On decoding algebraic codes

Sergei V. Fedorenko

Algebraic codes:

— Hamming codes

— Reed-Solomon codes

— Bose-Chaudhuri-Hocquenghem (BCH) codes

— Goppa codes

— alternant codes

Reed-Solomon codes in standards

CCSDS: Consultative Committee for Space Data Systems

CCSDS 101.0-B-6 standard

(255,223,33), (255,239,17) RS codes

IEEE: The Institute of Electrical and Electronics Engineers

IEEE 802.16 standard

the shortened (255,239,17) RS code

CD **DVD** **HDTV**

Reed-Solomon codes

(n, k, d) Reed-Solomon (RS) code over GF(q)

length $n = q - 1$

number of information symbols $k \in [1, n - 1]$

code distance $d = n - k + 1$

We consider primitive ($n = q - 1$) and narrow-sense ($b = 1$) RS codes.

The RS code generator polynomial is

$$g(x) = \prod_{i=b}^{b+d-2} (x - \alpha^i) = \sum_{i=0}^{n-k} g_i x^i,$$

where α is a primitive element of GF(q).

The RS code generator matrix is

$$G = \begin{pmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & g_0 & g_1 & \cdots & g_{n-k} \end{pmatrix}$$

The RS code parity-check matrix is

$$H = \begin{pmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\ (\alpha^0)^2 & (\alpha^1)^2 & (\alpha^2)^2 & \cdots & (\alpha^{n-1})^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ (\alpha^0)^{d-1} & (\alpha^1)^{d-1} & (\alpha^2)^{d-1} & \cdots & (\alpha^{n-1})^{d-1} \end{pmatrix}$$

Example 1

GF(5) $= \{0, 1, 2, 3, 4\} \mod 5$

GF(5) $= \{0, \alpha^0, \alpha^1, \alpha^2, \alpha^3\}$ $\alpha = 2$ is primitive

α^0	α^1	α^2	α^3	$\alpha^4 = 1$
1	2	4	3	1

(4,2,3) RS code over GF(5)

$$g(x) = (x - \alpha)(x - \alpha^2) = (x - 2)(x - 4) = x^2 + 4x + 3$$

$$G = \begin{pmatrix} g_0 & g_1 & g_2 & 0 \\ 0 & g_0 & g_1 & g_2 \end{pmatrix} = \begin{pmatrix} 3 & 4 & 1 & 0 \\ 0 & 3 & 4 & 1 \end{pmatrix}$$

$$H = \begin{pmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 \\ \alpha^0 & \alpha^2 & \alpha^0 & \alpha^2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \end{pmatrix}$$

Encoding

1. Encoding in the time domain

   (a) cyclic encoding

   (b) systematic encoding

2. Encoding in the frequency domain

Cyclic encoding

The information polynomial of RS code is $I(x) = \sum_{l=0}^{k-1} i_l x^l$

The codeword of RS code is

$$C(x) = \sum_{i=0}^{n-1} c_i x^i = I(x)g(x)$$

Systematic encoding

The information polynomial of RS code is $J(x) = \sum_{l=0}^{k-1} j_l x^l$

The codeword of RS code is

$$C(x) = \sum_{i=0}^{n-1} c_i x^i = x^{d-1} J(x) - \Big((x^{d-1} J(x)) \bmod g(x) \Big)$$

Spectral encoding

The information polynomial of RS code is $M(x) = \sum_{l=0}^{k-1} m_l x^l$

The codeword of RS code is

$$C(x) = \sum_{i=0}^{n-1} c_i x^i.$$

Vandermonde matrix is

$$V = \begin{pmatrix} \alpha^0 & \alpha^0 & \alpha^0 & \cdots & \alpha^0 \\ \alpha^0 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\ (\alpha^0)^2 & (\alpha^1)^2 & (\alpha^2)^2 & \cdots & (\alpha^{n-1})^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ (\alpha^0)^{n-1} & (\alpha^1)^{n-1} & (\alpha^2)^{n-1} & \cdots & (\alpha^{n-1})^{n-1} \end{pmatrix}$$

