# On the Additive Complexity of the Cyclotomic FFT Algorithm

Peter Trifonov
Distributed Computing and Networking Department
Saint-Petersburg State Polytechnic University
Polytechnicheskaya str., 21, office 104
194021, Saint-Petersburg, Russia
Email: petert@dcn.ftk.spbstu.ru

***Abstract*—The problem of efficient evaluation of the discrete Fourier transform over finite fields is considered. The techniques for additive complexity reduction of the cyclotomic FFT algorithm are proposed. The first one is based on the classical simultaneous reduction algorithm. The second one is based on a factorization of the presummation matrix into a sparse and block-diagonal ones. The proposed methods provide smaller asymptotic complexity, although for small-sized problems the required number of operations appears to be higher than the complexity of computer-optimized algorithms.**

## I. Introduction

The discrete Fourier transform over finite fields is extensively used in various channel coding and cryptographic applications. Classical FFT algorithms developed for the case of complex field are not well-suited for the case of finite fields. More efficient algorithms can be derived by exploiting the algebraic properties of the finite fields.

The cyclotomic fast Fourier transform [1], [2], [3], [4], [5], [6], [7], [8] was shown to provide the smallest complexity for small binary finite fields. Fast decoding algorithms for Reed-Solomon codes were constructed using this approach [8], [2]. However, its asymptotic complexity is dominated by the cost of multiplication of the input vector by a binary matrix. Computer search was used to derive optimized sequences of operations for solving this problem [9], [7], [4], but the complexity of the obtained algorithms grows rapidly with the dimension of FFT. Furthermore, the obtained algorithms do not appear to have any structure and may be difficult to implement in hardware.

In this paper a relationship between the presummation step of the cyclotomic FFT and classical simultaneous polynomial modulo reduction problem is established. This enables one to employ the corresponding fast algorithms, reducing thus the overall asymptotic additive complexity. Furthermore, a relationship between the pre-summation matrix and generator matrices of some cyclic codes is described. This enables one to construct a more structured implementation of the pre-summation step.

The paper is organized as follows. Section II presents the necessary background. The proposed asymptotically fast algorithm is described in Section III. Section IV introduces a sparse factorization of the pre-summation matrix of the cyclotomic FFT. Finally, some conclusions are drawn.

## II. Cyclotomic FFT

The discrete Fourier transform of the vector $(f_0, \ldots, f_{n-1})$ (or, equivalently, a polynomial $f(x) = \sum_{i=0}^{n-1} f_i x^i$) over $\mathbb{F}_{p^m}$ is given by

$$F_j = f(\omega^j) = \sum_{i=0}^{n-1} f_i \omega^{ij}, 0 \leq j < n, \tag{1}$$

where $p$ is a prime number, $n|(p^m - 1)$ and $\omega \in \mathbb{F}_{p^m}$ is a primitive $n$-th root of unity. The cyclotomic coset modulo $n$ is given by $C_c = \left\{s_c, s_c p, s_c p^2, \ldots, s_c p^{m_c - 1}\right\}$, where $s_c p^{m_c} \equiv s_c \bmod n$. It can be seen that the set of integers $\{0, 1 \ldots, n-1\}$ can be partitioned into a number of cyclotomic cosets $C_1, \ldots, C_l$. Let $s_1, \ldots, s_l$ be the generating elements of these cosets.

The inverse cyclotomic FFT algorithm can be obtained by re-ordering the components of the output vector according to the cyclotomic cosets [2], [5]:

$$F_{s_c p^j} = \sum_{i=0}^{n-1} f_i \omega^{i s_c p^j}. \tag{2}$$

Let $\left\{\gamma_c, \gamma_c^p, \ldots, \gamma_c^{p^{m_c-1}}\right\}$ be a normal basis of $\mathbb{F}_{p^{m_c}} \subset \mathbb{F}_{p^m}$, where $m_c | m$. Since $\omega^{s_c}$ is a root of $x^{p^{m_c}} - x = 0$, one obtains that $\omega^{i s_c} \in \mathbb{F}_{p^{m_c}}$, i.e. it can be expressed as $\omega^{i s_c} = \sum_{t=0}^{m_c-1} a_{cti} \gamma_c^{p^t}$. Hence, (2) can be rewritten as

$$F_{s_c p^j} = f(\omega^{s_c p^j}) = \sum_{i=0}^{n-1} \sum_{t=0}^{m_c-1} f_i a_{cti} \gamma_c^{p^{j+t}}. \tag{3}$$

