FAST ALGORITHMS FOR REDUCTION A MODULO POLYNOMIAL AND VANDERMONDE TRANSFORM USING FFT*

ALEXANDER M. KROT, HELENA B.MINERVINA

Institute of Engineering Cybernetics, National Academy of Sciences of Belarus,
Surganov Str. 6, 220012, Minsk, Belarus, alxkrot@newman.basnet.minsk.by

Abstract. This paper shows on how the real algorithms for the reduction a modulo arbitrary polynomial and fast Vandermonde transform (FVT) are realized on computer using fast Fourier transform (FFT). This real-valued FVT algorithm on the developed fast reduction polynomial algorithm is based. The realization of FVT algorithm on computer with real multiplicative complexity $O(2N\log_2^2 N)$ and real additive complexity $O(6N\log_2^2 N)$ is obtained. New FVT algorithm is applied in digital signal, filtering and interpolation problems.

Key Words. Computational complexity, reduction a modulo, linear and cyclic convolutions, fast Fourier transform, fast Vandermonde transform.

1. INTRODUCTION

The theoretical-minimal estimations of multiplicative complexity for the reduction a modulo arbitrary polynomial and FVT were obtained in [1]. In this report fast algorithms of the reduction a modulo arbitrary polynomial and FVT according the scheme [1] are realized on the computer using FFT. The obtained real estimations of computational complexity are higher than the theoretical ones from [1] because no each algorithm realizable on computer is able to reach theoretical minimal estimation for this class of problems. At the same time the real algorithms realized on computer and useful for filtering and interpolation of digital signals and images are proposed in this paper.

2. THE ALGORITHM AND COMPUTATIONAL COMPLEXITY OF THE REDUCTION A MODULO ARBITRARY POLYNOMIAL

The algorithm leading to estimations of Lemma 1 [1] presents below. Let *R(z)* " *A(z)mod q(z)* be a polynomial residue modulo q(z) over V. According to the Euclid theorem,

$$A(z) = B(z)q(z) + R(z) \qquad (1)$$

where $A(z) = \sum_{i=0}^{L} a_i z^i$, $q(z) = z^N - \sum_{i=1}^{N} q_i z^{N-i}$,

$B(z) = \sum_{i=0}^{L-N} b_i z^i$, $R(z) = \sum_{i=0}^{N-1} r_i z^i$

with a_i, q_i, b_i, r_i ¸ *V*. The first term in (1), is the linear convolution (LC) of the sequences $\{b_0, b_1, ..., b_{L-N}\}$, and $\{-q_N, -q_{N-1}, ..., -q_1, 1\}$ [2]. For clarity, let us write (1) in the matrix form:

$$\begin{bmatrix} a_L \\ a_{L-1} \\ a_{L-2} \\ \vdots \\ a_N \\ a_{N-1} \\ \vdots \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 1 & & & 0 \\ -q_1 & 1 & & \\ -q_2 & -q_1 & & \\ \vdots & \vdots & & \\ -q_{L-N} & -q_{L-N-1} & \cdots & 1 \\ & & & \\ -q_{L-N+1} & -q_{L-N} & \cdots & -q_1 \\ & & & \vdots \\ & & \ddots & -q_{N-1} \\ 0 & & & -q_N \end{bmatrix} \cdot \begin{bmatrix} b_{L-N} \\ b_{L-N-1} \\ b_{L-N-2} \\ \vdots \\ b_0 \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ r_{N-1} \\ \vdots \\ r_1 \\ r_0 \end{bmatrix} \qquad (2)$$

It follows from (1), that to determine the coefficients of the residue R(z) described by the vector $\vec{r} = (r_{N-1}, \ldots, r_1, r_0)^T$, first we have to find the coefficients of the incomplete quotient B(z), that is, $\vec{b} = (b_{L-N}, \ldots, b_{L-N-1}, b_0)^T$. If