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \cdots \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} \alpha^0 & \alpha^0 & \alpha^0 & \cdots & \alpha^0 \\ \alpha^0 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\ (\alpha^0)^2 & (\alpha^1)^2 & (\alpha^2)^2 & \cdots & (\alpha^{n-1})^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ (\alpha^0)^{n-1} & (\alpha^1)^{n-1} & (\alpha^2)^{n-1} & \cdots & (\alpha^{n-1})^{n-1} \end{pmatrix} \left(\begin{array}{c} m_0 \\ m_1 \\ m_2 \\ \cdots \\ m_{k-1} \\ \hline 0 \\ \cdots \\ 0 \end{array}\right)$$

$$C = VM$$

$$c_i = M(\alpha^i), \quad i \in [0, n-1]$$

$$C = \mathsf{DFT}(M)$$

Inverse transform

$$\left(\begin{array}{c} m_0 \\ m_1 \\ m_2 \\ \cdots \\ m_{k-1} \\ \hline 0 \\ \cdots \\ 0 \end{array}\right) = V^{-1} \left(\begin{array}{c} c_0 \\ c_1 \\ c_2 \\ \cdots \\ c_{n-1} \end{array}\right)$$

$$M = V^{-1} C$$

$$m_i = \frac{1}{n} C(\alpha^{-i}), \quad i \in [0, n-1]$$

$$M = \mathsf{IDFT}(C)$$

Example 2

Cyclic encoding

$$I(x) = \begin{pmatrix} 0 & 1 \end{pmatrix} = x$$

$$C(x) = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \end{pmatrix} = xg(x) = 3x + 4x^2 + x^3 = \begin{pmatrix} 0 & 3 & 4 & 1 \end{pmatrix}$$

Systematic encoding

$$J(x) = \begin{pmatrix} 4 & 1 \end{pmatrix} = 4 + x$$

$$x^2(x+4) = g(x)x + 2x$$

$$C(x) = x^2(x+4) - 2x = 3x + 4x^2 + x^3 = \begin{pmatrix} 0 & 3 & | & 4 & 1 \end{pmatrix}$$

Spectral encoding

Vandermonde matrix is

$$V = \begin{pmatrix} \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 \\ \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 \\ \alpha^0 & \alpha^2 & \alpha^0 & \alpha^2 \\ \alpha^0 & \alpha^3 & \alpha^2 & \alpha^1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix}$$

$$M(x) = \begin{pmatrix} m_0 & m_1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 0 & 0 \end{pmatrix} = 2 + 3x$$

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 \\ \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 \\ \alpha^0 & \alpha^2 & \alpha^0 & \alpha^2 \\ \alpha^0 & \alpha^3 & \alpha^2 & \alpha^1 \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 3 \\ 4 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$$

$$C(x) = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \end{pmatrix} = \begin{pmatrix} 0 & 3 & 4 & 1 \end{pmatrix} = 3x + 4x^2 + x^3$$

Inverse transform

$$V^{-1} = -\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{pmatrix}$$

$$M = V^{-1}C$$

$$C(x) = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \end{pmatrix} = \begin{pmatrix} 0 & 3 & 4 & 1 \end{pmatrix} = 3x + 4x^2 + x^3$$

$$\begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \end{pmatrix} = -\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{pmatrix} \begin{pmatrix} 0 \\ 3 \\ 4 \\ 1 \end{pmatrix}$$

$$M(x) = \begin{pmatrix} m_0 & m_1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 0 & 0 \end{pmatrix} = 2 + 3x$$

Definitions and notations

The received vector is

$$R(x) = \sum_{i=0}^{n-1} r_i x^i = C(x) + E(x) = \sum_{i=0}^{n-1} c_i x^i + \sum_{i=0}^{n-1} e_i x^i,$$

where $C(x)$ is the codeword,
$E(x)$ is the error vector.

$$\begin{array}{rcccccc} C(x) & = & (c_0 & c_1 & c_2 & \dots & c_{n-1}) \\ E(x) & = & (e_0 & e_1 & e_2 & \dots & e_{n-1}) \\ R(x) & = & (r_0 & r_1 & r_2 & \dots & r_{n-1}) \\ \hline Z & & \alpha^0 & \alpha^1 & \alpha^2 & \dots & \alpha^{n-1} \end{array}$$

The error locator polynomial is

$$W(x) = \prod_{i=1}^{t} (x - Z_i),$$

where $t \leq (d-1)/2$ is the number of errors,
Z_i is the locator of the error position in the error vector $E(x)$,
Y_i is the error value, $i \in [1, t]$.