This expression can be represented as

$$F = LAf, \tag{4}$$

where $A$ is a binary matrix (the presummation matrix) corresponding to the coefficients $a_{cti}$, and $L$ is a block-diagonal matrix, with blocks given by

$$L_c = \begin{pmatrix} \gamma_c & \gamma_c^p & \cdots & \gamma_c^{p^{m_c-1}} \\ \gamma_c^p & \gamma_c^{p^2} & \cdots & \gamma_c \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_c^{p^{m_c-1}} & \gamma_c & \cdots & \gamma_c^{p^{m_c-2}} \end{pmatrix}.$$

Computing a product of $L_c$ and a vector is equivalent to computing a cyclic convolution. By employing the appropriate fast algorithms, one can dramatically decrease the total number of multiplications needed to compute the DFT. However, computing the product $Af$ requires one either to employ the "Four Russians" algorithm [10] with complexity $O(n^2/\log n)$, or to construct an optimized algorithm by extensive computer search [9], [7], [4]. No fast algorithms exploiting the structure of matrix $A$ were reported up to now.

## III. Cyclotomic FFT via multiple modulo reduction

It is a well known fact that the elements $\omega^{s_c}, \omega^{s_c p}, \ldots, \omega^{s_c p^{m_c-1}}$ represent a conjugacy class over $\mathbb{F}_{p^m}$, and their minimal polynomial is given by

$$m_c(x) = \prod_{j=0}^{m_c-1} (x - \omega^{s_c p^j}).$$

Hence, (3) can be rewritten as

$$F_{s_c p^j} = f(\omega^{s_c p^j}) = r_c(\omega^{s_c p^j}), 0 \le j < m_c \qquad (5)$$

where $r_c(x) \equiv f(x) \bmod m_c(x)$. This represents an instance of the Goertzel algorithm [11]. Having obtained $r_c(x)$, one can compute $F_{s_c p^j}$ using the above described inverse cyclotomic algorithm. Computing $r_c(x)$ requires one to perform operations only in the prime field $\mathbb{F}_p$. In matrix notation, this can be represented as

$$F = LA'A''f, \qquad (6)$$

where computing $A''f$ is equivalent to reduction of $f(x)$ modulo $m_c(x), 1 \le c \le l$, and $A'$ is a block-diagonal matrix with the blocks given by

$$A'_c = ((a_{cti})), 0 \le t, i < m_c.$$

Multiplication by $A'_c$ requires $O(m_c^2) = O(\log^2 n)$ operations in $\mathbb{F}_p$. However, these operations can be combined with the presummation step of the cyclic convolution algorithm used to implement multiplication by $L_c$.

Fast simultaneous reduction algorithm can be used for computing $r_c(x)$ [12]. Namely, a tree[1] of polynomials $M_{i,j}(x), 0 \le j < 2^i, 0 \le i \le \lfloor \log l \rfloor$, can be constructed, so that $M_{i+1,j}(x) = M_{i,2j}(x) M_{i,2j+1}(x)$, and the leaves of the tree correspond to irreducible polynomials $m_c(x)$. Then

$$\begin{aligned} r_{i,2j}(x) &\equiv r_{i,j}(x) \bmod M_{i,2j}(x) \\ r_{i,2j+1}(x) &\equiv r_{i,j}(x) \bmod M_{i,2j+1}(x), \end{aligned}$$

where $r_{0,0}(x) = f(x)$, and $r_c(x)$ are obtained at the leaves of the corresponding tree. The cost of this algorithm is given by $R(n) = \sum_{i=1}^{\lfloor \log_2 l \rfloor} \sum_{j=0}^{2^i-1} 2D(\deg M_{i,j}(x)) \le 2D(n) \lfloor \log l \rfloor$, where $D(\mu)$ is the complexity of division of polynomial $a(x)$ of degree $2\mu$ by polynomial $b(x)$ of degree $\mu$. A fast algorithm implementing this operation can be obtained from the expression

