# A Method for Fast Computation of the Fourier Transform over a Finite Field

**P. V. Trifonov and S. V. Fedorenko**

*St. Petersburg State Polytechnic University*

**Abstract**—We consider the problem of fast computation of the Fourier transform over a finite field by decomposing an arbitrary polynomial into a sum of linearized polynomials. Examples of algorithms for the Fourier transform with complexity less than that of the best known analogs are given.

## 1. INTRODUCTION

Presently, a number of fast Fourier transform (FFT) algorithms over the real or complex field is known, but translation of these algorithms to finite fields is not always possible. Furthermore, an FFT algorithm specially constructed for a particular finite field may be better than an algorithm translated from another field [1].

The suggested method consists in decomposing an original polynomial into a sum of linearized polynomials (1) and evaluating them at a set of basis points (2). Components of the Fourier transform are computed as linear combinations of these values with coefficients from a prime field (3).

An approach based on representing a polynomial as a sum of linearized ones was first suggested in [2] and then generalized in [3]. In what follows, we consider basic notions and definitions, introduce the cyclotomic decomposition of polynomials, and present an FFT algorithm based on this decomposition. The algorithm is described for fields of characteristic 2 but can be generalized for the case of an arbitrary finite field.

## 2. BASIC NOTIONS AND DEFINITIONS

**Definition 1.** The *Fourier transform* of a polynomial $f(x) = \sum_{i=0}^{n-1} f_i x^i$ of degree $\deg f(x) = n-1$, $n \mid (2^m - 1)$, in the field $GF(2^m)$ is the collection of elements

$$F_j = f(\alpha^j) = \sum_{i=0}^{n-1} f_i \alpha^{ij}, \quad j \in [0, n-1],$$

where $\alpha$ is an element of order $n$ in the field $GF(2^m)$.

**Definition 2.** A *linearized polynomial* over $GF(2^m)$ is a polynomial of the form

$$L(x) = \sum_i \ell_i x^{2^i}, \quad \ell_i \in GF(2^m).$$

One can easily show that a linearized polynomial satisfies the equality $L(a+b) = L(a) + L(b)$. A consequence of this property is Theorem 1, which is presented here in a modified form.

**Theorem** [4, 7]. *Let $x \in GF(2^m)$ and let elements $\beta_0, \beta_1, \ldots, \beta_{m-1}$ form a basis of the field.*

$$\textit{If} \quad x = \sum_{i=0}^{m-1} x_i \beta_i, \quad x_i \in GF(2), \quad \textit{then} \quad L(x) = \sum_{i=0}^{m-1} x_i L(\beta_i).$$

Consider the set of cyclotomic cosets modulo $n$ over $GF(2)$:

$$\{0\},\ \{k_1, k_1 2, k_1 2^2, \ldots, k_1 2^{m_1-1}\},\ \ldots,\ \{k_\ell, k_\ell 2, k_\ell 2^2, \ldots, k_\ell 2^{m_\ell-1}\},$$

where $k_i \equiv k_i 2^{m_i} \bmod n$.

The polynomial $f(x) = \sum\limits_{i=0}^{n-1} f_i x^i$, $f_i \in GF(2^m)$, can be decomposed as

$$f(x) = \sum_{i=0}^{l} L_i(x^{k_i}), \quad L_i(y) = \sum_{j=0}^{m_i-1} f_{k_i 2^j \bmod n}\, y^{2^j}. \tag{1}$$

Indeed, expression (1) is a way of grouping the numbers $s \in [0, n-1]$ with respect to cyclotomic cosets: $s \equiv k_i 2^j \bmod n$. Obviously, such a decomposition always exists. Note that, for $k_i = 0$, the free term $f_0$ can be written as the value of the polynomial $L_0(y) = f_0 y$ at $y = x^0$.

We will call (1) the cyclotomic decomposition of $f(x)$.