Syndrome

1. Syndrome in the time domain

$$S(x) = \sum_{i=0}^{d-2} s_i x^i \equiv R(x) \bmod g(x)$$

2. Syndrome in the frequency domain

$$F(x) = \sum_{l=0}^{d-2} f_l x^l,$$

$$f_l = R\left(\alpha^{-(l+k)}\right) = \sum_{i=0}^{n-1} r_i \left(\alpha^{-(l+k)}\right)^i = \sum_{i=0}^{n-1} r_i \left(\alpha^{d-1-l}\right)^i$$

A relationship between $S(x)$ and $F(x)$

$$F(x) = \sum_{l=0}^{d-2} f_l x^l,$$

$$f_l = S\Big(\alpha^{-(l+k)}\Big), \quad l \in [0, d-2]$$

Interpolation

$$R = C + E$$

$$T = \mathsf{IDFT}(R) = \mathsf{IDFT}(C + E) = \mathsf{IDFT}(C) + \mathsf{IDFT}(E) =$$

$$M + \mathsf{IDFT}(E)$$

$$T(x) = \sum_{i=0}^{n-1} T_i x^i$$

$$T_i = \frac{1}{n} R(\alpha^{-i}), \quad i \in [0, n-1]$$

A relationship between $F(x)$ and $T(x)$

$$F(x) = \sum_{l=0}^{d-2} f_l x^l$$

$$T(x) = \sum_{i=0}^{n-1} T_i x^i$$

$$f_l = nT_{l+k}, \quad l \in [0, d-2]$$

Example 3

$$\begin{array}{rcl} C(x) & = & (0 \quad 3 \quad 4 \quad 1) \\ E(x) & = & (0 \quad 0 \quad 2 \quad 0) \\ R(x) & = & (0 \quad 3 \quad 1 \quad 1) \end{array}$$

Z	α^0	α^1	α^2	α^3
Z	1	2	4	3

Construct an interpolating polynomial $T(x)$

$$T = \mathsf{IDFT}(R) = V^{-1}R$$

$$\begin{pmatrix} 0 \\ 0 \\ 3 \\ 2 \end{pmatrix} = - \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{pmatrix} \begin{pmatrix} 0 \\ 3 \\ 1 \\ 1 \end{pmatrix}$$

$$T(x) = \begin{pmatrix} T_0 & T_1 & T_2 & T_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 3 & 2 \end{pmatrix} = 3x^2 + 2x^3$$

Calculate syndrome in the frequency domain $F(x)$

$$f_0 = nT_2 = 4 \cdot 3 = 2$$

$$f_1 = nT_3 = 4 \cdot 2 = 3$$

$$F(x) = 2 + 3x$$

Calculate syndrome in the time domain $S(x)$

$$S(x) = R(x) \bmod g(x) = 4 + 2x$$

A relationship between $S(x)$ and $F(x)$

$$f_0 = S\left(\alpha^{-2}\right) = S\left(\alpha^{2}\right) = S(4) = 2$$

$$f_1 = S\left(\alpha^{-3}\right) = S\left(\alpha\right) = S(2) = 3$$

Decoding Reed-Solomon codes

1. Calculation of syndrome

2. Solution of the key equation

3. Finding the roots of the error locator polynomial

4. Calculation of error values

1. Calculation of syndrome

— direct computation (via Horner method)

— partial DFT (Fedorenko et al., PIT, 2003;
Fedorenko et al., ETT, 2004; Fedorenko, PIT, 2006)

2. Solution of the key equation

— Direct method (Peterson-Gorenstein-Zierler decoder)

— Berlekamp-Massey algorithm

— Euclidean algorithm

3. Finding the roots of the error locator polynomial

— Chien search

— Goertzel algorithm

— Fedorenko and Trifonov algorithm for finding roots of polynomials over finite fields (Fedorenko et al., IEEE TComm., 2002; Fedorenko et al., ETT, 2003)