$$\begin{aligned} \underbrace{x^{\mu_{i,j/2}-1} r_{i,j/2}(\frac{1}{x})}_{r'_{i,j/2}(x)} &= \underbrace{x^{\mu_{i,j}} M_{i,j}(\frac{1}{x})}_{M'_{i,j}(x)} \underbrace{x^{\mu_{i,j/2}-1-\mu_{i,j}} q_{i,j}(\frac{1}{x})}_{q'_{i,j}(x)} + \\ & \quad x^{\mu_{i,j/2}-\mu_{i,j}} \underbrace{x^{\mu_{i,j}-1} r_{i,j}(\frac{1}{x})}_{r'_{i,j}(x)}, \end{aligned}$$

where $\deg r'_{i,j}(x) < \deg M_{i,j}(x) = \mu_{i,j}(x)$. It can be seen that [12]

$$q'_{i,j}(x) \equiv (M'_{i,j}(x))^{-1} r'_{i,j/2}(x) \bmod x^{\mu_{i,j/2}-\mu_{i,j}}. \qquad (7)$$

The polynomials $(M'_{i,j}(x))^{-1} \bmod x^{\mu_{i,j/2}-\mu_{i,j}}$ can be precomputed. Multiplication by these polynomials requires only operations in $\mathbb{F}_p$ (only summations in the case of $p = 2$). Hence, $D(\mu) = 2M(\mu) + \mu$, where $M(\mu)$ is the complexity of multiplying two polynomials of degree $\mu$. Assuming that most of the cyclotomic cosets have size $m_c \approx \log_p n$, one obtains that $l \approx n/\log_p n$, and the cost of the presummation step of the cyclotomic FFT algorithm is $O(\frac{n}{\log n}\log^2(n) + M(n)\log n) = O(M(n)\log n)$.

However, the fast polynomial division algorithm becomes useful only for very large $n$. In most practical cases more efficient implementation can be obtained by re-ordering the polynomials $m_c(x)$ so that some polynomials $M_{i,j}(x)$ become sparse. In this case the division can be implemented using the standard algorithm. Similar approach was considered in [13], where $M_{i,j}(x)$ were enforced to be affine polynomials with coefficients in $\mathbb{F}_{p^m}$. The method proposed in this paper relies on polynomials in $\mathbb{F}_p[x]$.

**Example 1.** *Consider the* 15*-point DFT of* $f(x) = \sum_{i=0}^{14} f_i x^i$ *over* $\mathbb{F}_{2^4}$. *Let* $\omega$ *be the primitive root of* $x^4 + x + 1$. *The generators of normal bases of* $F_2$, $F_4$, *and* $F_{16}$ *are* 1, $\omega^5$, *and* $\omega^3$, *respectively. Let*[2] $M_{0,0}(x) = x^8 + x^4 + x^2 + x, M_{0,1}(x) = x^8 + x^4 + x^2 + x + 1$. *It can be seen that constructing the decomposition* $f(x) = q_{0,0}(x)M_{0,0}(x) + r_{0,0}(x), \deg r_{0,0}(x) < \deg M_{0,0}(x)$ *requires* 21 *summation operations. Since* $M_{0,1}(x) = M_{0,0}(x) + 1$, *one obtains* $r_{0,1}(x) = r_{0,0}(x) + q_{0,0}(x) \equiv f(x) \bmod M_{0,1}(x)$. *This requires* 7 *summations. Let* $M_{1,0}(x) = x^4+x, M_{1,1}(x) = x^4+x+1, M_{1,2}(x) = x^4+x^3+1, M_{1,3}(x) = x^4+x^3+x^2+x+1$. *The decomposition* $r_{0,0}(x) = q_{1,0}(x)M_{1,0}(x)+r_{1,1}(x)$ *can be constructed using 4 summations, and computing* $r_{1,1}(x) = r_{1,0}(x) + q_{1,0}(x)$ *also requires 4 operations. Computing* $r_{1,2}(x) \equiv r_{1,0}(x) \bmod M_{1,2}(x)$ *requires* 8 *summations. To compute* $r_{1,3}(x) \equiv r_{1,0}(x) \bmod M_{1,3}(x)$ *one can first calculate* $r'_{1,3}(x) \equiv r_{1,0}(x) \bmod (x^5 + 1)$ *in* 3 *summations, and then obtain* $r_{1,3}(x) \equiv r'_{1,3}(x) \bmod x^4 + x^3 + x^2 + x + 1$ *in 4 summations. Furthermore, let* $M_{2,0}(x) = x^2 + x, M_{2,1}(x) = x^2 + x + 1$. *Division by* $M_{2,0}(x)$ *requires* 2 *operations, and computing* $r_{2,1}(x) = r_{2,0}(x) + q_{2,0}(x) \equiv r_{1,0} \bmod M_{2,1}(x)$

