N_2 and the correlation of dimension $N_1 - 1$ by Winograd's algorithms and to calculate the correlation of dimension $(N_1 - 1) \times (N_2 - 1)$ by polynomial transforms. This method can of course be extended recursively to DFTs of length $N = N_1 N_2, \cdots, N_i$ provided the various factors N_i are primes. In this case the computation is performed with multidimensional polynomial transforms. Similar results are obtained for factors N_i^d that are powers of primes. In this case, the correlations have a length $(N_i - 1) N_i^{d-1}$.

Since $g^{(N_i-1)/2} \equiv -1$ modulo N_i, the coefficients corresponding to the W^{nk} in the polynomial multiplications are

Table 5 Number of real operations for DFTs computed by polynomial transforms.

DFT size	*Polynomial transform approach*		*Winograd's method*	
	No. of mults.	*No. of adds.*	*No. of mults.*	*No. of adds.*
63	174 (168)	1408	234	1440
504	1392 (1356)	14540	1872	14796
1008	3132 (3084)	34668	4212	35244
2520	8352 (8316)	96364	11232	102348
9×9	242 (224)	1742	338	1936
7×7	138 (136)	1156	162	1152
$7 \times 7 \times 7$	1002 (998)	11820	1458	13896

either real or pure imaginary so that all complex multiplications are implemented with two real multiplications, as with Winograd's algorithms. Moreover, when the factors are powers of primes, additional simplifications are possible. For instance, in the case of a factor N_i^2, N_i prime, the property $\Sigma_{u=0}^{N_i(N_i-1)-1} W^{g^u} = 0$ forces one of the transform terms to be null.

The individual factors $N_1, N_2, \cdots, N_i$ must be chosen in such a way that the multidimensional correlations can be implemented easily with polynomial transforms. A 63-term DFT can, for instance, be computed with five DFTs of seven terms, one correlation of dimension 6 and one correlation of dimension 6×6.

We give in Table 5 the number of operations for various DFTs computed by polynomial transforms. It can be seen that polynomial transforms allow a significant reduction in number of operations when compared to the original Winograd algorithms. The savings are much smaller, however, when the figures are compared to the Winograd method applied to larger fields. In the case of a DFT of $7 \times 7 \times 7$ points, for example, this newer Winograd method [12] requires 1029 real multiplications against 1002 multiplications for the polynomial transform approach. It should be noted, however, that this last Winograd technique requires operation on two sets of data simultaneously and thus may require more storage than the polynomial transform method.

Small additional savings on the number of multiplications can be achieved if "simple" multiplications by ± 1 or $\pm j$ are not counted. The corresponding figures are given in parentheses in Table 5.

Concluding remarks

In this paper we have introduced various polynomial transforms. We have shown that these transforms are computed without multiplications and provide an efficient means of calculating two-dimensional convolutions.

Polynomial transforms can be viewed as generalized NTTs defined in rings of polynomials instead of rings of integers. They have the same computational efficiency as NTTs but they have the advantage over the latter that calculations can be done in ordinary arithmetic, with or without truncation after each arithmetic operation.

When computed by polynomial transforms, two-dimensional convolutions reduce to additions and one-dimensional convolutions. These last convolutions are usually relatively short and can be calculated efficiently by convolution algorithms. In this case, the polynomial transform approach is better than the Agarwal-Cooley approach. It is also better than FFT for convolution sizes up to around 100×100. Moreover, calculations do not require complex arithmetic with sines and cosines and can be carried out, if desired, entirely in integer arithmetic, without roundoff errors.

These results have been extended to cover the case of one-dimensional convolutions computed by the two-dimensional to one-dimensional mapping introduced by Agarwal and Burrus. We have shown that, in spite of the loss of efficiency incurred in this mapping, the polynomial transform method yields a significant reduction in number of multiplications over the Agarwal-Cooley algorithm for convolution sizes above 200.

Polynomial transforms can also be used for the computation of DFTs, and are in some cases better than FFT or WFTA methods for this application.

Polynomial transforms are well adapted for implementing digital filtering processes in general purpose computers. It is expected that these techniques will be particularly useful for signal processing applications having large computation requirements such as image processing.