* This work was supported by International Science and Technology Center (ISTC) under project B-95

L=2N±1 or L=2N, this may be done by inverting the lower triangular matrix with unit diagonal:

$$\begin{bmatrix} b_{L-N} \\ b_{L-N-1} \\ b_{L-N-2} \\ \vdots \\ b_0 \end{bmatrix} = \begin{bmatrix} 1 & & & & 0 \\ -q_1 & 1 & & & \\ -q_2 & -q_1 & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ -q_{L-N} & -q_{L-N-1} & -q_{L-N-2} & \cdots & 1 \end{bmatrix}^{-1} \times$$

$$\times \begin{bmatrix} a_L \\ a_{L-1} \\ a_{L-2} \\ \vdots \\ a_N \end{bmatrix} \qquad (3)$$

Indeed, for *L = 2N-1*, the LC submatrix under consideration becomes *N x N* lower triangular matrix and, therefore invertible. As shown in [3], the matrix inverse of any lower triangular *NxN* matrix with a unit diagonal can be written as the product of *N-1* elementary lower-column matrices.

Because *2N = L*, i.e. *L-N = N*, and the elements q_i *(i=1,...,N-1)* are defined, the inverse matrix in (3) can be computed in advance according to [3]. It is a lower triangular matrix with a unit diagonal and the upper submatrix of the LC matrix of sequences $\{a_N, ..., a_{L-1}, a_L\}$ and $\left\{-q_N^{(-1)}, \ldots, -q_1^{(-1)}, 1\right\}$, where $-q_i^{(-1)}$, *i = 1, 2,..., N*, are elements of the inverse matrix.

Using the FFT algorithms [2], we can determine the minimal number of multiplications to compute the LC-product of two polynomials $\tilde{B}(z) = \tilde{A}(z)q^{(-1)}(z)$, where

$$\tilde{B}(z) = z^N \sum_{i=0}^{L-N} b_i z^i + \sum_{i=0}^{N-1} \tilde{b}_i z^i, \quad \tilde{A}(z) = \sum_{i=0}^{L-N} a_{i+N} z^i, \qquad \text{and}$$

$$q^{(-1)}(z) = z^N - \sum_{i=1}^{N} q_i^{(-1)} z^{N-i}.$$

Thus, the polynomial *B(z)* (which is a part of the polynomial $\tilde{B}(z) = z^N B(z) + \sum_{i=0}^{N-1} \tilde{b}_i z^i$) can be calculated using FFTs. According to (1), the determination of the residue R(z) can be reduced to computing the LC as well:

$$R(z) = A(z) - B(z)q(z).$$

The LC described by the product of the polynomials *W(z) = B(z)q(z)* is also calculated using the FFT algorithms.

Note, however, that as long as the polynomials $q^{(-1)}(z)$ and *q(z)* are normalized, $\tilde{B}(z)$ and *W(z)* may be calculated as follows:

$$\tilde{B}(z) = \tilde{A}(z)\left[z^N - \sum_{i=1}^{N} q_i^{(-1)} z^{N-i}\right] = \qquad (4)$$

$$= z^N \tilde{A}(z) - \tilde{A}(z)\sum_{i=1}^{N} q_i^{(-1)} z^{N-i}$$

$$W(z) = B(z)\left[z^N - \sum_{i=1}^{N} q_i z^{N-i}\right] = \qquad (5)$$

$$= z^N B(z) - B(z)\sum_{i=1}^{N} q_i z^{N-1}$$

The products $z^N \tilde{A}(z) \stackrel{\Delta}{=} \tilde{B}_1(z)$ and $z^N B(z) \stackrel{\Delta}{=} W_1(z)$ can be computed by a shift without any multiplication, so the computation of (4) and (5) is reduced to the computation of the following polynomial products:

$$\tilde{B}_2(z) = \tilde{A}(z)\sum_{i=1}^{N} q_i^{(-1)} z^{N-i}, \qquad (6)$$

$$W_2(z) = B(z)\sum_{i=1}^{N} q_i z^{N-i}, \qquad (7)$$

where $\deg \tilde{A}(z) = \deg B(z) = L - N$ and

$$deg \sum_{i=1}^{N} q_i^{(-1)} z^{N-i} = deg \sum_{i=1}^{N} q_i z^{N-i} = N - 1.$$

In practical applications of this algorithm to compute polynomial residues, long LCs are computed using FFTs, [4]-[8].