*Example 1.* The polynomial $f(x) = \sum\limits_{i=0}^{6} f_i x^i$, $f_i \in GF(2^3)$, can be represented as

$$\begin{aligned} f(x) &= L_0(x^0) + L_1(x) + L_2(x^3); \\ L_0(y) &= f_0 y, \\ L_1(y) &= f_1 y + f_2 y^2 + f_4 y^4, \\ L_2(y) &= f_3 y + f_6 y^2 + f_5 y^4. \end{aligned}$$

## 3. FAST COMPUTATION OF THE FOURIER TRANSFORM

According to decomposition (1), let us write $f(\alpha^j) = \sum\limits_{i=0}^{l} L_i(\alpha^{jk_i})$. As is well known [5], the element $\alpha^{k_i}$ is a root of the corresponding minimal polynomial of degree $m_i$ and hence lies in the subfield $GF(2^{m_i})$, $m_i \mid m$. Thus, all elements $(\alpha^{k_i})^j$ lie in the field $GF(2^{m_i})$ and can be decomposed with respect to some basis $(\beta_{i,0}, \ldots, \beta_{i,m_i-1})$ of this field: $\alpha^{jk_i} = \sum\limits_{s=0}^{m_i-1} a_{ijs}\beta_{i,s}$, $a_{ijs} \in GF(2)$. Then each of the linearized polynomials can be evaluated at the basis points of the corresponding subfield by the formula

$$L_i(\beta_{i,s}) = \sum_{p=0}^{m_i-1} \beta_{i,s}^{2^p} f_{k_i 2^p}, \quad i \in [0, \ell], \quad s \in [0, m_i - 1]. \tag{2}$$

Bases $(\beta_{i,0}, \ldots, \beta_{i,m_i-1})$ for each of the linearized polynomials $L_i(y)$ can be chosen independently.

According to the theorem, components of the Fourier transform of a polynomial $f(x)$ are linear combinations of these values:

$$F_j = f(\alpha^j) = \sum_{i=0}^{\ell} \sum_{s=0}^{m_i-1} a_{ijs} L_i(\beta_{i,s}) = \sum_{i=0}^{\ell} \sum_{s=0}^{m_i-1} a_{ijs} \left( \sum_{p=0}^{m_i-1} \beta_{i,s}^{2^p} f_{k_i 2^p} \right), \quad j \in [0, n-1]. \tag{3}$$

The last expression can be written in the matrix form as $F = ALf$, where $F = (F_0, F_1, \ldots, F_{n-1})^T$, $f = (f_0, f_{k_1}, f_{k_1 2}, f_{k_1 2^2}, \ldots, f_{k_1 2^{m_1-1}}, \ldots, f_{k_\ell}, f_{k_\ell 2}, f_{k_\ell 2^2}, \ldots, f_{k_\ell 2^{m_\ell-1}})^T$ is a permutation of the coefficient vector of the original polynomial $f(x)$ corresponding to decomposition (1), $A$ is the matrix

composed of the elements $a_{ijs} \in GF(2)$, and $L$ is the block diagonal matrix composed of the elements $\beta_{i,s}^{2^p}$. It is clear that, for linearized polynomials of the same degree $m_i$ that enter decomposition (1), it is possible to choose the same bases $(\beta_{i,s})$ in the subfields $GF(2^{m_i})$, so that the matrix $L$ will contain a large number of equal blocks.

Thus, the FFT problem is divided into two stages: multiplying the block diagonal matrix $L$ by the original vector $f$ and multiplying the binary matrix $A$ by the vector $S = Lf$ obtained:

$$F = ALf. \tag{4}$$

Let us consider in more detail the first stage of the Fourier transform, the problem of computing $S = Lf$. The block diagonal matrix

$$L = \begin{pmatrix} L_0 & 0 & \dots & 0 \\ 0 & L_1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & L_\ell \end{pmatrix}$$

consists of the blocks

$$L_i = \begin{pmatrix} \beta_{i,0} & \beta_{i,0}^2 & \dots & \beta_{i,0}^{2^{m_i-1}} \\ \beta_{i,1} & \beta_{i,1}^2 & \dots & \beta_{i,1}^{2^{m_i-1}} \\ \dots & \dots & \dots & \dots \\ \beta_{i,m_i-1} & \beta_{i,m_i-1}^2 & \dots & \beta_{i,m_i-1}^{2^{m_i-1}} \end{pmatrix}.$$

