Available online at www.sciencedirect.com

SCIENCE DIRECT®

Finite Fields and Their Applications 11 (2005) 269–277

FINITE FIELDS AND THEIR APPLICATIONS

http://www.elsevier.com/locate/ffa

A note on the factorization method of Niederreiter

Sangtae Jeong[a,*,1], Young-Ho Park[b]

[a] *Department of Mathematics, Inha University, Incheon 402-751, Republic of Korea*
[b] *Department of Information Security, Sejong Cyber University, Seoul 143-150, Republic of Korea*

Received 14 January 2004; revised 21 July 2004

Communicated by Harald Niederreiter

Available online 16 December 2004

Abstract

In this paper, we explicitly obtain the coefficient matrix arising from a linearization of Niederreiter's factorization algorithm and analyze the complexity of setting it up. It turns out that its setup cost is linear both in the degree of a polynomial to be factored and in the size of the base field.

Keywords: Hasse–Teichmüller derivatives; Niederreiter's factorization algorithm

1. Introduction

In 1992, Niederreiter [10,11] proposed a new deterministic algorithm for factoring univariate polynomials over finite fields, which is based on suitable differential equations defined by the Hasse–Teichmüller derivatives over the rational function field. As in the classical algorithm of Berlekamp, Niederreiter's method reduces the factorization of a polynomial $f \in \mathbb{F}_q[x]$ to the calculation of the null space of a certain matrix, which will be called a coefficient matrix. Indeed, Niederreiter obtained a formula of the coefficient matrix in terms of the Berlekamp matrix and the Hankel matrices associated to a polynomial f to be factored and used this formula to give an estimate of its setup cost. We refer to [12,15] for arithmetic complexity of the whole algorithm including

* Corresponding author. Fax: +82-32-874-5615.
E-mail addresses: stj@math.inha.ac.kr (S. Jeong), youngho@cybersejong.ac.kr (Y.-H. Park).
[1] Supported by KOSEF R05-2003-000-10160-0-2004.

1071-5797/$ - see front matter © 2004 Published by Elsevier Inc.
doi:10.1016/j.ffa.2004.08.002

the setup cost. On the other hand, several authors in [3,4,15] recently reported on implementation results of Niederreiter's algorithm, which show that it is well suited for factoring polynomials over small finite fields.

Since the coefficient matrix is closely related to the Berlekamp matrix, it is very desirable to obtain the matrix explicitly with much less cost. The purpose of this paper is thus to derive an explicit formula for the coefficient matrix, which obtains a general case of the formula in [4, Lemma 2.3] and then to give an improved estimate of its setup cost. It turns out that the setup cost is linear both in the degree of a polynomial to be factored and in the size of the base field. Moreover, as in characteristic two in [5], we see that the coefficient matrix has a special form which may result in a speedup in the calculation of the null space of the matrix on a more general setting.

2. Hasse–Teichmüller derivatives

Let $\mathbb{F}_q$ be a finite field of q elements where q is a power of prime characteristic p. For each integer $n \geqslant 0$ the Hasse–Teichmüller derivative $H^{(n)}$ of order n is defined on the field $\mathbb{F}_q((x^{-1}))$ of formal Laurent series in the variable x^{-1} over $\mathbb{F}_q$ as follows:

$$H^{(n)}\left(\sum_{i=w\in\mathbb{Z}}^{\infty} c_i x^{-i}\right) = \sum_{i=w\in\mathbb{Z}}^{\infty} \binom{-i}{n} c_i x^{-i-n}.$$

For later use we here state the product rule and the quotient rule, which were shown by Teichmüller and Göttfert, respectively, but we refer to [2,6–8,17] for more basic properties of Hasse–Teichmüller derivatives such as the chain rule.

Product rule. *For* f, g *in* $\mathbb{F}_q((x^{-1}))$, $H^{(n)}(fg) = \sum_{i=0}^{n} H^{(i)}(f)H^{(n-i)}(g)$.

Quotient rule. *For* f, g *in* $\mathbb{F}_q((x^{-1}))$,

$$H^{(n)}\left(\frac{g}{f}\right) = \frac{H^{(n)}g}{f} + \sum_{i=1}^{n} H^{(n-i)}(g) \sum_{j=1}^{i} \frac{(-1)^j}{f^{j+1}} \sum_{\substack{i_1,\ldots,i_j\geqslant 1,\\ i_1+\cdots i_j=i}} H^{(i_1)}(f)\cdots H^{(i_j)}(f).$$

We have the following simple lemma.