Let us estimate the number of multiplications for the case when N is not precisely equal to *L/2* (or *(L–1)/2* for odd *L*). In Lemma 2 [1] the estimation of multiplications for this case was derived. In this report the algorithm based on Lemma 2 will be carried out by analogy with the one based on Lemma 1.

Let us obtain the arithmetic complexity for such algorithm. The fast procedures for LC are used for the implementation of this algorithm. Minimum multiplicative complexity algorithms for computing the LC were obtained only for small lengths *(N=2,3,4)* [2], [9]. Therefore, it would be reasonable to compute the LC of two N-point sequences $\{x_n\}$ and $\{h_n\}$ *(n=0,1,...,N-1)* as the cyclic convolution (CC) of 2*N*-point sequences $\{\tilde{x}_n\} = \{x_0, ..., x_{N-1}, 0, ..., 0\}$ and $\{\tilde{h}_n\} = \{h_0, ..., h_{N-1}, 0, ..., 0\}$ (it is obvious that LC of two *(N-1)*-point or two (N-2)-point sequences may be also computed by *2N*-point CC). In its turn, the set of algorithms for computing with the minimum multiplicative complexity

of the CC is confined to small lengths *N£9* [2], [10]. The lengths of *N*-point CC may be increased by the Agarwal-Cooley algorithm [2],[10] if $N = \prod_i N_i$ and N_i are relatively prime numbers, or by the recursive nesting Nussbaumer algorithms [2] if N = 2^{α} . The recent computation algorithms for the CC, built around the improved FFT called split-radix FFT (SRFFT) [5]-[8], are more efficient.

Let us use the estimate for the arithmetic complexity of an N-point real CC (N = 2^{α}) obtained in [7],[11]. We assume that V = R is the field of real numbers. The computation of the N-point of sequences $\{x_n\}$ and $\{h_n\}$ over R requires the following number of real multiplications M(N) and real additions A(N):

$$M(N) = N\left(\log_2 N - 3/2\right) + 3; \quad (8)$$

$$A(N) = N\left(3\log_2 N - 7/2\right) + 5. \quad (9)$$

It follows from (8) and (9), that the number of multiplications and additions to compute the LC of N-point sequences through the CC of 2N-point sequences can be estimated as

$$M(N) = N\left(2\log_2 N - 1\right) + 3; \quad (10)$$

$$A(N) = N\left(6\log_2 N - 1\right) + 5. \quad (11)$$

According to the algorithm based on Lemma 2 the reduction of the polynomial *A(z)* modulo *q(z)* over *R*, where *deg A(z)= L= 2N-2* and *degq(z)=N*, amounts to the computation of LC of (N-2)-point sequences ($\tilde{B}(z) = \tilde{A}(z)q^{(-1)}(z)$) and LC of *N*-point sequences (*W(z) = B(z)q(z)*) and to *N* subtraction (*R(z)=A(z)-W(z)*). The algorithm for *2N*-point CC may be used for the computation of each LC. By virtue of (10) and (11), the numbers of real multiplications and additions for the computation of R(z) ≡ A(z)modq(z) are

$$M(N,2N-2) = 2N\left(2\log_2 N - 1\right) + 6; \quad (12)$$

$$A(N,2N-2) = N + 2N\left(6\log_2 N - 1\right) + 10 = \\ = N\left(12\log_2 N - 1\right) + 10 \quad (13)$$

where *M(N,2N-2)* and *A(N,2N-2)* are functions depending both on *N* and *L = 2N-2*.