[1] This tree may not be balanced, since there exist minimal polynomials of different degrees.

[2] The straightforward implementation would use $M_{0,0}(x) = x^7 + x^3 + x + 1$. However, this would not allow one to re-use the results of the division.

*also requires* 2 *operations. Finally, computing* $r_{3,0}(x) \equiv r_{2,0}(x) \bmod x+1$ *requires* 1 *operation. Hence, the cost of computing* $A''f$ *is* 56 *operations. Matrix* $A'$ *consists of the following blocks:* $A'_0 = (1)$, $A'_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$, $A'_3 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$, $A'_5 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$, $A'_7 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$. *By combining these matrices with the presummation matrices of cyclic convolution algorithms given in [11], one obtains the following expressions:*

$$
\begin{aligned}
F_0 &= r_{3,0,0} \\
\begin{pmatrix} F_5 \\ F_{10} \end{pmatrix} &= \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \left( \begin{pmatrix} \omega^5 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} r_{2,1,0} \\ r_{2,1,1} \end{pmatrix} \right) \\
\begin{pmatrix} F_1 \\ F_2 \\ F_4 \\ F_8 \end{pmatrix} &= P_4 \left( \begin{pmatrix} \omega^{10} \\ \omega^9 \\ \omega^8 \\ 1 \\ \omega^{10} \\ 1 \\ \omega^4 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{1,1,0} \\ r_{1,1,1} \\ r_{1,1,2} \\ r_{1,1,3} \end{pmatrix} \right) \\
\begin{pmatrix} F_3 \\ F_6 \\ F_{12} \\ F_9 \end{pmatrix} &= P_4 \left( \begin{pmatrix} \omega^{10} \\ \omega^9 \\ \omega^8 \\ 1 \\ \omega^{10} \\ 1 \\ \omega^4 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} r_{1,3,0} \\ r_{1,3,1} \\ r_{1,3,2} \\ r_{1,3,3} \end{pmatrix} \right) \\
\begin{pmatrix} F_7 \\ F_{14} \\ F_{13} \\ F_{11} \end{pmatrix} &= P_4 \left( \begin{pmatrix} \omega^{10} \\ \omega^9 \\ \omega^8 \\ 1 \\ \omega^{10} \\ 1 \\ \omega^4 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} r_{1,2,0} \\ r_{1,2,1} \\ r_{1,2,2} \\ r_{1,2,3} \end{pmatrix} \right)
\end{aligned}
$$

*where* $P_4 = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$, *and* $\cdot$ *denotes componentwise product of two vectors. Evaluating these expressions requires* $1, 4, 4, 5$ *additions for the pre-summation step, and* $2, 9, 9, 9$ *additions for the post-summation step, respectively. Hence, the overall complexity of the algorithm is 99 operations. This is higher than the complexity of the computer-optimized algorithm reported in [1].*

## IV. Sparse factorization of the presummation matrix

The implementation of the presummation step based on fast polynomial division may be quite impractical for small values of $n$. On the other hand, the computer-optimized sequence of operations for evaluating $Af$ may be completely unstructured, and thus not well-suited for a practical implementation. In this section we consider an alternative way to factor the presummation matrix $A$, which results in more regular sequnces of operations.