Lemma 2.1. *Let* $r > 1$ *be a divisor of* q. *Then for each* n *with* $1 \leqslant n \leqslant r-1$, $H^{(n)}$ *kills the* r*th powers of* $\mathbb{F}_q((x^{-1}))$.

Proof. Write $f^r = \sum_{i=w\in\mathbb{Z}}^{\infty} f_i^r x^{-ri}$ for $f = \sum_{i=w\in\mathbb{Z}}^{\infty} f_i x^{-i} \in \mathbb{F}_q((x^{-1}))$. We compute

$$H^{(n)}(f^r) = \sum_{i=w\in\mathbb{Z}}^{\infty} \binom{-ri}{n} f_i^r x^{-ri-n} = 0$$

since $\binom{ri}{n} \equiv 0 \pmod{p}$ for all integers i and n with $1 \leqslant n \leqslant r-1$. □

3. Description of Niederreiter's algorithm

Following the approach in [12], we will describe the algorithm of Niederreiter for factoring a monic polynomial f in $\mathbb{F}_q[x]$. Throughout, we fix an integer $r > 1$ so that $\mathbb{F}_r$ can be a subfield of order r of $\mathbb{F}_q$. Two cases where $r = q$ or $r = p$, the characteristic of $\mathbb{F}_q$, are of greater interest to us.

Niederreiter considers the following differential equation over the rational function field $\mathbb{F}_q(x)$:

$$H^{(r-1)}(y) = y^r. \tag{1}$$

Niederreiter's polynomial factorization is based on the following theorem.

Theorem 3.1. *Let $f = g_1^{e_1} \cdots g_m^{e_m}$ be the canonical factorization over $\mathbb{F}_q$ of a monic nonconstant polynomial f. Then a basis of the $\mathbb{F}_r$-space of solutions to (1) of the form $y = \frac{h}{f}$ with f fixed, denoted $L_{q,r}(f)$, is given by $\{\frac{g_1'}{g_1}, \ldots, \frac{g_m'}{g_m}\}$.*

From Theorem 3.1 every rational function $\frac{h}{f} \in L_{q,r}(f)$ has a unique representation

$$h/f = \sum_{i=1}^{m} c_i \frac{g_i'}{g_i} \text{ with } c_i \in \mathbb{F}_r. \tag{2}$$

For a square-free polynomial f, we derive from (2)

$$\gcd(f, h) = \prod_{\substack{1 \leqslant i \leqslant m \\ c_i = 0}} g_i \text{ for } h \neq 0, f'.$$

Since reducing a polynomial to be factored to a square-free one is not a serious problem it is very crucial to efficiently find a basis of $L_{q,r}(f)$ in the factorization procedure. To do so, we write solutions in $L_{q,r}(f)$ as

$$f^r H^{(r-1)}\left(\frac{h}{f}\right) = h^r, \tag{3}$$

where $h \in \mathbb{F}_q[x]$ is an unknown polynomial of degree $< d := \deg(f)$. The restriction on the degree of h follows easily by comparing the degrees of polynomials on both sides of (3). Now comparing the coefficients of both sides of (3) we obtain a system

of d algebraic equations for the unknown coefficients $h_0, \ldots, h_{d-1}$ of $h = \sum_{i=0}^{d-1} h_i x^i$. If $N_r(f)$ denotes the $d \times d$ coefficient matrix over $\mathbb{F}_q$ on the left-hand side of (3) and $\mathbf{h} = (h_0, \ldots, h_{d-1})$ is the coefficient vector of h and $\mathbf{h}^{[r]}$ stands for the vector $(h_0^r, \ldots, h_{d-1}^r)$ then (3) is equivalent to the system

$$N_r(f)\mathbf{h}^T = (\mathbf{h}^{[r]})^T. \tag{4}$$

Niederreiter gave two ways of setting up $N_r(f)$ and described a polynomial-time algorithm of computing it. For arbitrary d and r, the generic method of computing $N_r(f)$ is to use the quotient rule to compute the left-hand side of (3) but it does not enable us to give a precise analysis of the complexity for setting up $N_r(f)$ since it involves complicated calculations of the quotient rule.

Another way of computing $N_r(f)$ is to use connections with decimation operators on sequences over $\mathbb{F}_q$ to obtain an explicit formula for $N_r(f)$ in terms of the Berlekamp matrix $B_r(f)$ and Hankel matrices $G(f^{[r]})$ and $U(f)$ associated to the polynomial f. For comparison with ours we just state the results on the formula of $N_r(f)$ and its complexity bound in [12,13].

