Comparative Implementations of Berlekamp's and Niederreiter's Polynomial Factorization Algorithms

Peter Fleischmann and Peter Roelse

Institut für Experimentelle Mathematik
Universität Essen
45326 Essen
Germany
{peter, roelse}@exp-math.uni-essen.de

Abstract

New C^{++}-implementations of the classical factorization algorithms for polynomials over finite fields of Berlekamp and the new ones of Niederreiter are presented. Their performances on various types of inputs are compared.

1 Introduction

The basic problem of factorizing univariate polynomials over the finite field $\mathbf{F}_q$ has got new impulses in the past few years with a new linearization technique developed by Niederreiter in [8], [9], [10]. Unlike Berlekamp's classical approach, which uses the Frobenius fixed point algebra in $A := \mathbf{F}_q[X]/(f)$ (where f is the polynomial to be factored), Niederreiter's method is based on the analysis of the solution space of certain differential equations in the field of rational functions $\mathbf{F}_q(X)$.
From the very beginning there have been several striking similarities between Niederreiter's and Berlekamp's algorithms in each step. Suppose for simplicity that the polynomial is monic and squarefree. Then in both algorithms a certain system of linear equations has to be set up and solved, leading to an $\mathbf{F}_q$ - subspace $\mathcal{S}$ of A, whose dimension coincides with the number of irreducible factors of f. Now the elements of $\mathcal{S}$ can be used to extract the

irreducible factors of f by suitable gcd operations.
Niederreiter's algorithm has the following practical advantages: In the case of small fields the linear equations to be solved can be set up very efficiently. In particular in $\mathbf{F}_2$ they can be read off directly from the coefficients of f. Moreover in characteristic 2 a sparse polynomial f leads to a sparse system of linear equations. On the other hand Niederreiter's solution space is a mere vectorspace whereas that of Berlekamp's algorithm is an $\mathbf{F}_q$ algebra. This additional structure has proved to be very valuable especially for randomized versions, which dramatically accelerate the process of extracting the irreducible factors in the last step of the algorithm for 'large' fields. Soon after Niederreiter's discovery several authors [3], [4], [7], [6] independently found the theoretical connection between both algorithms, which explains their similarities but also shows how to combine their advantages. This yields for example a randomized 'big field' version of Niederreiter's algorithm (see [4]).
The main purpose of this paper is an experimental study and comparison of the practical performance of both methods and their randomized variants.

2 Basic concepts

Let $\mathbf{F}_q$ be the finite field with $q = p^s$ elements and $f \in \mathbf{F}_q[X]$ a monic polynomial of degree $d \geq 1$. For simplicity we assume that f is squarefree, i.e.

$$f = g_1 \cdot g_2 \dots g_t$$

with $g_i \in \mathbf{F}_q[X]$ pairwise different monic irreducible polynomials of degree d_i. The assumption on f to be squarefree is not a serious restriction, since efficient methods exist to reduce the factorization problem to the squarefree case.

The purpose is to determine the g_i from the given f. The chinese remainder theorem states that for all $i \in \{1, 2, ..., t\}$ there is $e_i \in \mathbf{F}_q[X]$ with $\deg e_i < d$ and

$$e_i \equiv \begin{cases} 1 \bmod \; g_i & \\ 0 \bmod \; g_j & \forall\, j \neq i \end{cases} .$$

Hence $\mathcal{A} := \mathbf{F}_q[X]/(f) \cong \oplus_{i=1}^t \mathbf{F}_q[X]/(g_i)$, $\mathbf{F}_q[X]/(g_i) \cong \mathbf{F}_{q^{d_i}}$, $d_i := \deg g_i$.

In particular $\mathcal{A}$ is a semisimple $\mathbf{F}_q$ - algebra with primitive idempotents $\bar{e}_i := e_i + (f)$ and 'Wedderburn - components' $\mathcal{A}\bar{e}_i \cong \mathbf{F}_{q^{d_i}}$.

The principal computational tool for splitting polynomials is the euclidian algorithm since $\gcd(f, h)$ is a nontrivial divisor of f whenever $h + (f)$ is not a unit in $\mathcal{A}$.

In order to find non - units of $\mathcal{A}$ efficiently it is important to reduce the data on which one has to search.