4. Calculation of error values

— direct computation

— Forney algorithm

1. Calculation of syndrome

$$F(x) = \sum_{l=0}^{d-2} f_l x^l,$$

$$f_l = R\left(\alpha^{-(l+k)}\right), \quad l \in [0, d-2]$$

2. Solution of the key equation

$$\begin{cases} F(x)W(x) \equiv N(x) \bmod x^{d-1} \\ \deg W(x) \leq \frac{d-1}{2} \\ \text{maximize} \deg W(x) \end{cases}$$

3. Finding the roots of the error locator polynomial

$$W(x) = 0$$

4. Calculation of error values

$$Y_i = -\frac{N(Z_i)}{x^d W'(Z_i)}, \quad i \in [1, t],$$

where $W'(x)$ is a formal derivative for $W(x)$

Example 4. Decoding

1. Syndrome

$$F(x) = 2 + 3x$$

2. Key equation

$$\begin{cases} (2+3x)W(x) \equiv N(x) \bmod x^2 \\ \deg W(x) \leq 1 \end{cases}$$

$$x^2 = (2+3x)(2x+2) + 1$$

$$(2+3x)(2x+2) = -1 + x^2$$

$$(2+3x)(2x+2) \equiv 4 \bmod x^2$$

$$\begin{cases} W(x) = 2x+2 = 2(x-4) = 2(x-\alpha^2) \\ N(x) = 4 \end{cases}$$

3. error locators

$$Z_1 = 4$$

4. error values

$$Y_1 = 2$$

$$\begin{array}{rcl} R(x) & = & (0 \;\; 3 \;\; 1 \;\; 1) \\ E(x) & = & (0 \;\; 0 \;\; 2 \;\; 0) \\ C(x) & = & (0 \;\; 3 \;\; 4 \;\; 1) \end{array}$$

A spectral algorithm for decoding Reed-Solomon codes

(Shiozaki, IEEE TIT, 1988; Gao, CINS, 2003;
Fedorenko, IEEE TIT, 2005)

Key equation derivation

If $r_i = c_i$ then $r_i = c_i = M(\alpha^i)$

If $r_i \neq c_i$ then $W(\alpha^i) = 0$.

$$W(\alpha^i) r_i = W(\alpha^i) M(\alpha^i), \quad i \in [0, n-1].$$

Let $P(x) = W(x)M(x)$.

$$\begin{cases} W(\alpha^i) r_i = P(\alpha^i) \\ i \in [0, n-1] \end{cases}$$

Construct an interpolating polynomial $T(x)$ such that

$$T(\alpha^i) = r_i, \quad i \in [0, n-1],$$

where $\deg T(x) < n$.

$$\begin{cases} W(\alpha^i)T(\alpha^i) = P(\alpha^i) \\ i \in [0, n-1] \end{cases}$$

$$\begin{cases} W(x)T(x) - P(x) = (x - \alpha^i)y_i(x) \\ i \in [0, n-1] \end{cases}$$

$$W(x)T(x) - P(x) = (x^n - 1)Y(x)$$

$$W(x)T(x) \equiv P(x) \bmod x^n - 1$$

We solve the key equation and obtain polynomials $P(x)$ and $W(x)$. The information polynomial is

$$M(x) = \frac{P(x)}{W(x)}.$$

Spectral decoding algorithm

1. Interpolation.

Construct an interpolating polynomial $T(x)$ such that

$$T(\alpha^i) = r_i, \quad i \in [0, n-1],$$

where $\deg T(x) < n$.

2. Partial Greatest Common Divisor (GCD).

Solve a congruence

$$\begin{cases} W(x)T(x) \equiv P(x) \bmod x^n - 1 \\ \deg P(x) < \frac{n+k}{2} \\ \text{maximize} \deg P(x) \end{cases}$$

by applying the extended Euclidean algorithm to $x^n - 1$ and $T(x)$, and we obtain a unique pair of polynomials $P(x)$ and $W(x)$.

3. Division.

The information polynomial is

$$M(x) = \frac{P(x)}{W(x)}.$$

4. Re-encoding.

For BCH, Goppa, or alternant codes, the additional re-encoding step

$$C = \mathsf{DFT}(M)$$

is needed.