3. THE ALGORITHM AND COMPUTATIONAL COMPLEXITY OF FAST VANDERMONDE TRANSFORM

Let us note that the computation of the Vandermonde transform of the sequence *{x_n}, n = 0,..., N-1*, reduces to the computation of the polynomial $X(z) = \sum_{n=0}^{N-1} x_n z^n$ at the points $z = \lambda_0, \lambda_1, \ldots, \lambda_{N-1}$. Stated differently, we need to determine the polynomial residues *X(z)* to *mod(z-*λ_0*)*, *mod(z-*λ_1*),..., mod(z-*λ_{N-1}*)*. The usual consecutive division of *X(z)* by each polynomial results in the multiplicative complexity *O(N^2)*.

To construct a more efficient algorithm, we will use the following property of polynomial residues:

$$X_k = X(\lambda_k) = X(z) \bmod (z - \lambda_k) = \\ = \left[X(z) \bmod S(z)\right] \bmod (z - \lambda_k) \quad (14)$$

where *S(z)* is the divisor of *q(z)* and *z-*λ_k is the divisor of *S(z)*.

Theorem 2 from [1] proves the existence of a fast Vandermonde transform (FVT). To construct a FVT algorithm, we use tree-structured computation stages of the dichotomous method [12] and then we adapt the algorithm based on Lemma 2. This means that at the preprocessing stage the reduction polynomials *S(z)* are computed as follows: all linear reduction polynomials $z-\lambda_k$ are pairwise multiplied to determine second-degree reduction polynomials; next, the *N/2* second-degree reduction polynomials are pairwise multiplied and fourth-degree reduction polynomials are obtained, and so on until two reduction polynomials of degree *N/2* are obtained. The computation of polynomial residues *X(z)modS(z)* with a view to the determination of the polynomial residues *Xk" X(z)mod(z-*λ *k)* according to (14) is realized by the inverse order of computing stages. In the first stage, the residues modulo the reduction polynomials $\prod_{k=0}^{N/2-1}(z - \lambda_k)$ and $\prod_{k=N/2}^{N-1}(z - \lambda_k)$, are computed, i.e.,

$$X_0^{(1)}(z) \equiv X(z) \bmod \prod_{k=0}^{N/2-1}(z - \lambda_k) \text{ and}$$

$$X_1^{(1)}(z) \equiv X(z) \bmod \prod_{k=N/2}^{N-1}(z - \lambda_k)$$

At the second stage

$$X_0^{(2)}(z) \equiv X_0^{(1)}(z) \bmod \prod_{k=0}^{N/4-1}(z - \lambda_k)$$

$$X_1^{(2)}(z) \equiv X_0^{(1)}(z) \bmod \prod_{k=N/4}^{N/2-1}(z - \lambda_k)$$

$$X_2^{(2)}(z) \equiv X_1^{(1)}(z) \bmod \prod_{k=N/2}^{3N/4-1}(z-\lambda_k) \text{ and}$$

$$X_3^{(2)}(z) \equiv X_1^{(1)}(z) \bmod \prod_{k=3N/4}^{N-1}(z-\lambda_k)$$

are computed, and so forth. At the last stage a, the desired values of the polynomials at the points λ_k are determined:

$$X_0 \equiv X_0^{(\alpha-1)}(z) \bmod (z-\lambda_0),$$
$$X_{N-1} \equiv X_{N/2-1}^{(\alpha-1)}(z) \bmod (z-\lambda_{N-1}).$$

We assume that $V=R$ and the roots $\lambda_0, \lambda_1,..., \lambda_{N-1}$, R are known.

