Inversion in GF (2^n) with the aid of Hankel Polynomials

Alexei, Uteshev
St.Petersburg State University
St.Petersburg, Russia
e-mail: alexeiuteshev@gmail.com

Ivan, Baravy
St.Petersburg State University
St.Petersburg, Russia
e-mail: ivan.baravy@apmath.spbu.ru

ABSTRACT

We treat the problem of inversion of an element in $GF(2^n)$. Our approach is based on representation of this inversion in the form of an appropriate determinant (Hankel polynomial) and further its computation with the aid of suggested recursive procedure.

Keywords

Inversion in finite fields, error-correcting codes, Hankel polynomials

1. INTRODUCTION

The problem of finding inversion for a Galois field $GF(p^n)$ nonzero element $\mathfrak{a}$ is a known theoretical problem which is of vital importance for the error-correcting codes practice. The two constructive algorithms for computation of $\mathfrak{a}^{-1}$ consist either in application of the extended Euclidean algorithm (computing the continuant) or in exponentiation $\mathfrak{a}^{p^n-2}$; the latter, however, is impractical for large fields.

2. ALGEBRAIC PRELIMINARIES

Let $F(x)=x^n+A_1x^{n-1}+\ldots+A_n$ and $G(x)=B_0x^{n-1}+B_1x^{n-2}+\ldots+B_{n-1}$ be polynomials over $\mathbb{C}$. Consider the following Laurent expansion

$$\frac{1}{F(x)}=\sum_{k=n-1}^{\infty}\frac{d_k}{x^{k+1}},\quad \frac{G(x)}{F(x)}=\sum_{k=0}^{\infty}\frac{c_k}{x^{k+1}}. \qquad (1)$$

The coefficients $\{d_k\}_{k=n-1}^{\infty}$ can be determined by the recurrent formulae

$$d_k=\begin{cases}1 & if\ k=n-1;\\ -d_{n-1}A_1 & if\ k=n;\\ -\left(d_{k-1}A_1+\ldots+d_{n-1}A_{k-n+1}\right) & if\ k<2n-1;\\ -\left(d_{k-1}A_1+\ldots+d_{k-n}A_n\right) & if\ k\ge 2n-1.\end{cases} \qquad (2)$$

Once they are calculated, the coefficients $\{c_k\}_{k=0}^{\infty}$ are determined by

$$c_k=\begin{cases}d_{k+n-1}B_0+\ldots+d_{n-1}B_k & if\ k<n-1;\\ d_{k+n-1}B_0+\ldots+d_kB_{n-1} & if\ k\ge n-1.\end{cases} \qquad (3)$$

Theorem 1 (Kronecker [1, 2]). *Polynomials $F(x)$ and $G(x)$ are relatively prime iff the Hankel matrix*

$$C=[c_{j+k}]_{j,k=0}^{n-1} \qquad (4)$$

is nonsingular. One can also compute polynomials $u(x)$ and $v(x)$ providing the linear representation

$$v(x)F(x)+u(x)G(x)\equiv\det C \qquad (5)$$

with the aid of the minors of the matrix (4). For instance, $u(x)$ can be computed as the determinant of the matrix obtained from (4) by replacing its last row by $[1,x,x^2,\ldots,x^{n-1}]$:

$$u(x)=\left|\begin{matrix} c_0 & c_1 & \ldots & c_{n-1}\\ c_1 & c_2 & \ldots & c_n\\ \cdots & & & \cdots\\ c_{n-2} & c_{n-1} & \ldots & c_{2n-3}\\ 1 & x & \ldots & x^{n-1}\end{matrix}\right|. \qquad (6)$$

Equality (5) is equivalent to the fact that $u(x)/\det C=[G(x)]^{-1} \pmod{F(x)}$ can be computed with the aid of specially constructed determinant composed from the coefficients of expansion (1). Our aim now is to organize efficient computation of this determinant. We begin with

Definition 1. *For any sequence $\{a_k\}_{k=0}^{\infty}\subset\mathbb{C}$ the polynomial represented in determinantal form as*

$$H_k(x)=\left|\begin{matrix} a_0 & a_1 & \ldots & a_k\\ a_1 & a_2 & \ldots & a_{k+1}\\ \cdots & & & \cdots\\ a_{k-1} & a_k & \ldots & a_{2k-1}\\ 1 & x & \ldots & x^k\end{matrix}\right| \qquad (7)$$

is called the k-th Hankel polynomial [3] generated by the sequence $\{a_k\}_{k=0}^{\infty}$. We also set $H_0(x)\equiv 1$.