An example of using the normal basis as $(\beta_{i,0}, \dots, \beta_{i,m_i-1})$ is considered in [6]. In the case of the normal basis $\left(\gamma_i, \gamma_i^2, \dots, \gamma_i^{2^{m_i-1}}\right)$, the matrix $L$ consists of blocks of the form

$$L_i = \begin{pmatrix} \gamma_i^{2^0} & \gamma_i^2 & \dots & \gamma_i^{2^{m_i-1}} \\ \gamma_i^2 & \gamma_i^4 & \dots & \gamma_i^{2^0} \\ \dots & \dots & \dots & \dots \\ \gamma_i^{2^{m_i-1}} & \gamma_i^{2^0} & \dots & \gamma_i^{2^{m_i-2}} \end{pmatrix}.$$

Due to the block diagonal structure of $L$, evaluation of the product $S = Lf$ can be represented as $S = (b_0, b_1, \dots, b_\ell)^T = L\,(a_0, a_1, \dots, a_\ell)^T$, where $b_i = (b_{i,0}, b_{i,1}, \dots, b_{i,m_i-1})$ are subvectors of the vector $S$ sought for and $a_i = (a_{i,0}, a_{i,1}, \dots, a_{i,m_i-1})$ are subvectors of the original vector $f$.

Let us represent the computation of $b_i^T = L_i a_i^T$ as a cyclic convolution

$$\begin{aligned} b_i(x) &= b_{i,0} + b_{i,m_i-1}x + \dots + b_{i,1}x^{m_i-1} \\ &= \left(\gamma_i + \gamma_i^{2^{m_i-1}}x + \dots + \gamma_i^2 x^{m_i-1}\right)\left(a_{i,0} + a_{i,1}x + \dots + a_{i,m_i-1}x^{m_i-1}\right) \mod (x^{m_i} - 1). \end{aligned}$$

For its evaluation, known algorithms can be applied [1, 7, 8]. Here, using the normal basis property $\gamma_i + \gamma_i^2 + \dots + \gamma_i^{2^{m_i-1}} = 1$ considerably reduces the number of operations in the computation of the cyclic convolution. Note that evaluation of linearized polynomials with the use of cyclic convolution was described in the monograph [7].

The approach described reduces the problem of multiplying the block diagonal matrix $L$ by the original vector $f$ over $GF(2^m)$ to that of computing $\ell + 1$ cyclic convolutions of a small length $m_i$. Known algorithms for computation of cyclic convolutions $b_i(x) = \gamma_i(x)a_i(x) \bmod (x^{m_i} - 1)$ can be written in the matrix form as

$$b_i = \begin{pmatrix} b_{i,0} \\ b_{i,1} \\ \dots \\ b_{i,m_i-1} \end{pmatrix} = Q_i \left( D_i \begin{pmatrix} \gamma_i \\ \gamma_i^{2^{m_i-1}} \\ \dots \\ \gamma_i^2 \end{pmatrix} \cdot (P_i a_i) \right),$$

where $Q_i$, $D_i$, and $P_i$ are binary matrices and the symbol $\cdot$ denotes componentwise multiplication of vectors. It is clear that the vector $C_i = D_i \left(\gamma_i, \gamma_i^{2^{m_i-1}}, \ldots, \gamma_i^2\right)^T$ can be precomputed beforehand. Thus, (4) can be rewritten as

$$F = AQ\,(C \cdot (Pf))\,, \tag{5}$$

where $Q$ is the binary block diagonal matrix of combined post-additions for $\ell+1$ cyclic convolutions, $C$ is the combined vector of constants, and $P$ is the binary block diagonal matrix of combined pre-additions.

On account of (4) and (5), the second stage of the FFT can be considered as multiplying the binary matrix $AQ$ by the vector $C \cdot (Pf)$. To compute the product $(AQ)\,(C \cdot (Pf))$, one can use either the modified "four Russians" algorithm (V.L. Arlazarov, E.A. Dinits, M.A. Kronrod, and I.A. Faradzhev) for multiplication of Boolean matrices, with complexity of $O(n^2/\log n)$ additions over elements of $GF(2^m)$ [9], or a heuristic algorithm, whose complexity we could not estimate. However, for all examples considered, the complexity of the heuristic algorithm was found to be less than that of the modified four Russians algorithm.