Taking into account these considerations, estimate the total number of arithmetic operations involved in the N-point FVT algorithm:

$$M(N) = \sum_{i=1}^{a}\left[2N\left(2\log_2 N - 2i - 1\right) + 6\cdot 2^i\right] =$$

$$= 4\alpha N \log_2 N - \left(2\sum_{i=1}^{\alpha} i + \alpha\right)2N + 6\sum_{i=1}^{\alpha} 2^i =$$

$$= 2N\log_2^2 N - 4N\log_2 N + 12(N-1); \qquad (15)$$

$$A(N) = \sum_{i=1}^{a}\left[N\left(12\log_2 N - 2i - 1\right) + 10\cdot 2^i\right] =$$

$$= 6N\log_2^2 N - 7N\log_2 N + 20(N-1). \qquad (16)$$

As it follows from (15) the multiplicative complexity of the proposed real-valued FVT algorithm is $O(2N\log_2^2 N)$ which is equivalent to the one of the Aho-Hopcroft-Ulman algorithm [12].

4. CONCLUSION

The comparison of (15) with Theorem 2 from [1] shows that for large N, the real-valued FVT algorithm requires $1/2\log_2 N$ times more multiplications than does the theoretically achievable multiplicative complexity $O(4N\log_2 N)$. The FVT algorithm is much more efficient than the computation of the polynomial $X(z)$ at N arbitrary points $z = \lambda_0, \lambda_1,..., \lambda_{N-1}$ by the N-fold application of the Gorner scheme [4]. This means that FVT algorithm can be conveniently used for constructing fast Lagrange interpolation procedures for an arbitrary choice of points $\lambda_0, \lambda_1,..., \lambda_{N-1}$. The interpolation procedures here are well-posed from the numerical point of view, because the proposed FVT algorithm makes repeated use of the FFT algorithm.

5. REFERENCES

1.A.M.Krot, "The multiplicative complexity of the reduction a modulo arbitrary polynomial, generalized K_N-convolution and fast Vandermonde transform" Proc. 13th Intern. Conf. on Digital Signal Proc. (DSP'97), vol.2, Santorini, Greece, pp. 893-897, 1997.

2.H.J.Nussbaumer, Fast Fourier Transform and Convolution Algorithms. Berlin, Heidelberg, and New York: Springer,1981.

3.S.Pissanetzky. Space Matrix Technology. London: Academic Press, 1984.

4.D.E.Knuth. The Art of Computer Programming. Reading, MA. Addison-Wesley, vol.2 (Seminumerical algorithms), 1969.

5.P.Duhamel and H.Hollmann, "Split-radix FFT algorithm", Electron. Lett., vol.20, no. 1, pp. 14-16, 1984.

6.A.M.Krot and H.B. Minervina, "Fast Fourier transform algorithms for real and Hermitian-symmetrical sequences", Soviet J.Comm. Tech. Electron., vol.34, no.12, pp. 122-129, 1989 (a translation of Radiotekhnika i Elektronika, vol.34, no. 2, pp. 369-376, 1989).

7.A.M.Krot, "The method of eigentransforms in different fields for computing cyclic convolutions and discrete Fourier transforms", U.S.S.R. Comput. Math. and Math. Phys., vol. 29, no. 3, pp. 23-34, 1989 (a translation of Zh. Vychisl. Mat. i Mat. Fiz., vol. 29, no. 5, p.p. 675-692, 1989).

8.A.M.Krot and H.B. Minervina, "Comment. Conjugate pair fast Fourier transform", Electron. Letters, vol. 28, no. 12, pp. 1143-1144, 1992.

9.H.Krishna. Computation Complexity of Bilinear Forms, Berlin: Springer, 1987.

10.J.H. McClellan and C.M. Rader, Number Theory in Digital Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1979.

11. A.M.Krot, Discrete Models of Dynamic Systems Based on the Polynomial Algebra. Minsk: Nauka i Tekhnika, 1990 (in Russian).

12.A.Aho, J.Hopcroft and J. Ulman, The Design and Analysis of Computer Algorithms. Reading, MA: Addison-Wesley,1974.