By construction, matrix $A$ consists of the blocks given by

$$
A_c = \begin{pmatrix} a_{c,0,0} & a_{c,0,1} & \dots & a_{c,0,n-1} \\ a_{c,1,0} & a_{c,1,1} & \dots & a_{c,1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{c,m_c-1,0} & a_{c,m_c-1,1} & \dots & a_{c,m_c-1,n-1} \end{pmatrix},
$$

where $(a_{c,0,i}, \dots, a_{c,m_c-1,i})$ represents a binary image of $\omega^{s_c i}$. This is similar to the construction of the parity check matrix of cyclic codes. Namely, any collection of such blocks given by $\Delta = \{c_1, \dots, c_u\}$ can be considered as a parity check matrix of some $p$-ary cyclic code $\mathcal{C}_\Delta$ of length $n$, having a generator polynomial $g_\Delta(x) : g_\Delta(\omega^{s_{c_i}}) = 0, c_i \in \Delta$. Let $\{b_1, \dots, b_k\} \subset \mathbb{F}_p^n$ be a minimum weight basis of the dual of this code (i.e. a set of linearly independent vectors generating the row space of $A_\Delta = \begin{pmatrix} A_{c_1} \\ \vdots \\ A_{c_u} \end{pmatrix}$, such that the sum of Hamming weights of these vectors is minimal), where $k = \sum_{i=1}^{u} m_{c_i}$. The elements of this basis can be represented as a $u \times n$ matrix $B_\Delta$. Then $A_\Delta = U_\Delta B_\Delta$ for some non-singular $u \times u$ $p$-ary matrix $U_\Delta$. Consider a partitioning of the set of cyclotomic coset generators $\{s_1, \dots, s_l\} = \Delta_1 \cup \Delta_2 \cup \dots \cup \Delta_v$. Then one obtains a factorization

$$
A = \begin{pmatrix} U_{\Delta_1} & 0 & \dots & 0 \\ 0 & U_{\Delta_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & U_{\Delta_v} \end{pmatrix} \underbrace{\begin{pmatrix} B_{\Delta_1} \\ B_{\Delta_2} \\ \vdots \\ B_{\Delta_v} \end{pmatrix}}_{B} \tag{8}
$$

Observe that the dual of code $\mathcal{C}_\Delta$ is also a cyclic code $\mathcal{C}'_\Delta$. A simple, also not always optimal, way to obtain $B_{\Delta_i}$ is to use the classical construction of generator matrices of the corresponding cyclic codes, i.e.

$$
B_{\Delta_i} = \begin{pmatrix} g_{\Delta_i,0} & g_{\Delta_i,1} & \dots & g_{\Delta_i,n-k_i} & \dots & 0 \\ 0 & g_{\Delta_i,0} & \dots & g_{\Delta_i,n-k_i-1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & & \dots & & g_{\Delta_i,n-k_i} \end{pmatrix}
$$

where $g_{\Delta_i}(x) = \sum_{j=0}^{n-k_i} g_{\Delta_i,j} x^i$ is a generator polynomial of $\mathcal{C}'_\Delta$, and $k_i$ is its dimension.

By construction, matrix $B$ is sparse, so an efficient algorithm for computing $Bf$ can be derived immediately from it.

Its complexity can be estimated as $O(\sum_{i=1}^{v} d_i k_i)$, where $d_i$ is the minimum distance of the corresponding cyclic codes.

Matrices $U_{\Delta_i}$ are dense, so one still needs to employ computer search to derive fast algorithms for multiplication by them [9], [7], [4]. However, the dimension of these matrices is smaller than the one of the original matrix $A$. This results in smaller complexity of fast algorithm construction. In some cases, this may also result in better optimization quality.

**Example 2.** *Consider* 15*-point cyclotomic FFT algorithm over* $\mathbb{F}_{16}$ *as shown in Figure 1, where* $\gamma_2 = \omega^5$ *and* $\gamma_4 = \omega^3$ *are normal basis generators of* $\mathbb{F}_{2^2}$ *and* $\mathbb{F}_{2^4}$*, respectively, and* $\omega$ *is the primitive root of* $x^4 + x + 1$*. The kernel of the DFT is given by* $\omega = \omega$*. It is possible to construct an algorithm for computing* $Af$*, which requires 47 summations. It appears that the first 7 rows of* $A$ *represent a generator matrix for a cyclic code with generator polynomial* $g_1(x) = 1+x^4+x^6+x^7+x^8$*, while the last 8 ones correspond to a cyclic code generated by* $g_2(x) = 1 + x^4 + x^6 + x^7$*. Hence, one obtains a factorization*