Implementation

1. Interpolation

Interpolation is IDFT over GF(q).

2. Partial GCD

$$W(x)T(x) \equiv P(x) \bmod x^n - 1, \quad P(x) = W(x)M(x)$$

$$(x^n - 1)Y(x) - T(x)W(x) = -M(x)W(x)$$

Apply the extended Euclidean algorithm to $x^n - 1$ and $T(x)$, and we obtain a unique pair of polynomials $P(x) = W(x)M(x)$ and $W(x)$.

$$\begin{cases} (x^n - 1)\psi(x) + T(x)\varphi(x) = \eta(x) \\ \deg \eta(x) < (n+k)/2 \\ \deg \varphi(x) \leq (d-1)/2 \end{cases}$$

$$\begin{cases} Y(x) = const \cdot \psi(x) \\ W(x) = const \cdot \varphi(x) \\ M(x)W(x) = const \cdot \eta(x) \end{cases}$$

One of the best implementations of the Partial GCD is the Moenck algorithm (1973).

Example 5. Decoding

$$R(x) = \begin{pmatrix} r_0 & r_1 & r_2 & r_3 \end{pmatrix} = \begin{pmatrix} 0 & 3 & 1 & 1 \end{pmatrix} = 3x + x^2 + x^3$$

1. Interpolation

$T(x) = \mathrm{IDFT}(R) = V^{-1}R = \begin{pmatrix} T_0 & T_1 & T_2 & T_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 3 & 2 \end{pmatrix} = 3x^2 + 2x^3$

2. Partial GCD

$$\begin{cases} W(x)T(x) \equiv P(x) \bmod x^n - 1 \\ \deg P(x) < \frac{n+k}{2} \\ \text{maximize} \deg P(x) \end{cases}$$

$$\begin{cases} W(x)(3x^2 + 2x^3) \equiv P(x) \bmod x^4 - 1 \\ \deg P(x) < 3 \end{cases}$$

$$x^4 - 1 = (3x^2 + 2x^3)(3x + 3) + (x^2 - 1)$$

$$(3x + 3)(3x^2 + 2x^3) = -(x^2 - 1) + x^4 - 1$$

$$(3x + 3)(3x^2 + 2x^3) \equiv 4x^2 + 1 \bmod x^4 - 1$$

$$\begin{cases} W(x) = 3x + 3 = 3(x - 4) = 3(x - \alpha^2) \\ P(x) = 4x^2 + 1 \end{cases}$$

3. Division

$$M(x) = \frac{P(x)}{W(x)} = \frac{4x^2 + 1}{3x + 3} = 3x + 2$$

$$M(x) = \begin{pmatrix} m_0 & m_1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 0 & 0 \end{pmatrix} = 2 + 3x$$

Asymptotic complexity

1. Interpolation
   $O(n(\log n)^2)$

2. Partial GCD
   $O(n(\log n)^2)$

3. Division
   $O(n \log n \log\log n)$

The asymptotic complexity of the spectral decoding algorithm $O(n(\log n)^2)$ coincides with complexity of the best classical RS decoding algorithms.

Finding roots of polynomials over finite fields

(Fedorenko et al., IEEE TComm., 2002;
Fedorenko et al., ETT, 2003)

$$F(x) = \sum_{i=0}^{t} f_i x^i, \quad f_i \in \mathrm{GF}(2^m)$$

An element α is a root of $F(x)$ if $F(\alpha) = 0$.

Remark. $\alpha \in \mathrm{GF}(2^m)$ or in the field extension.

Definition. A polynomial $L(y) = \sum_i l_i y^{2^i}, \quad l_i \in \mathrm{GF}(2^m)$ over $\mathrm{GF}(2^m)$ is called a linearized polynomial.

Property.

$$L(\alpha + \beta) = L(\alpha) + L(\beta),$$

where $\alpha, \beta \in \mathrm{GF}(2^m)$.

Definition. A polynomial $A(y) = L(y) + \beta, \quad \beta \in \mathrm{GF}(2^m)$, over $\mathrm{GF}(2^m)$ is called an affine polynomial, where $L(y)$ is a linearized polynomial.