Theorem 3.2. *We have* $N_r(f) = G(f^{[r]})B_r(f)U(f)$.

Theorem 3.3. *The setup cost for the matrix* $N_r(f)$ *is* $O(d^\omega + (d^2 + \log r)(\log d \log\log d))$ *arithmetic operations in* $\mathbb{F}_q$, *where* $d = \deg(f)$ *and* $\omega < 2.38$ *is the exponent of fast matrix multiplication.*

4. Setup cost of the coefficient matrix

In this section, we aim to derive an explicit formula for the coefficient matrix $N_r(f)$ and use this formula to analyze its setup cost in detail. By the product rule and Lemma 2.1 we see that (3) is equivalent to the system

$$H^{(r-1)}(f^{r-1}h) = h^r. \tag{5}$$

Set $f^{r-1} = \sum_{i=0}^{(r-1)d} a_i x^i$ for $f = \sum_{i=0}^{d} f_i x^i$ and put $h = \sum_{i=0}^{d-1} h_i x^i$. Substituting these expressions into (5), we obtain, by the product rule and $\mathbb{F}_q$-linearity,

$$\begin{aligned} H^{(r-1)}(f^{r-1}h) &= \sum_{\alpha+\beta=r-1} H^{(\alpha)}(f^{r-1})H^{(\beta)}(h) \\ &= \sum_{\alpha+\beta=r-1} \sum_{i=0}^{(r-1)d} \binom{i}{\alpha} a_i x^{i-\alpha} \cdot \sum_{j=0}^{d-1} \binom{j}{\beta} h_j x^{j-\beta}. \end{aligned}$$

Rewrite it as

$$H^{(r-1)}(f^{r-1}h) = \sum_{\alpha+\beta=r-1} \sum_{k=0}^{(d-1)r} \left(\sum_{i+j=k+r-1} \binom{i}{\alpha} \binom{j}{\beta} a_i h_j \right) x^k.$$

Interchanging two outer sums in the preceding equation gives

$$H^{(r-1)}(f^{r-1}h) = \sum_{k=0}^{(d-1)r} \left(\sum_{\alpha+\beta=r-1} \sum_{i+j=k+r-1} \binom{i}{\alpha} \binom{j}{\beta} a_i h_j \right) x^k.$$

Interchanging two inner sums in the preceding equation gives

$$H^{(r-1)}(f^{r-1}h) = \sum_{k=0}^{(d-1)r} \left(\sum_{i+j=k+r-1} \sum_{\alpha+\beta=r-1} \binom{i}{\alpha} \binom{j}{\beta} a_i h_j \right) x^k.$$

Rewrite it as

$$H^{(r-1)}(f^{r-1}h) = \sum_{k=0}^{(d-1)r} \left(\sum_{j=0}^{k+r-1} \sum_{\alpha+\beta=r-1} \binom{k+r-1-j}{\alpha} \binom{j}{\beta} a_{k+r-1-j} h_j \right) x^k.$$

Since the innermost sum is the coefficient of x^{r-1} in the polynomial $(1+x)^{k+r-1}$, we have, for each $0 \leqslant j \leqslant \min\{d-1, k+r-1\}$,

$$\sum_{\alpha+\beta=r-1} \binom{k+r-1-j}{\alpha} \binom{j}{\beta} = \binom{k+r-1}{r-1}.$$

By the Lucas' congruence theorem, we see that

$$\binom{k+r-1}{r-1} \equiv \begin{cases} 1 \pmod{p} \text{ if } r|k, \\ 0 \pmod{p} \text{ otherwise.} \end{cases}$$

Now, by this property, (5) simplifies to

$$\sum_{k=0}^{d-1} \left(\sum_{j=0}^{rk+r-1} a_{rk+r-1-j} h_j \right) x^{rk} = \sum_{k=0}^{d-1} h_k^r x^{rk}. \tag{6}$$

Hence we see that (6) (hence (4)) is equivalent to the system

$$\sum_{j=\max\{r(k+1-d)+d-1,0\}}^{\min\{r(k+1)-1,d-1\}} a_{rk+r-1-j}h_j = h_k^r \text{ for } 0\leqslant k\leqslant d-1. \tag{7}$$