Appendix A: Algorithms for short polynomial products modulo $(Z^q - 1)/(Z - 1)$

- A1 *Polynomial product modulo* $(Z^3 - 1)/(Z - 1)$:
  3 multiplications
  3 additions

$a_0 = h_0 - h_1 \qquad b_0 = x_1$
$a_1 = h_0 \qquad b_1 = x_0 - x_1$
$a_2 = h_1 \qquad b_2 = x_0$

$$m_k = a_k b_k \qquad k = 0, 1, 2$$

$$y_0 = m_0 + m_1$$
$$y_1 = m_0 + m_2$$

Corresponding convolution of dimension 3:
4 multiplications
11 additions

• A2 *Polynomial product modulo* $(Z^5 - 1)/(Z - 1)$:
9 multiplications
21 additions

$$a_0 = x_0 \qquad b_0 = h_0 - h_2 + h_3$$
$$a_1 = x_1 \qquad b_1 = h_1 - h_2 + h_3$$
$$a_2 = x_0 + x_1 \qquad b_2 = (-2h_0 - 2h_1 + 3h_2 - 2h_3)/5$$
$$a_3 = x_2 \qquad b_3 = -h_0 + h_1 - h_2 + h_3$$
$$a_4 = x_3 \qquad b_4 = -h_0 + h_1 - h_2$$
$$a_5 = x_2 + x_3 \qquad b_5 = (3h_0 - 2h_1 + 3h_2 - 2h_3)/5$$
$$a_6 = a_0 - a_3 \qquad b_6 = -h_2 + h_3$$
$$a_7 = a_1 - a_4 \qquad b_7 = h_1 - h_2$$
$$a_8 = a_2 - a_5 \qquad b_8 = (-h_0 - h_1 + 4h_2 - h_3)/5$$

$$m_k = a_k b_k \qquad k = 0, 1, \cdots, 8$$

$$u_0 = m_3 + m_0$$
$$u_1 = m_4 + m_1$$
$$u_2 = m_5 + m_2$$
$$u_3 = m_6 - m_0$$
$$u_4 = m_7 - m_1$$
$$u_5 = m_8 - m_2$$
$$u_6 = u_4 + u_5$$
$$u_7 = u_6 + u_6$$
$$u_8 = -u_3 + u_4$$
$$u_9 = u_7 - u_8$$
$$u_{10} = u_1 + u_2$$
$$u_{11} = u_0 + u_2$$
$$u_{12} = -u_{10} - u_{11}$$

$$y_0 = u_8$$
$$y_1 = u_6 + u_{12}$$
$$y_2 = u_7 + u_{10}$$
$$y_3 = u_9 + u_{11}$$

Corresponding convolution of dimension 5:
10 multiplications
31 additions

• A3 *Polynomial product modulo* $(Z^7 - 1)/(Z - 1)$:
15 multiplications
55 additions

$$f_0 = (-2h_5 + 3h_4 - h_3 - 2h_2 + h_1 + 2h_0)/2$$
$$f_1 = (3h_5 - 11h_4 + 10h_3 + 3h_2 - 11h_1 - 4h_0)/14$$
$$f_2 = (3h_5 - h_4 - 2h_3 + 3h_2 - h_1)/6$$
$$f_3 = (h_4 - h_3 + h_1)/6$$
$$f_4 = -h_5 - 2h_4 + 3h_3 - h_2 - 2h_1 + h_0$$
$$f_5 = (h_5 + 2h_4 - h_3 - 2h_2 + 3h_1 - h_0)/2$$
$$f_6 = (-11h_5 - 4h_4 + 10h_3 + 3h_2 - 11h_1 + 10h_0)/14$$
$$f_7 = (-h_5 - 2h_3 + 3h_2 - h_1 - 2h_0)/6$$
$$f_8 = (h_5 - h_3 + h_1 - h_0)/6$$
$$f_9 = -2h_5 + h_4 + 2h_3 - h_2 - 2h_1 + 3h_0$$
$$f_{10} = (2h_4 - h_3 - 2h_2 + h_1)/2$$
$$f_{11} = (-2h_5 - 2h_4 + 12h_3 + 5h_2 - 9h_1 - 2h_0)/14$$
$$f_{12} = (-2h_3 + 3h_2 - h_1)/6$$
$$f_{13} = (-h_3 + h_1)/6$$
$$f_{14} = 2h_3 - h_2 - 2h_1 + h_0$$