Statement. The Horner's scheme is

$$F(\alpha) = ([\{\ldots(f_t\alpha + f_{t-1})\alpha + \ldots + f_3\}\alpha + f_2]\alpha + f_1)\alpha + f_0.$$

The time complexity of Chien search via Horner's scheme for a polynomial of degree t is equal to

$$C_{\mathsf{H}} = (C_{\mathsf{add}} + C_{\mathsf{mul}})\, t\, (2^m - 1),$$

where C_{add} and C_{mul} are time complexity of one addition and multiplication in the finite field respectively.

Theorem. Each polynomial $F(x) = \sum_{i=0}^{t} f_i x^i, \quad f_i \in \mathrm{GF}(2^m)$ can be represented as

$$F(x) = f_3 x^3 + \sum_{i=0}^{\lceil (t-4)/5 \rceil} x^{5i}(f_{5i} + L_i(x)),$$

where $L_i(x) = \sum_{j=0}^{3} f_{5i+2^j} x^{2^j} =$
$f_{5i+1}x + f_{5i+2}x^2 + f_{5i+4}x^4 + f_{5i+8}x^8$,
$\lceil p \rceil$ is the smallest integer greater or equal than p,
$f_l = 0$ for $l > t$.

Example 6

Let t=8 then

$$F(x) = \sum_{i=0}^{8} f_i x^i = A_1(x) + x^3(A_2(x) + f_6 x^3),$$

where $A_1(x) = f_0 + L_1(x) = f_0 + f_1 x + f_2 x^2 + f_4 x^4 + f_8 x^8$,
$A_2(x) = f_3 + L_2(x) = f_3 + f_5 x^2 + f_7 x^4$.

Algorithm for finding roots of polynomials over finite fields

1. Preliminary compute $L_i^{(k)} = L_i(\alpha^k), \quad k = [0, m-1]$, $i \in [0, \lceil (t-4)/5 \rceil]$,

   where $L_i(x)$ is linearized polynomial $L_i(x) = \sum_{j=0}^{3} f_{5i+2^j}\, x^{2^j}$.

2. Initialize $A_i^{(0)} = f_{5i}$.

3. Represent each $\mathbf{x}_j \in \mathrm{GF}(2^m), \quad j \in [0, 2^m-1]$, in the standard basis as an element of Gray code with $\mathbf{x}_0 = 0$, compute $A_i^{(j)} = A_i^{(j-1)} + L_i^{(\delta(\mathbf{x}_j, \mathbf{x}_{j-1}))}, \quad j \in [1, 2^m-1]$, where $\delta(\mathbf{x}_j, \mathbf{x}_{j-1})$ indicates position in which $\mathbf{x}_j$ differs from $\mathbf{x}_{j-1}$.

4. Compute $F(x_j) = f_3 x_j^3 + \sum_{i=0}^{\lceil (t-4)/5 \rceil} x_j^{5i} A_i^{(j)}, \quad j \in [0, 2^m - 1]$.

   If $F(x_j) = 0$ then x_j is a root of the polynomial.

The time complexity

$$C_{\mathsf{TF}} = m \left\lceil \frac{t+1}{5} \right\rceil (4C_{\mathsf{mul}} + 3C_{\mathsf{add}}) +$$

$$+ \left(\left\lceil \frac{t+1}{5} \right\rceil (2C_{\mathsf{add}} + C_{\mathsf{mul}}) + 2C_{\mathsf{exp}} \right) (2^m - 1),$$

where C_{add} (C_{mul}, C_{exp}) is the time complexity of one addition (multiplication, exponentiation) in the finite field.

A method for computation of the discrete Fourier transform over GF(2^m)

(Fedorenko et al., PIT, 2003;
Fedorenko et al., ETT, 2004; Fedorenko, PIT, 2006)

$$F = Vf$$

$$\begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ \cdots \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} \alpha^0 & \alpha^0 & \alpha^0 & \cdots & \alpha^0 \\ \alpha^0 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\ (\alpha^0)^2 & (\alpha^1)^2 & (\alpha^2)^2 & \cdots & (\alpha^{n-1})^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ (\alpha^0)^{n-1} & (\alpha^1)^{n-1} & (\alpha^2)^{n-1} & \cdots & (\alpha^{n-1})^{n-1} \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \cdots \\ f_{n-1} \end{pmatrix},$$

where $V = (\alpha^{ij}), \quad i, j \in [0, n-1]$, is a Vandermonde matrix, an element $\alpha \in \mathrm{GF}(2^m)$ of order $\mathrm{ord}(\alpha) = n \mid 2^m - 1$ is the Fourier transform kernel.