2.1 Equivalent subspaces

The following concept has been introduced in [6]:

Definition 2.1 *A subspace $S \leq \mathcal{A}$ is said to* **factor** f **completely** *if there exist elements $s_1, \ldots, s_t \in S$, such that for each $i \in \{1, \ldots, t\}$ there is a subset $m_i \subseteq \underline{t} := \{1, \ldots, t\}$ with $g_i = \gcd(s_j, f \mid j \in m_i)$. Two subspaces $S_1, S_2 \leq \mathcal{A}$ are* **equivalent** *if and only if there is a unit $u \in \mathcal{A}^*$ such that $S_2 = S_1 \cdot u$.*

If $S_1, S_2 \leq \mathcal{A}$ are equivalent then of course S_1 factors f completely if and only if so does S_2. Clearly the subspace $\mathcal{S}$ generated by the set $\{f/g_i \mid i \in \underline{t}\}$ factors f completely but for practical use, the main task is to find subspaces that can be described independently of knowing the factors g_i.

Let $\mu + (f)$ be the unique solution in $\mathcal{A}$ of the system of congruences $\mu \equiv f/g_k \bmod g_k$ for $1 \leq k \leq t$. Then $\mu+(f)$ is a unit and $f/g_i \equiv \mu e_i \mathrm{mod}\ f$. This shows that the subspaces $\mathcal{S}$ and $\mathcal{B} := \langle e_i + (f) \ |1 \leq i \leq t\rangle_{\mathbf{F}_q}$ are equivalent. Moreover if

$$F : \mathcal{A} \to \mathcal{A},\ \bar{h} \mapsto \bar{h}^q$$

denotes the Frobenius - automorphism of $\mathcal{A}$, then it is immediately clear that

$$\mathcal{B} = \oplus_{i=1}^t \mathbf{F}_q \overline{e_i} = \oplus_{i=1}^t (\mathcal{A}\overline{e_i})^F = \mathcal{A}^F = \{\bar{h} \in \mathcal{A} \mid \bar{h} = \bar{h}^q\}.$$

Hence the 'Berlekamp - algebra' $\mathcal{B}$ can be expressed as the $\mathbf{F}_q$ - algebra of F - fixed points on $\mathcal{A}$, i.e. without any knowledge on the g_i's. Moreover we see that all subspaces which are equivalent to $\mathcal{S}$ have $\mathbf{F}_q$ - dimension t, the number of irreducible factors.

2.2 Extracting factors and randomization

Notice that $\mathcal{B}$ is not only a vector space but also an $\mathbf{F}_q$ - algebra. This additional structure can be used to accelerate the process of extracting factors of f once a subspace equivalent to $\mathcal{B}$ has been constructed: Suppose we have constructed a subspace S which is equivalent to $\mathcal{B}$. Each element $b \in \mathcal{B}$ is of the form $b = \sum_{i=1}^t c_i \overline{e_i}$ with $c_i \in \mathbf{F}_q$ and therefore satisfies $f = \prod_{c \in \mathbf{F}_q} \gcd(b - c, f)$, involving a nontrivial factorization as long as $b \notin \mathbf{F}_q \cdot 1_A$. If $S = \mathcal{B} \cdot u$

with the unit $u \in \mathcal{A}^*$, then $\gcd(bu-cu, f)$ is associated to $\gcd(b-c, f)$. Hence for any $s \in S$

$$f = \prod_{c \in \mathbf{F}_q} \gcd(s - cu, f)$$

involving a nontrivial factorization iff $s \notin u\mathbf{F}_q \cdot 1_A$.

In the case of large field sizes one can use an efficient randomization, due to Cantor and Zassenhaus [2]:

If the characteristic of $\mathbf{F}_q$ is odd and f is not irreducible then, with a chance of $\geq 1/2$, a random choice of $b \in \mathcal{B}$ leads to such an element that has at least two coefficients c_{i_1}, c_{i_2} that are not both squares or nonsquares. In this case the element $b^{\frac{q-1}{2}} - 1$ is a nonunit in $\mathcal{A}$ and $1 \neq \gcd(b^{\frac{q-1}{2}} - 1, f)$ is a nontrivial factor of f. Again $\gcd((bu)^{\frac{q-1}{2}} - u^{\frac{q-1}{2}}, f)$ is associated to $\gcd(b^{\frac{q-1}{2}} - 1, f)$ and random choices $s \in S$ efficiently lead to factors of f by taking

$$\gcd(s^{\frac{q-1}{2}} - u^{\frac{q-1}{2}}, f).$$

For characteristic two one has to adapt this method because all elements are squares in that case. For example if f is not irreducible, then in Berlekamp's algorithm, computing $\gcd(\mathrm{Tr}_{\mathbf{F}_q|\mathbf{F}_2}(b), f)$ will lead to a nontrivial factor with probability $\geq 1/2$.