$$a_0 = x_0 \qquad a_1 = x_1 \qquad a_2 = x_2$$
$$a_3 = x_3 \qquad a_4 = x_4 \qquad a_5 = x_5$$
$$a_6 = a_1 + a_2 \qquad a_7 = -a_1 + a_2 \qquad a_8 = a_4 + a_5$$
$$a_9 = -a_4 + a_5$$

$$e_0 = a_0$$
$$e_1 = a_0 + a_6$$
$$e_2 = a_0 + a_7$$
$$e_3 = e_1 + a_7 + a_6 + a_6$$
$$e_4 = a_2$$
$$e_5 = a_3$$
$$e_6 = a_3 + a_8$$
$$e_7 = a_3 + a_9$$
$$e_8 = e_6 + a_9 + a_8 + a_8$$
$$e_9 = a_5$$
$$e_{10} = e_5 - e_0$$
$$e_{11} = e_6 - e_1$$
$$e_{12} = e_7 - e_2$$
$$e_{13} = e_8 - e_3$$
$$e_{14} = e_9 - e_4$$

$$m_k = e_k f_k \qquad k = 0, 1, \cdots, 14$$

$$u_0 = m_5 + m_0$$
$$u_1 = m_6 + m_1$$
$$u_2 = m_7 + m_2$$
$$u_3 = m_8 + m_3$$
$$u_4 = m_9 + m_4$$
$$u_5 = m_{10} + m_0$$
$$u_6 = m_{11} + m_1$$
$$u_7 = m_{12} + m_2$$
$$u_8 = m_{13} + m_3$$
$$u_9 = m_{14} + m_4 \qquad y_0 = u_{18}$$
$$u_{10} = u_1 + u_3 \qquad y_1 = u_{16} + u_{19}$$
$$u_{11} = u_{10} + u_2 \qquad y_2 = u_{15} + u_{24}$$
$$u_{12} = u_0 + u_{11} \qquad y_3 = u_{14} + u_{22}$$
$$u_{13} = u_{10} + u_3 \qquad y_4 = u_{12} + u_{23}$$
$$u_{14} = u_{13} - u_2 \qquad y_5 = u_{20}$$
$$u_{15} = u_{13} + u_3 + u_3 + u_4 + u_2$$
$$u_{16} = -u_{12} - u_{15} - u_{14}$$
$$u_{17} = u_7 - u_8$$
$$u_{18} = u_5 + u_{17} + u_7$$
$$u_{19} = u_{17} - u_6 - u_8$$
$$u_{20} = (u_7 + u_8) + (u_7 + u_8) + u_9$$
$$u_{22} = u_{19} + u_{19}$$
$$u_{23} = u_{22} - u_{18}$$
$$u_{24} = u_{22} - u_{20}$$

Corresponding convolution of dimension 7:
16 multiplications
70 additions

Appendix B: Example of a convolution of dimension 3 × 3 computed by polynomial transforms

This example illustrates the computation of a convolution of dimension 3 × 3 evaluated by polynomial transforms as shown in the second and third sections.

$q = 3 \qquad M(Z) = Z^2 + Z + 1 \qquad Z^3 \equiv 1$

Input sequences:

$h_{i,0} = (4, 3, 0) \qquad x_{i,0} = (2, 0, 2)$
$h_{i,1} = (4, 3, 1) \qquad x_{i,1} = (0, 1, 3)$
$h_{i,2} = (2, 1, 0) \qquad x_{i,2} = (3, 4, 4)$

Reduction modulo $M(Z)$:

$H_{1,n}(Z)$: (4, 3) $\qquad X_{1,r}(Z)$: (0, −2)
(3, 2) (−3, −2)
(2, 1) (−1, 0)

Polynomial transforms:

$\check{H}_k(Z)$: (9, 6) $\qquad \check{X}_k(Z)$: (−4, −4)
(1, 2) (3, −2)
(2, 1) (1, 0)

$1/(Z - 1) \equiv -(Z + 2)/3 \bmod M(Z)$
$\check{H}_k(Z)/(Z - 1) \bmod M(Z)$:
−(4, 5)
−(0, 1)
−(1, 1)