Thus, the matrix in (4) is given by

$$N_r(f) = \begin{pmatrix} a_{r-1} & \cdots & a_0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 \\ a_{2r-1} & \cdots & a_r & \cdots & a_0 & 0 & \cdots & 0 & \cdots & 0 \\ a_{3r-1} & \cdots & a_{2r} & \cdots & a_r & a_{r-1} & \cdots & 0 & \cdots & 0 \\ \cdot & \cdots & \cdot & \cdots & \cdot & \cdot & \cdots & \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot & \cdots & \cdot & \cdot & \cdots & \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot & \cdots & \cdot & \cdot & \cdots & \cdot & \cdots & \cdot \\ 0 & \cdots & 0 & \cdots & 0 & \cdot & \cdots & a_{(r-1)d} & \cdots & a_{(r-1)d-r} \\ 0 & \cdots & 0 & \cdots & 0 & \cdot & \cdots & 0 & \cdots & 1 \end{pmatrix}. \tag{8}$$

In case of $r = q$, the matrix in (8) reduces to the matrix which was obtained by Fleischmann and Roelse [4, Lemma 2.3]. In case of $r = 2$, which can be made if $\mathbb{F}_q$ has characteristic 2, $a_i = f_i$ for all $0\leqslant i\leqslant d$, so that (7) reduces to

$$\sum_{j=\max\{2k+1-d,0\}}^{\min\{2k+1,d-1\}} f_{2k+1-j}h_j = h_k^2 \text{ for } 0\leqslant k\leqslant d-1, \tag{9}$$

which was observed by Niederreiter [12].

We point out here that there is no setup cost for $N_2(f)$ from (8) or (9) since its entries can be read off immediately from the coefficients of the given polynomial f. Likewise, for arbitrary d and r, one can set up $N_r(f)$ with no further cost as long as the coefficients of f^{r-1} are explicitly obtained. So the setup cost for $N_r(f)$ depends only upon the explicit expansion of f^{r-1}.

As for the complexity of $N_r(f)$, we need to expand out the polynomial f^{r-1} completely. The naive method for computing the coefficients of f^{r-1} is to multiply f by itself $r-1$ times and it requires much cost simply because it does not involve polynomial reduction. But there is an efficient way of obtaining the coefficients of f^{r-1} with much less cost when compared to that of Niederreiter's (Theorem 3.3). To this end, let us now consider

$$f^{r-1}(x) = \frac{f^r(x)}{f(x)}. \tag{10}$$

Each coefficient of the numerator polynomial $f^r(x)$ can be calculated by $O(\log r)$ arithmetic operations in $\mathbb{F}_q$, so we can obtain the coefficients of f^r in $O(d\log r)$ arithmetic operations in $\mathbb{F}_q$. We note that at this stage, if a normal basis for an extension

field $\mathbb{F}_q$ over $\mathbb{F}_r$ is exploited, f^r can be calculated for free. By (10) the computation of f^{r-1} is obtained by only one division with remainder 0, applied to f^r and f. According to standard results on the complexity of polynomial divisions [1,17], the computation of f^{r-1} can be done in $O(rd \cdot \log rd \cdot \log\log rd)$ arithmetic operations in $\mathbb{F}_q$ using fast methods. To sum up the discussion above, we state the complexity bound for $N_r(f)$.

Theorem 4.1. *The setup cost for the matrix $N_r(f)$ is $O(rd \cdot \log rd \cdot \log\log rd)$ arithmetic operations in $\mathbb{F}_q$.*

We remark that the arithmetic complexity for $N_r(f)$ in Theorem 4.1 is linear both in $d = \deg(f)$ and in r ignoring log parts. The reason for this is that it does not involve the computation of the Berlekamp matrix, unlike the Niederreiter's formula for $N_r(f)$. Such observation on the complexity of $N_r(f)$ makes the probabilistic factorization algorithm of Gao and Gathen improve its run time significantly (see [5, Theorem 3.1]). For their algorithm is mainly based on the algorithm of Niederreiter and they used Kaltofen and Saunders' version [9] of Wiedermann's method [19] for solving sparse linear systems. This is the case when $\deg(f) = d$ is large compared to r.

Following the approach in [5], we now calculate the left-hand side of (5) in a slightly different way. For any $g = \sum_{i=0}^{n} g_i x^i \in \mathbb{F}_q[x]$ of degree n, we set

$$g^{\langle j\rangle}(x) := \sum_{\substack{0 \leqslant i \leqslant n \\ i \equiv j \pmod r}} g_i x^{\frac{i-j}{r}}$$

for each $0 \leqslant j \leqslant r-1$.