Polynomials of this type appear in different areas of computational mathematics such as approximation theory, Reed–Solomon codes, etc. Thus, the polynomial (6) is the $(n-1)$-th Hankel polynomial generated by the sequence $\{c_k\}_{k=0}^{\infty}$. If we denote by C_k the k-th leading principal minors of the matrix (4), then $u(x)\equiv C_{n-1}x^{n-1}+\ldots$ where dots mean the terms of degree $<n-1$.

Example 1. *For $F(x)=x^8+x^4+x^3+x^2+1$ and $G(x)=x^7+x^5+x^2+x$ find Hankel polynomials $\{H_k(x)\}_{k=1}^{7}$ generated by the sequence $\{c_k\}_{k=0}^{\infty}$.*

Solution. We first compute the values $\{d_k\}_{k=7}^{20}$ via (2) and then $\{c_k\}_{k=0}^{13}$ via (3):

$$\{1,0,1,0,-1,0,-1,-1,-1,1,1,2,4,1\}\ .$$

Thus, one has

$$H_1(x) = x, \quad H_2(x) = x^2 - 1, \quad H_3(x) = -2\,x^3 - 2\,x,$$
$$H_4(x) = 4\,x^4 - 2\,x^3 - 2\,x + 4,$$
$$H_5(x) = -6\,x^5 - 8\,x^4 + 10\,x^3 + 3\,x^2 + 4\,x - 11,$$
$$H_6(x) = -31\,x^6 + 6\,x^5 + 8\,x^4 - 10\,x^3 - 3\,x^2 - 35\,x - 20,$$
$$H_7(x) = -56\,x^7 - 51\,x^6 + 46\,x^5 + 24\,x^4 - 58\,x^3 - 79\,x^2 - 119\,x - 4\ .$$

It turns out that

$$H_3(x) \equiv -2\,x\,H_2(x) - 4\,H_1(x),$$
$$H_4(x) \equiv (-2\,x + 1)H_3(x) - 4\,H_2(x),$$
$$H_5(x) \equiv (-3/2x - 11/4)H_4(x) - 9/4\,H_3(x),$$
$$H_6(x) \equiv (31/6\,x - 71/9)H_5(x) - 961/36\,H_4(x),$$
$$H_7(x) \equiv (56/31\,x + 1917/961)H_6(x) - 3136/961\,H_5(x)\ .$$

This means: for $k > 2$ every polynomial $H_k(x)$ is a linear combination of $H_{k-1}(x)$ and $H_{k-2}(x)$. $\square$

Denote by h_{kj} the coefficients of the expansion of Hankel polynomials generated by $\{c_k\}_{k=0}^{\infty}$ in powers of x:

$$H_k(x) \equiv \sum_{j=0}^{k} h_{kj} x^{k-j} = C_k x^k + h_{k1} x^{k-1} + \dots\ .$$

Theorem 2. Three consecutive Hankel polynomials $H_{k-2}(x), H_{k-1}(x)$ and $H_k(x)$ are connected by the identity:

$$[C_k]^2 H_{k-2}(x) + [C_{k-1}]^2 H_k(x) + (C_k h_{k-1,1} - C_{k-1} h_{k1} - C_k C_{k-1} x) H_{k-1}(x) \equiv 0\ . \tag{8}$$

Provided that the coefficients of $H_{k-1}(x)$ are precomputed, the values for C_k and h_{k1} can be calculated via the formulae

$$C_k = \sum_{j=0}^{k-1} c_{k-1+j} h_{k-1,k-1-j},\ h_{k1} = \sum_{j=0}^{k-1} c_{k+j} h_{k-1,k-1-j}\ . \tag{9}$$

This result provides one with an opportunity to compute recursively the sequence $\{H_k(x)\}_{k=0}^{\infty}$ of Hankel polynomials: for $k \geq 2$ every polynomial can be determined as the linear combination of the two preceding ones. However, it should be mentioned that the case of vanishment of C_{k-1} makes the formula (8) useless for computing $H_k(x)$. Though the condition $C_{k-1} = 0$ should be treated as an exceptional case when dealing with polynomials in $\mathbb{C}$, this condition cannot be ignored for the case of finite fields.

3. INVERSION IN GF (2^n)

We treat $GF(2^n)$ as $\mathbb{Z}_2[x]/F(x)$ where $F(x) \in \mathbb{Z}_2[x]$ is the generating polynomial of the field, i.e., an irreducible polynomial of degree n over $\mathbb{Z}_2$.

Example 2. Let us recompute Hankel polynomials from Example 1 modulo 2.