Suggested method turns n-point Fourier transform into several different cyclic convolutions for any n.

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

is a basis circulant matrix,
where $\left(\gamma_i^{2^0}, \gamma_i^{2^1}, \dots, \gamma_i^{2^{m_i-1}}\right)$ is a normal basis of GF(2^{m_i}) $\subset$ GF(2^m).

The equivalent Fourier transform $F_e = V_e f_e$ has the following structure:

$$F_e = V_e f_e = A_e D_e f_e = \begin{pmatrix} A_{11} & \cdots & A_{1l} \\ \cdots & \cdots & \cdots \\ A_{l1} & \cdots & A_{ll} \end{pmatrix} \begin{pmatrix} C_1 & 0 & \ldots & 0 \\ 0 & C_2 & \ldots & 0 \\ \ldots & \ldots & \ddots & \ldots \\ 0 & 0 & \ldots & C_l \end{pmatrix} f_e,$$

where A_e is a binary matrix,
A_e consists of binary circulants A_{ij} and block circulants,
D_e is a block diagonal matrix,
D_e consists of basis circulant matrices C_i,
l is the number of cyclotomic cosets modulo n over GF(2).

The Fourier transform algorithms of length $n = 2^m - 1$ over GF(2^m) take two stages:

1. The first stage is calculation of l m-point cyclic convolutions;

2. The second stage is multiplying the binary matrix A_e by the vector $D_e f_e$.

The complexity of the first stage is about $n \log n$ multiplications and additions over elements of GF(2^m). The complexity of the second stage is $N_{add} < 2n^2 / \log n$ additions over elements of GF(2^m).

If the number of multiplications is to be minimized, then the best algorithm for computation of the DFT for small lengths ($n \leq 511$) is algorithm by Trifonov and Fedorenko
(Fedorenko et al., PIT, 2003).

The table presents the complexity of the DFT of length $n = 2^m - 1$ over the fields GF(2^m) in the number of multiplications N_{mul}, and additions, N_{add}.

The complexity of some DFT algorithms

	Horner's method		Goertzel's algorithm		Afanasyev's method		Zakharova's method		Trifonov-Fedorenko's method	
n	N_{mul}	N_{add}	N_{mul}	N_{add}	N_{mul}	N_{add}	N_{mul}	N_{add}	N_{mul}	N_{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	15876	16002	756	16002	594	5770	468	3737	216	2780
255	64516	64770	1718	64770	1225	4715	646	35503	586	7919
511	260100	260610	4044	260610	4374	—	—	—	1014	26643

Example 7. 7-point Fourier transform over GF(2^3)

GF(2^3): $\alpha^3 + \alpha + 1 = 0$

Let $\left(\gamma^1, \gamma^2, \gamma^4\right) = \left(\alpha^6, \alpha^5, \alpha^3\right)$ be a normal basis for GF(2^3).

$$\begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_4 \\ F_6 \\ F_5 \\ F_3 \end{pmatrix} = \left(\begin{array}{c|ccc|ccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ \hline 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right) \left(\begin{array}{c|ccc|ccc} 1 & & & & & & \\ \hline & \gamma^1 & \gamma^2 & \gamma^4 & & & \\ & \gamma^2 & \gamma^4 & \gamma^1 & & & \\ & \gamma^4 & \gamma^1 & \gamma^2 & & & \\ \hline & & & & \gamma^1 & \gamma^2 & \gamma^4 \\ & & & & \gamma^2 & \gamma^4 & \gamma^1 \\ & & & & \gamma^4 & \gamma^1 & \gamma^2 \end{array}\right) \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \\ f_6 \\ f_5 \\ f_3 \end{pmatrix}.$$