Then $g^{\langle j\rangle}$ is called the contracted $j \pmod r$ part of $g(x)$, and we see that $g(x) = \sum_{j=0}^{r-1} x^j g^{\langle j\rangle}(x^r)$. In particular, if $r = 2$, $g^{\langle 0\rangle}$ and $g^{\langle 1\rangle}$ are the contracted even and odd part of g as defined in [5]. Using this expression of a polynomial one can compute the left-hand side of (5) as follows.

$$\begin{aligned}
H^{(r-1)}(f^{r-1}h) &= H^{(r-1)}\left(\sum_{i=0}^{r-1} x^i (f^{r-1})^{\langle i\rangle}(x^r) \cdot \sum_{j=0}^{r-1} x^j h^{\langle j\rangle}(x^r)\right) \\
&= H^{(r-1)}\left(\sum_{k=0}^{2(r-1)} \sum_{i+j=k} x^k ((f^{r-1})^{\langle i\rangle} h^{\langle j\rangle})(x^r)\right) \\
&= \sum_{k=0}^{2(r-1)} \sum_{i+j=k} H^{(r-1)}\left(x^k ((f^{r-1})^{\langle i\rangle} h^{\langle j\rangle})(x^r)\right) \\
&= \sum_{k=0}^{2(r-1)} \sum_{i+j=k} H^{(r-1)}(x^k) \cdot ((f^{r-1})^{\langle i\rangle} h^{\langle j\rangle})(x^r)
\end{aligned}$$

$$= \sum_{k=0}^{2(r-1)} \sum_{i+j=k} \binom{k}{r-1} x^{k-r+1}((f^{r-1})^{\langle i\rangle} h^{\langle j\rangle})(x^r)$$

$$= \sum_{i+j=r-1} \left((f^{r-1})^{\langle i\rangle} h^{\langle j\rangle}\right)(x^r),$$

where the Lucas' congruence theorem applies to the last equality. From the very last identity above, the computation of $H^{(r-1)}(f^{r-1}h)$ can be done in r multiplications of polynomials of degree at most d. But the problem we face here is we should know the coefficients of f^{r-1} in advance. Without knowing those coefficients, one may use the multinomial coefficient formula to compute $(f^{r-1})^{\langle j\rangle}$ in terms of $f^{\langle i\rangle}$ but computing $H^{(r-1)}(f^{r-1}h)$ this way requires more cost than only one polynomial division applied to f^r and f.

On the other hand, if $r = 2$, then $H^{(1)}(fh) = (fh)'(x) = (f^{\langle 0\rangle} h^{\langle 1\rangle} + f^{\langle 1\rangle} h^{\langle 0\rangle})(x^2)$. From this equation, Gao and Gathen [5] observed that a suitable rearrangement of the columns of the matrix $N_2(f)$ becomes the Sylvester matrix of the contracted polynomials $f^{\langle 0\rangle}$ and $f^{\langle 1\rangle}$. We also observe that in the case where $\deg(f) = d$ is large compared to r, a suitable rearrangement of the columns of $N_r(f)$ in (8) turns out to be the Sylvester matrix of the contracted polynomials $(f^{r-1})^{\langle j\rangle}(x)$ for all $0 \leqslant j \leqslant r-1$. As in characteristic 2 in [5], it is also expected that this special feature of the coefficient matrix may result in efficient methods for finding a basis of the null space of the system corresponding to (4).

We close this paper by making several remarks on $N_r(f)$.

Remarks.

1. As Niederreiter's factorization algorithm works over any effectively computable field of positive characteristic so does the formula for $N_r(f)$ in (8).

2. It is shown in [12] that there is a close tie of $N_q(f)$ with the Berlekamp matrix $B_q(f)$ for $r = q$. Indeed, $N_q(f)$ and $B_q(f)$ are similar matrices, i.e., $B_q(f) = G(f)^{-1} N_q(f) G(f)$. From this relation one can use Theorem 4.1 to deduce that the setup cost of $B_q(f)$ is $O(d^s + qd \cdot \log qd \cdot \log\log qd)$ arithmetic operations in $\mathbb{F}_q$, where $s := \max\{\omega, 2\}$ and ω is as in Theorem 3.3. Compare this with $O((d^2 + d\log q)\log d \cdot \log\log d)$, which is the standard complexity for $B_q(f)$ with fast arithmetic applied.