The transforms presented have much common with [10]. The main distinctions are the following:

1. The matrix $L$ has a regular structure. This makes it possible to reduce the problem of minimizing the number of multiplications to the classical problem of computing the cyclic convolution.
2. The algorithm involves only two multiplications of binary matrices by vectors, which can be used for a more profound optimization.
3. More efficient optimization algorithm for the sequence of additions is used.

The suggested FFT algorithm is efficient for small values of the transform length (see the table below), while, asymptotically, more efficient FFT algorithms for finite fields are known, with complexity of $O(n \log^2 n)$ operations in the original field [11, 12].

## 4. EXAMPLE

*Example 2.* Let us continue examining the FFT of length 7 over $GF(2^3)$. Let $\alpha$ be a root of the primitive polynomial $x^3 + x + 1$. As a basis of $GF(2^3)$, choose the normal basis $(\gamma, \gamma^2, \gamma^4)$, where $\gamma = \alpha^3$. Decompose the polynomial $f(x)$ as in Example 1 and represent the components of the Fourier transform as the sums

$$\begin{aligned}
f(\alpha^0) &= L_0(\alpha^0) + L_1(\alpha^0) + L_2(\alpha^0) = L_0(1) + L_1(\gamma) \;\; + L_1(\gamma^2) + L_1(\gamma^4) \\
&\qquad\qquad + L_2(\gamma) \;\; + L_2(\gamma^2) + L_2(\gamma^4), \\
f(\alpha^1) &= L_0(\alpha^0) + L_1(\alpha) \;\; + L_2(\alpha^3) = L_0(1) + L_1(\gamma^2) + L_1(\gamma^4) + L_2(\gamma), \\
f(\alpha^2) &= L_0(\alpha^0) + L_1(\alpha^2) + L_2(\alpha^6) = L_0(1) + L_1(\gamma) \;\; + L_1(\gamma^4) + L_2(\gamma^2), \\
f(\alpha^3) &= L_0(\alpha^0) + L_1(\alpha^3) + L_2(\alpha^2) = L_0(1) + L_1(\gamma) \;\; + L_2(\gamma) \;\; + L_2(\gamma^4), \\
f(\alpha^4) &= L_0(\alpha^0) + L_1(\alpha^4) + L_2(\alpha^5) = L_0(1) + L_1(\gamma) \;\; + L_1(\gamma^2) + L_2(\gamma^4), \\
f(\alpha^5) &= L_0(\alpha^0) + L_1(\alpha^5) + L_2(\alpha) \;\; = L_0(1) + L_1(\gamma^4) + L_2(\gamma^2) + L_2(\gamma^4), \\
f(\alpha^6) &= L_0(\alpha^0) + L_1(\alpha^6) + L_2(\alpha^4) = L_0(1) + L_1(\gamma^2) + L_2(\gamma) \;\; + L_2(\gamma^2).
\end{aligned}$$

This system can be written in the matrix form as

$$F = \begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} L_0(1) \\ L_1(\gamma) \\ L_1(\gamma^2) \\ L_1(\gamma^4) \\ L_2(\gamma) \\ L_2(\gamma^2) \\ L_2(\gamma^4) \end{pmatrix} = AS.$$

Then the FFT problem is rewritten as

$$F = A \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma^1 & \gamma^2 & \gamma^4 & 0 & 0 & 0 \\ 0 & \gamma^2 & \gamma^4 & \gamma^1 & 0 & 0 & 0 \\ 0 & \gamma^4 & \gamma^1 & \gamma^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma^1 & \gamma^2 & \gamma^4 \\ 0 & 0 & 0 & 0 & \gamma^2 & \gamma^4 & \gamma^1 \\ 0 & 0 & 0 & 0 & \gamma^4 & \gamma^1 & \gamma^2 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \\ f_3 \\ f_6 \\ f_5 \end{pmatrix}.$$