$$A = \begin{pmatrix} U_1 & 0 \\ 0 & U_2 \end{pmatrix} \underbrace{\begin{pmatrix} 1&0&0&0&1&0&1&1&1&0&0&0&0&0&0 \\ 0&1&0&0&0&1&0&1&1&1&0&0&0&0&0 \\ 0&0&1&0&0&0&1&0&1&1&1&0&0&0&0 \\ 0&0&0&1&0&0&0&1&0&1&1&1&0&0&0 \\ 0&0&0&0&1&0&0&0&1&0&1&1&1&0&0 \\ 0&0&0&0&0&1&0&0&0&1&0&1&1&1&0 \\ 0&0&0&0&0&0&1&0&0&0&1&0&1&1&1 \\ 1&0&0&0&1&0&1&1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&1&0&1&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&1&0&1&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&1&0&1&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0&1&0&1&1&0&0&0 \\ 0&0&0&0&0&1&0&0&0&1&0&1&1&0&0 \\ 0&0&0&0&0&0&1&0&0&0&1&0&1&1&0 \\ 0&0&0&0&0&0&0&1&0&0&0&1&0&1&1 \end{pmatrix}}_{B},$$

*where* $U_1 = \begin{pmatrix} 1&1&1&1&0&0&1 \\ 1&1&0&1&0&1&0 \\ 1&0&1&1&1&1&1 \\ 1&1&1&1&1&1&0 \\ 1&0&1&0&0&1&1 \\ 1&0&0&0&0&0&1 \\ 1&1&0&0&1&0&1 \end{pmatrix}$ *and* $U_2 = \begin{pmatrix} 1&1&0&0&1&0&0&0 \\ 1&0&1&0&1&1&0&0 \\ 1&0&0&0&0&1&1&1 \\ 1&0&0&1&1&1&1&0 \\ 1&1&0&0&0&1&1&0 \\ 1&1&1&1&1&0&0&0 \\ 1&1&1&0&0&1&1&1 \\ 1&0&1&0&0&1&0&1 \end{pmatrix}$*. It can be seen that the straightforward multiplication by last* 8 *rows of* $B$ *requires 24 summations. Adding* $f_8, \ldots, f_{14}$ *to the result of this operation provides, one obtains a product of first* 7 *rows of* $B$ *and vector* $f$*. Computer optimization can be used to obtain algorithms for multiplication by* $U_1$ *and* $U_2$*, which require* 12 *and* 14 *summations, respectively. Hence, the total complexity of the algorithm for computing* $Af$ *based on factorization* (8) *is 57 summations. While this value is higher than the complexity of the computer-optimized algorithm obtained for the original matrix* $A$*, the proposed approach results in a more structured sequence of operations, which is more convenient for implementation.*

## V. Conclusions

In this paper the problem of efficient implementation of the presummation step of the cyclotomic FFT algorithm was investigated. It was shown that this step can be implemented via the classical simultaneous modulo reduction algorithm, which is based on fast polynomial division and multiplication operations. This results in the reduction of the asymptotic number of summations from $O(n^2/\log n)$ (the implementation based on the "Four Russians" algorithm) to $O(M(n)\log n)$, where $M(n)$ is the complexity of multiplying two polynomials of degree $n$. Similar approach was used in [14] in order to reduce the complexity of multiplication by $L_c$.

Furthermore, it was shown that the presummation matrix can be factored into a block-diagonal matrix and a matrix, consisting of sparse generator matrices of some cyclic codes with small minimum distance. This factorization enables one to obtain a more structured implementation of the presummation algorithm. Furthermore, the dimension of matrices, which require computer optimization for construction of a fast multiplication algorithm, is reduced.


## Acknowledgements

This work was supported by the grant MK-1976.2011.9 of the President of Russia. The author thanks Prof. S. V. Fedorenko for many stimulating discussions and for pointing a few errors in the original version of this paper.