Polynomial multiplications: (using algorithm given in Appendix A1)

$\check{X}_k(Z)\check{H}_k(Z)/3(Z - 1) \bmod M(Z)$: (−4, 16)/3
(−2, −5)/3
(−1, −1)/3

Inverse polynomial transform:

$Y_{1,l}/(Z - 1) \bmod M(Z)$: (−7, 10)/3
(−6, 18)/3
(1, 20)/3

Reduction modulo $(Z - 1)$:

$H_{2,m}$: (7, 8, 3) $\qquad Q_{2,r}$: (4, 4, 11)

Convolution of dimension 3:

$Y_{2,l}/3$: (128, 93, 121)/3

Chinese remainder reconstruction: [Eq. (18)]

$y_{u,l}$: (45, 37, 46, 33, 23, 37, 40, 34, 47)

References

1. J. W. Cooley and J. W. Tuckey, "An Algorithm for Machine Calculation of Complex Fourier Series," *Math. Comput.* **19,** 297 (1966).
2. J. M. Pollard, "The Fast Fourier Transform in a Finite Field," *Math. Comput.* **25,** 365 (1971).
3. C. M. Rader, "Discrete Convolution via Mersenne Transforms," *IEEE Trans. Computers* **C-21,** 1269 (1972).
4. R. C. Agarwal and C. S. Burrus, "Number Theoretic Transforms to Implement Fast Digital Convolution," *Proc. IEEE* **63,** 550 (1975).
5. H. J. Nussbaumer, "Digital Filtering Using Complex Mersenne Transforms," *IBM J. Res. Develop.* **20,** 498 (1976).
6. C. M. Rader, "On the Application of the Number Theoretic Methods of High-Speed Convolution to Two-dimensional Filtering," *IEEE Trans. Acoust. Speech Signal Processing* **ASSP-23,** 575 (1975).
7. R. C. Agarwal and J. W. Cooley, "New Algorithms for Digital Convolution," *Research Report RC-6446,* IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1977.
8. R. C. Agarwal and J. W. Cooley, "New Algorithms for Digital Convolution," *Proc. 1977 Intl. Conf., Acoust., Speech and Signal Processing* (Hartford) 360 (1977).
9. H. J. Nussbaumer, "Digital Filtering using Polynomial Transforms," *Electron. Lett.* **13,** 386 (1977).
10. R. C. Agarwal and C. S. Burrus, "Fast One-dimensional Digital Convolution by Multidimensional Techniques," *IEEE Trans. Acoust. Speech Signal Processing* **ASSP-22,** 1 (1974).
11. S. Winograd, "On Computing the Discrete Fourier Transform," *Proc. Nat. Acad. Sci. USA* **73,** 1005 (1976).
12. S. Winograd, "A New Method for Computing DFT," *Proc. 1977 Intl. Conf., Acoust., Speech and Signal Processing* (Hartford) 366 (1977).
13. H. F. Silverman, "A Method for Programming the Complex General-*N* Winograd Fourier Transform Algorithm," *Proc. 1977 Intl. Conf., Acoust., Speech and Signal Processing* (Hartford) 369 (1977).
14. T. Nagell, *Introduction to Number Theory,* Chelsea Publishing Co., New York, 1964, Ch. 5, p. 156.
15. H. J. Nussbaumer, "Complex Convolutions via Fermat Number Transforms," *IBM J. Res. Develop.* **20,** 282 (1976).
16. B. Gold, C. M. Rader, A. V. Oppenheim, and T. G. Stockham, *Digital Processing of Signals,* McGraw-Hill Book Co., Inc., New York, 1969, Ch. 7, p. 203.
17. C. M. Rader, "Discrete Fourier Transforms When the Number of Data Samples is Prime," *Proc. IEEE* **56,** 1107 (1968).
18. I. J. Good, "The Relationship Between Two Fast Fourier Transforms," *IEEE Trans. Computers* **20,** 310 (1971).

Received June 6, 1977; revised October 18, 1977

H. J. Nussbaumer is located at Compagnie IBM France, Centre d'Etudes et Recherches, 06610 La Gaude, France; P. Quandalle is located at the University of Nice, France.