Solution. One has

$$H_1(x) \equiv_2 x,\ H_2(x) \equiv_2 x^2 + 1,\ H_3(x) \equiv_2 H_4(x) \equiv_2 0$$
$$H_5(x) \equiv_2 H_2(x),\ H_6(x) \equiv_2 H_7(x) \equiv_2 x^6 + x^2 + x\ .$$

If $F(x) = x^8 + x^4 + x^3 + x^2 + 1$ is treated as the generating polynomial for $GF(2^8)$, then in this field the inversion of the element $G(x) = x^7 + x^5 + x^2 + x$ equals $H_7(x)$. One can also notice that $H_6(x) \equiv_2 (x^4 + x^2)H_2(x) + H_1(x)$. $\square$

This example illustrates that, firstly, the case $H_k(x) \equiv 0$ should not be treated as an exceptional one, and, secondly, if $H_k(x) \not\equiv 0$ then this polynomial can be represented in the form $H_k(x) \equiv q(x)H_j(x) + H_i(x)$ with $i < j < k$ and $q(x) \in \mathbb{Z}_2[x]$. In comparison with the case of infinite fields, for the finite fields one cannot expect that, generically, $j = k - 1, i = k - 2$.

The constructive generalization in $GF(2^n)$ of the recursive computational scheme for computing the inverse element can be executed with the aid of

3.1 Berlekamp–Massey algorithm

Hereinafter it will be referred to as the *BM-algorithm* [4]. Hankel polynomial $H_k(x)$, defined by (7), can be interpreted as the *characteristic polynomial* for the linear recurrent sequence of the order k with the first $2k$ elements coinciding with $\{a_j\}_{j=0}^{2k-1}$. That means: if $H_k(x) \equiv \sum_{j=0}^{k} h_{kj} x^{k-j}$ then

$$\sum_{j=0}^{k} h_{kj} a_{k-j+\ell} = 0 \quad for\ \ell \in \{0, \dots, k-1\}.$$

The goal of the BM-algorithm is to find the *minimal annihilating polynomial* for a linear recurrent sequence $\{c_k\}_{k=0}^{2(n-1)-1} \subset GF(2)$. Minimal polynomial coincides with $H_{n-1}(x)$ iff $h_{n-1,n-1} \neq 0$; otherwise it equals $H_{n-1}(x)/x^s$ where s denotes the multiplicity of $x = 0$ as a root of $H_{n-1}(x)$.

The pseudocode representation of the BM-algorithm internals permits one to watch its recursive character.

```
      length = 2*(n-1)
(1)   B = H = 1
      L = 0
      m = -1

      d = 0
(2)   for (j = 0; j <= L; j++)
          d = d + H[j] * C[i - j]

      if (d != 0)
          T = H
          for (k = 0; k < length - i + m; k++)
(3)           H[i - m + k] = H[i - m + k] + B[k]
          if (2*L <= i)
              L = i + 1 - L
              m = i
              B = T

      if (i < length)
(4)       i = i+1
          goto (2)
```

We have modified this scheme with the aim to restore the string of the coefficients of the polynomial $H_{n-1}(x)$ from the string of the coefficients of the minimal polynomial for the sequence $\{c_k\}_{k=0}^{2n-3}$.

The multiplicity s itself is computed implicitly via the above scheme. It equals $2(n-1)-m$ which naturally leads from the following algorithm properties:

- The string of coefficients of any minimal polynomial starts with 1;
- If $1\dots1$ is the string of the minimal annihilating polynomial then $\underbrace{0\dots0}_{s\,times}1\dots1$ is also the string of annihilating polynomial;
- Multiplicity s does not exceed $i-m$ at each step and, therefore, equals $2(n-1)-m$ after $2(n-1)$ cycles.

Example 3. *Let $GF\left(2^{16}\right)$ be generated by*

$$F(x) = x^{16} + x^5 + x^3 + x^2 + 1 \ .$$

Find the inversion for

$$G(x) = x^{15} + x^{12} + x^{11} + x^{10} + x^9 + x^4 + x^3 \ .$$

Solution. With the sequence $\{c_k\}_{k=0}^{29}$ computed via (3) and represented as a string:

$$100111100000110111010111111011,$$

the list of strings of the coefficients of annihilating polynomials for all the $2(n-1)=30$ steps is as follows (corresponds to the decreasing order of powers of x):

```
01: 1100000000000000   16: 1110011000000000
02: 1000000000000000   17: 1110011000000000
03: 1000000000000000   18: 1110011000000000
04: 1001000000000000   19: 1110011000000000
05: 1101000000000000   20: 1110011000101010
06: 1101000000000000   21: 1001010100101010
07: 1100000000000000   22: 1001010100101010
08: 1010100000000000   23: 1000100111101010
09: 1010100000000000   24: 1000011110001010
10: 1011001000000000   25: 1000000010111010
11: 1110011000000000   26: 1000000010111010
12: 1110011000000000   27: 1000000010111010
13: 1110011000000000   28: 1000000001011100
14: 1110011000000000   29: 1000000000101111
15: 1110011000000000   30: 1100000000000001
```