## References


[1] P. V. Trifonov and S. V. Fedorenko, "A method for fast computation of the Fourier transform over a finite field," *Problems of Information Transmission*, vol. 39, no. 3, pp. 231–238, July-September 2003, translation of Problemy Peredachi Informatsii.

[2] E. Costa, S. V. Fedorenko, and P. V. Trifonov, "On computing the syndrome polynomial in Reed-Solomon decoder," *European Transactions on Telecommunications*, vol. 15, no. 4, pp. 337–342, May/June 2004.

[3] S. Fedorenko, "A method for computation of the discrete Fourier transform over a finite field," *Problems of information transmission*, vol. 42, no. 2, pp. 139–151, 2006.

[4] S. Bellini, M. Ferrari, and A. Tomasoni, "On the structure of cyclotomic Fourier transforms and their applications to Reed-Solomon codes," *IEEE Transactions on Communications*, vol. 59, no. 8, pp. 2110 –2118, August 2011.

[5] S. Fedorenko, "The discrete Fourier transform over a finite field with reduced multiplicative complexity," in *Proceedings of IEEE International Symposium on Information Theory*, 2011, pp. 1200 –1204.

[6] X. Wu and Z. Yan, "Computational complexity of cyclotomic fast Fourier transforms over characteristic-2 fields," in *IEEE Workshop on Signal Processing Systems (SiPS)*, October 2011, pp. 1 –6.

[7] N. Chen and Z. Yan, "Cyclotomic FFTs with reduced additive complexities based on a novel common subexpression elimination algorithm," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1010–1020, 2009.

[8] ——, "Reduced-complexity Reed-Solomon decoders based on cyclotomic FFTs," *IEEE Signal Processing Letters*, vol. 16, no. 4, pp. 279–282, 2009.

$$
\begin{pmatrix} F_0 \\ F_5 \\ F_{10} \\ F_1 \\ F_2 \\ F_4 \\ F_8 \\ F_3 \\ F_6 \\ F_{12} \\ F_9 \\ F_7 \\ F_{14} \\ F_{13} \\ F_{11} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma_2 & \gamma_2^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma_2^2 & \gamma_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma_4 & \gamma_4^2 & \gamma_4^4 & \gamma_4^8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma_4^2 & \gamma_4^4 & \gamma_4^8 & \gamma_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma_4^4 & \gamma_4^8 & \gamma_4 & \gamma_4^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma_4^8 & \gamma_4 & \gamma_4^2 & \gamma_4^4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_4 & \gamma_4^2 & \gamma_4^4 & \gamma_4^8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_4^2 & \gamma_4^4 & \gamma_4^8 & \gamma_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_4^4 & \gamma_4^8 & \gamma_4 & \gamma_4^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_4^8 & \gamma_4 & \gamma_4^2 & \gamma_4^4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_4 & \gamma_4^2 & \gamma_4^4 & \gamma_4^8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_4^2 & \gamma_4^4 & \gamma_4^8 & \gamma_4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_4^4 & \gamma_4^8 & \gamma_4 & \gamma_4^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_4^8 & \gamma_4 & \gamma_4^2 & \gamma_4^4 \end{pmatrix} \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}}_{A} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \\ f_{10} \\ f_{11} \\ f_{12} \\ f_{13} \\ f_{14} \end{pmatrix}
$$

Fig. 1. 15-point cyclotomic FFT


[9] P. Trifonov, "Matrix-vector multiplication via erasure decoding," in *Proceedings of XI International Symposium on Problems of Redundancy in Information and Control Systems*, 2007.

[10] A. Aho, J. Hopcroft, and J. Ullman, *The design and analysis of computer algorithms*. Addison-Wesley, 1976.

[11] R. Blahut, *Theory and Practice of Error Control Codes*. Addison-Wesley, 1984.

[12] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*. Cambridge University Press, 1999.

[13] Y. Wang and X. Zhu, "A fast algorithm for the Fourier transform over finite fields and its VLSI implementation," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 3, pp. 572–577, 1988.

[14] S. V. Fedorenko, "A novel method for computation of the discrete Fourier transform over characteristic two finite field of even extension degree," *IEEE Transactions on Information Theory*, 2011, submitted.