3. Niederreiter gave two ways of obtaining the system of linear equations corresponding to (4). A first and easy way for this is to choose $r = q$. Another way is to use a normal basis for a nontrivial extension field of the underlying finite field. In using normal bases, we can improve the run time for setting $N_q(f)$ up significantly because computing matrix multiplication in $N_q(f)$ is avoided. See [15, Theorems 2 and 3] for details.

4. Except for $r = 2$, sparsity of f does not imply that of $N_r(f)$ but sparsity of f^{r-1} does. Thus the matrix in (8) is very useful to implementations of the factorization algorithm when the degree of f is large compared to r.

References

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Algorithms, Addision-Wesley, Reading, MA, 1974.

[2] K. Conrad, The digit principle, J. Number Theory 84 (2000) 230–257.

[3] P. Fleischmann, M.C. Holder, P. Roelse, The black-box Niederreiter algorithm and its implementation over the binary field, Math. Comp. 72 (2003) 1887–1899.

[4] P. Fleischmann, P. Roelse, Comparative implementations of Berlekamp's and Niederreiter's polynomial factorization algorithms, in: S. Cohen, H. Niederreiter (Eds.), Finite Fields and Applications, London Mathematical Society Lecture Note Series, vol. 233, Cambridge University Press, Cambridge, 1996, pp. 73–83.

[5] S. Gao, J. von zur Gathen, Berlekamp's and Niederreiter's polynomial factorization algorithms, Finite fields: theory, applications, and algorithms (Las Vegas, NV, 1993), Contemporary Mathematics, vol. 168, American Mathematical Society, Providence, RI, 1994, pp. 101–116.

[6] R. Göttfert, H. Niederreiter, Hasse–Teichmüller derivatives and products of linear recurring sequences, Finite fields: theory, applications, and algorithms (Las Vegas, NV, 1993), Contemporary Mathematics, vol. 168, American Mathematical Society, Providence, RI, 1994, pp. 117–125.

[7] H. Hasse, Theorie der höheren Differentiale in einem algebraischen Funktionenkörper mit vollkommenem Konstantenkörper bei beliebiger Charakteristik, J. Reine Angew. Math. 175 (1936) 50–54.

[8] H. Hasse, F.K. Schmidt, Noch eine Begründung der Theorie der höheren Differentialquotienten in einem algebraischen Funktionenkörper einer Unbestimmten, J. Reine Angew. Math. 177 (1937) 215–237.

[9] E. Kaltofen, B.D. Saunders, On Wiedemann's method of solving sparse linear systems, Applied algebra, algebraic algorithms and error-correcting codes (New Orleans, LA, 1991), Proc. AAECC-10, Lecture Notes in Computer Science, 539, Springer, Berlin, 1991, pp. 29–38.

[10] H. Niederreiter, A new efficient factorization algorithm for polynomials over small finite fields, Appl. Algebra Eng. Comm. Comput. 4 (2) (1993) 81–87.

[11] H. Niederreiter, Factorization of polynomials and some linear-algebra problems over finite fields, Computational linear algebra in algebraic and related problems (Essen, 1992), Linear Algebra Appl. 192 (1993) 301–328.

[12] H. Niederreiter, New deterministic factorization algorithms for polynomials over finite fields, Finite fields: theory, applications, and algorithms (Las Vegas, NV, 1993), Contemporary Mathematics, vol. 168, American Mathematical Society, Providence, RI, 1994, pp. 251–268.

[13] H. Niederreiter, Factoring polynomials over finite fields using differential equations and normal bases, Math. Comp. 62 (206) (1994) 819–830.

[14] H. Niederreiter, R. Göttfert, On a new factorization algorithm for polynomials over finite fields, Math. Comp. 64 (209) (1995) 347–353.

[15] P. Roelse, Factoring high-degree polynomials over $\mathbb{F}_2$ with Niederreiter's algorithm on the IBM SP2, Math. Comp. 68 (1999) 869–880.

[16] I.E. Shparlinski, Computational and Algorithmic Problems in Finite Fields, Kluwer Academic Publishers, Dordrecht, 1993.

[17] O. Teichmüller, Differentialrechung bei Charakteristik p, J. Reine Angew. Math. 175 (1936) 89–99.

[18] D. Wiedemann, Solving sparse linear equations over finite fields, IEEE Trans. Inform. Theory 32 (1986) 54–62.