Thus $[G(x)]^{-1} = x^{15}+x^{14}+1$. The strings numbered 1, 2, 4, 7, 8, 10, 18, 27 yield Hankel polynomials $H_k(x)$ for $k \in \{1,2,3,4,5,6,13,14\}$ respectively. For this example $H_7(x) \equiv \ldots \equiv H_{12}(x) \equiv 0$. $\square$

3.2 Modifications and optimization

Before considering any optimization we should first remind important propositions on complexity of BM-algorithm. As it was mentioned above, minimal annihilation polynomial represents the recurrent relationship for the input sequence. It is calculated in $O(t^2)$ time, where t stands for the order of the recurrent sequence and thus can significantly vary depending on the input sequence. In other words, $t+1$ equals the *length* of the minimal polynomial $H(x)$, which is defined as 1+ the difference between the highest and the lowest exponents of the variable x contained in the expansion of $H(x)$. It can be shown that the length of an average polynomial $H(x)$ with $\deg H \le n$ equals $n-1$. Therefore, the asymptotic complexity of the proposed algorithm is nearly the same as that one based on the extended Euclidean algorithm (which is a standard tool used for inversion in finite fields).

It is worth mentioning the following advantages of the proposed algorithm over the extended Euclidean one:

- The BM-algorithm does not use division operation which is known to be highly computationally expensive;
- There are no true multiplications in it since multiplication in $GF(2)$ is equivalent to logical AND operation;
- Both **for** cycles can be easily vectorized to speed up them several times on modern CPUs.

While the first two statements are rather evident the last one should be explained.

Vectorization technique implies packing data into blocks as to process them one block at a time. Since coefficients of $H(x)$ are either 0 or 1 they can be naturally represented as bits and then grouped into bytes and data words. The longer are words we operate on, the faster is data processing. Speaking in terms of machine codes and CPU architecture, we are interested in registers of a maximum length capable to handle logical XOR (binary addition), AND (multiplication), SHL/SHR (shift) and ROL/ROR (rotating) instructions. Those are 32- and 64-bit general purpose registers on modern CPUs. This means that finite field inversion problem can be solved especially successful for fields up to $GF(2^{32})$ or $GF(2^{64})$. To be concrete, the whole first pseudocode **for** cycle becomes single[1] AND instruction acting on two registers. The second cycle is replaced by the pared SHL and XOR instructions.

A direct consequence of the above mentioned optimization procedure is nearly linear computational complexity of the BM-algorithm for $GF(2^n)$ with $n \le 64$. For $n > 64$, this complexity is $\sim \gamma n^2$, however the constant γ can be diminished drastically with the aid of vectorization.

Another valuable kind of optimization consists in choosing a specific generating polynomial $F(x)$. Its coefficients influence the sequence $\{c_k\}$ by virtue of the sequence $\{d_k\}$ computed via (2). Thus, in order to inverse several elements in the field, one should compute first $\{d_k\}$ and then to utilize purely multiplication formulae (3) to obtain the coefficients $\{c_k\}$. Therefore, the *sparse* structure of $F(x)$ induces the sparse structure of $\{d_k\}$ and therefore affects the algorithm complexity.

One additional possible modification is related to the recursive versions of both extended Euclidean and BM-algorithms [4]. In comparison with their nonrecursive counterparts, their computational complexity is $O(n \log^2 n)$.

4. ACKNOWLEDGEMENT

[1]On Intel x86 and AMD x86_64 CPUs a series of XOR instructions can be omitted here in favour of cheap checking of parity flag.

This research was supported by the St.Petersburg State University research grant **9.38.674.2013** and by the research grant of the company RAIDIX[2].

REFERENCES

[1] L. Kronecker, “Zur Theorie der Elimination einer Variabeln aus zwei algebraischen Gleichungen”, *Werke*, Bd. 2, Teubner, Leipzig, pp. 113-192, 1897.

[2] A.Yu. Uteshev, T.M. Cherkasov, “The search for the maximum of a polynomial”, *J. Symbolic Computation*, Vol. 25, 5, pp. 587-618, 1998.

[3] P. Henrici,“Applied and Computational Complex Analysis”, V. 1, NY. Wiley, 1974.

[4] R. Blahut,“Fast Algorithms for Digital Signal Processing”, Addison-Wesley, 1985.

[2]http://www.raidixstorage.com