The first stage of the FFT algorithm consists in computing two cyclic convolutions

$$\begin{pmatrix} b_{i,0} \\ b_{i,1} \\ b_{i,2} \end{pmatrix} = \begin{pmatrix} \gamma^1 & \gamma^2 & \gamma^4 \\ \gamma^2 & \gamma^4 & \gamma^1 \\ \gamma^4 & \gamma^1 & \gamma^2 \end{pmatrix} \begin{pmatrix} a_{i,0} \\ a_{i,1} \\ a_{i,2} \end{pmatrix}, \quad i = 1, 2,$$

where

$$S = \begin{pmatrix} L_0(1) \\ L_1(\gamma) \\ L_1(\gamma^2) \\ L_1(\gamma^4) \\ L_2(\gamma) \\ L_2(\gamma^2) \\ L_2(\gamma^4) \end{pmatrix} = \begin{pmatrix} b_{0,0} \\ b_{1,0} \\ b_{1,1} \\ b_{1,2} \\ b_{2,0} \\ b_{2,1} \\ b_{2,2} \end{pmatrix}, \qquad f = \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \\ f_3 \\ f_6 \\ f_5 \end{pmatrix} = \begin{pmatrix} a_{0,0} \\ a_{1,0} \\ a_{1,1} \\ a_{1,2} \\ a_{2,0} \\ a_{2,1} \\ a_{2,2} \end{pmatrix}.$$

Using the algorithm for computing the three-point cyclic convolution $b_i(x) = b_{i,0}+b_{i,2}x+b_{i,1}x^2 = (\gamma + \gamma^4 x + \gamma^2 x^2)(a_{i,0} + a_{i,1}x + a_{i,2}x^2) \bmod (x^3 - 1)$ presented in [1], we obtain

$$b_i = \begin{pmatrix} b_{i,0} \\ b_{i,1} \\ b_{i,2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} \left( \left[ \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \gamma \\ \gamma^4 \\ \gamma^2 \end{pmatrix} \right] \cdot \left[ \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{i,0} \\ a_{i,1} \\ a_{i,2} \end{pmatrix} \right] \right)$$
$$= Q_i \left( C_i \cdot (P_i a_i) \right), \quad i = 1, 2.$$

Taking into account that $\gamma + \gamma^2 + \gamma^4 = 1$, we see that the algorithm requires 3 multiplications, 4 pre-additions, and 5 post-additions.

Now, we can write (5) for the considered example in the matrix form:

$$F = \begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{pmatrix} = \left( \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \right)$$

$$\times \left( \begin{pmatrix} 1 \\ 1 \\ \gamma^2+\gamma^4 \\ \gamma+\gamma^4 \\ \gamma+\gamma^2 \\ 1 \\ \gamma^2+\gamma^4 \\ \gamma+\gamma^4 \\ \gamma+\gamma^2 \end{pmatrix} \cdot \left[ \begin{pmatrix} 1&0&0&0&0&0&0 \\ 0&1&1&1&0&0&0 \\ 0&0&1&1&0&0&0 \\ 0&1&1&0&0&0&0 \\ 0&1&0&1&0&0&0 \\ 0&0&0&0&1&1&1 \\ 0&0&0&0&0&1&1 \\ 0&0&0&0&1&1&0 \\ 0&0&0&0&1&0&1 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \\ f_3 \\ f_6 \\ f_5 \end{pmatrix} \right] \right) = (AQ)(C \cdot (Pf)).$$

The second stage of the FFT consists in multiplying the binary matrix $AQ$ by the vector $C \cdot (Pf)$:

$$F = \begin{pmatrix} 1&1&0&0&0&1&0&0&0 \\ 1&0&0&1&1&1&0&1&1 \\ 1&0&1&1&0&1&1&1&0 \\ 1&1&0&1&1&0&1&1&0 \\ 1&0&1&0&1&1&1&0&1 \\ 1&1&1&0&1&0&0&1&1 \\ 1&1&1&1&0&0&1&0&1 \end{pmatrix} (C \cdot (Pf)).$$

This stage can be made in 17 additions.

Thus, the FFT of length 7 is reduced to the following sequence of operations:

- pre-additions $P \times f$:

$$\begin{aligned} V_1 &= f_2 + f_4, & V_8 &= f_6 + f_5, \\ V_2 &= f_1 + f_2, & V_9 &= f_3 + f_6, \\ V_3 &= f_1 + f_4, & V_{10} &= f_3 + f_5, \\ V_4 &= f_1 + V_1, & V_{11} &= f_3 + V_8, \end{aligned}$$

- multiplications by constants $C \cdot (Pf)$:

$$\begin{aligned} V_5 &= V_1\alpha, & V_{12} &= V_8\alpha, \\ V_6 &= V_2\alpha^2, & V_{13} &= V_9\alpha^2, \\ V_7 &= V_3\alpha^4, & V_{14} &= V_{10}\alpha^4, \end{aligned}$$

- multiplication of the matrix $AQ$ by the vector $C \cdot (Pf)$:

$$\begin{aligned} T_{10} &= V_{12} + V_{14}, & F_2 &= T_8 + T_{11}, \\ T_{11} &= f_0 + V_{11}, & F_3 &= T_7 + T_{14}, \\ T_{14} &= f_0 + V_4, & T_{12} &= F_2 + T_{10}, \\ T_{15} &= V_5 + V_6, & T_{13} &= F_3 + T_{15}, \\ T_{16} &= V_6 + V_{13}, & F_4 &= T_7 + T_{12}, \\ F_0 &= V_4 + T_{11}, & F_5 &= T_{10} + T_{13}, \\ T_9 &= V_{12} + T_{16}, & F_6 &= F_5 + T_7, \\ T_7 &= V_7 + T_9, & F_1 &= F_4 + T_8. \\ T_8 &= V_5 + T_9, \end{aligned}$$

The overall complexity of the algorithm is $2 \times 3 = 6$ multiplications and $2 \times 4 + 17 = 25$ additions, which is one addition less than that of the algorithm presented in [10].

Complexity of some FFT algorithms

| | Horner's method | | Goertzel's algorithm | | Algorithms from [7, 8] | | Zakharova's method [10] | | Suggested method | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $N_{\text{mul}}$ | $N_{\text{add}}$ | $N_{\text{mul}}$ | $N_{\text{add}}$ | $N_{\text{mul}}$ | $N_{\text{add}}$ | $N_{\text{mul}}$ | $N_{\text{add}}$ | $N_{\text{mul}}$ | $N_{\text{add}}$ |
| 7 | 36 | 42 | 12 | 42 | 9 | 35 | 6 | 26 | 6 | 25 |
| 15 | 196 | 210 | 38 | 210 | 20 | 70 | 16 | 100 | 16 | 77 |
| 31 | 900 | 930 | 120 | 930 | 108 | 645 | 60 | 388 | 54 | 315 |
| 63 | 3844 | 3906 | 282 | 3906 | 158 | 623 | 97 | 952 | 97 | 805 |
| 127 | 15 876 | 16 002 | 756 | 16 002 | 594 | 5770 | 468 | 3737 | 216 | 2780 |
| 255 | 64 516 | 64 770 | 1718 | 64 770 | 1225 | 4715 | 646 | 35 503 | 586 | 7919 |
| 511 | 260 100 | 260 610 | 4044 | 260 610 | 4374 | — | — | — | 1014 | 26 643 |

## 5. COMPARISON OF THE COMPLEXITY OF FFT ALGORITHMS

Let us consider $a + b$ (or $a \times b$) to be an addition (a multiplication) only if both summands (factors) lie in the original field [13]; i.e., operations in the prime field are neglected [1]. The table presents the complexity of the FFT of length $n = 2^m - 1$ over the fields $GF(2^m)$ in the number of multiplications, $N_{\text{mul}}$, and additions, $N_{\text{add}}$. Horner's method is described, e.g., in [7], and the modification of Goertzel's algorithm for finite fields is considered in [1]. All algorithms suggested in the paper have been implemented in software.

The suggested method was presented at the 8th International Workshop "Algebraic and Combinatorial Coding Theory" (Tsarskoe Selo, Russia), the seminar of the Information Systems Chair of the St. Petersburg State Aerospace Instrumentation University, and the seminar at the Institute for Information Transmission Problems of the RAS (Moscow). The authors are grateful to organizers of the workshop and participants of the seminars. They are also grateful to the referee for informing them of the papers [11, 12]. The algorithm of cyclic convolution of length 8 over $GF(2)$, used in constructing the table, was suggested by N. Churkov. The second author (S.V. Fedorenko) is also grateful to the Alexander von Humboldt Foundation for the many years' support of his research.

## REFERENCES

1. Blahut, R.E., *Theory and Practice of Error Control Codes*, Reading, Massachusetts: Addison-Wesley, 1983. Translated under the title *Teoriya i praktika kodov, ispravlyayushchikh oshibki*, Moscow: Mir, 1986.
2. Truong, T.-K., Jeng, J.-H., and Reed, I.S., Fast Algorithm for Computing the Roots of Error Locator Polynomials up to Degree 11 in Reed–Solomon Decoders, *IEEE Trans. Commun.*, 2001, vol. 49, no. 5, pp. 779–783.
3. Fedorenko, S.V. and Trifonov, P.V., Finding Roots of Polynomials over Finite Fields, *IEEE Trans. Commun.*, 2002, vol. 50, no. 11, pp. 1709–1711.
4. Berlekamp, E.R., *Algebraic Coding Theory*, New York: McGraw-Hill, 1968. Translated under the title *Algebraicheskaya teoriya kodirovaniya*, Moscow: Mir, 1971.
5. MacWilliams, F.J. and Sloane, N.J.A., *The Theory of Error-Correcting Codes*, Amsterdam: North-Holland, 1977. Translated under the title *Teoriya kodov, ispravlyayushchikh oshibki*, Moscow: Svyaz', 1979.
6. Fedorenko, S. and Trifonov, P., On Computing the Fast Fourier Transform over Finite Fields, in *Proc. 8th Int. Workshop on Algebraic and Combinatorial Coding Theory, Tsarskoe Selo, Russia*, 2002, pp. 108–111.
7. Gabidulin, E.M. and Afanasyev, V.B., *Kodirovanie v radioelektronike* (Coding in Radio Electronics), Moscow: Radio i Svyaz', 1986.

8. Afanasyev, V.B. and Grushko, I.I., FFT Algorithms for the Fields $GF(2^m)$, in *Pomekhoustoichivoe kodirovanie i nadezhnost' EVM* (Error-Resistant Coding and Reliability of Computers), Moscow: Nauka, 1987, pp. 33–55.

9. Aho, A.V., Hopcroft, J.E., and Ullman, J.D., *The Design and Analysis of Computer Algorithms*, Reading, Massachusetts: Addison-Wesley, 1976. Translated under the title *Postroenie i analiz vychislitel'nykh algoritmov*, Moscow: Mir, 1979.

10. Zakharova, T.G., Fourier Transform Evaluation in Fields of Characteristic 2, *Probl. Peredachi Inf.*, 1992, vol. 28, no. 2, pp. 62–77 [*Probl. Inf. Trans.* (Engl. Transl.), 1992, vol. 28, no. 2, pp. 154–167].

11. Wang, Y. and Zhu, X., A Fast Algorithm for the Fourier Transform over Finite Fields and Its VLSI Implementation, *IEEE J. Select. Areas Commun.*, 1988, vol. 6, no. 3, pp. 572–577.

12. Afanasyev, V.B., On Complexity of FFT over Finite Field, in *Proc. 6th Joint Swedish–Russian Int. Workshop on Information Theory, Mölle, Sweden*, 1993, pp. 315–319.

13. Blahut, R.E., *Fast Algorithms for Digital Signal Processing*, Reading, Massachusetts: Addison-Wesley, 1985. Translated under the title *Bystrye algoritmy tsifrovoi obrabotki signalov*, Moscow: Mir, 1989.