It is clear from the remarks above that those subspaces that are equivalent to $\mathcal{B}$ are of particular importance to factorize polynomials. By definition these subspaces form one orbit $\mathcal{O}$ under the action of the group $\mathcal{A}^*$ on the set of all $\mathbf{F}_q$ - subspaces of $\mathcal{A}$ given by multiplication. Since the stabilizer $Stab_{\mathcal{A}}(\mathcal{B})$ coincides with the group of units $\mathcal{B}^*$ we see immediately (compare [6] thm. 6.1):

$$|\mathcal{O}| := \frac{|\mathcal{A}^*|}{|\mathcal{B}^*|} = \prod_{i=1}^{t} \frac{q^{d_i} - 1}{q - 1}.$$

Of course the elements of $\mathcal{B} \cdot u$ are solutions of the equation

$$h^q = hu^{q-1} \bmod f.$$

This could be used to set up a system of linear equations and to determine a basis for the corresponding nullspace. But to be of practical interest, i.e. outperform Berlekamp's algorithm, one has to find a unit wich leads to a system that can be set up or eliminated faster.

2.3 Berlekamp's and Niederreiter's Subspaces

In Berlekamp's algorithm, a basis of $\mathcal{B}$ is constructed by explicitly computing the F - fixed point space in $\mathcal{A}$: If Q denotes the matrix representation of F

with respect to the basis $\{1, \bar{x}, \bar{x}^2, ..., \bar{x}^{d-1}\}$ of $\mathcal{A}$ then $Q = (q_{k,l})_{k,l \in \{0,1,...,d-1\}}$ with

$$x^{qk} \equiv \sum_{l=0}^{d-1} q_{k,l} \cdot x^l \mod f.$$

Since $\mathcal{B} = Kern(F - id)$ we get:

$$\bar{h} = \sum_{l=0}^{d-1} h_l \cdot \bar{x}^l \in \mathcal{B} \iff \mathbf{h} \circ (Q - E) = 0,$$

where $\mathbf{h} := (h_0, h_1, ..., h_{d-1})$ and E is the $d \times d$ - identity matrix.

The starting point of Niederreiter's algorithm is the following modular linear differential equation in the field of rational functions $\mathbf{F}_q(x)$:

$$H^{(q-1)}(y) - y^q = 0. \tag{1}$$

Here $H^{(k)}$ is the **Hasse - Teichmüller - derivative**, defined in the field of formal Laurent series in x^{-1} over $\mathbf{F}_q$:

$$H^{(k)}(\sum_{n=j}^{\infty} s_n x^{-n}) = \sum_{n=j}^{\infty} \binom{-n}{k} s_n x^{-n-k}, \ j \in \mathbf{Z}.$$

The following theorem gives the connection to polynomial factorization:

Theorem 2.2 *(Niederreiter): The set of solutions of Equation 1, with fixed denominator f form an $\mathbf{F}_q$ - space $\mathcal{N}$ of elements of degree ≤ -1:*

$$\mathcal{N} = \{ \ y = \sum_{i=1}^{t} c_i \cdot \frac{g_i{}'}{g_i} \mid c_1, ..., c_t \in \mathbf{F}_q \ \}.$$

These solutions satisfy the equation $f^q \cdot H^{(q-1)}(\frac{h}{f}) = h^q$. Both sides of this equation are polynomials in x^q, hence a comparison of coefficients at x^q yields a system of linear equations with coefficient matrix N such that for $h = h_0 + h_1 x + ... + h_{d-1} x^{d-1}$ and $\mathbf{h} := (h_0, ..., h_{d-1})$:

$$h/f \in \mathcal{N} \iff (N - E)(\mathbf{h})^t = 0.$$

From this it is possible to compute a basis of $\mathcal{L} := f\mathcal{N}/(f)$ without knowing the irreducible factors g_i of f. In particular $\mathcal{L}$ is the kernel of the linear operator $\mathbf{n} := f^q H^{(q-1)}(\cdot/f) - F$ on $\mathcal{A}$, whereas $\mathcal{B} = \ker \mathbf{b}$ with $\mathbf{b} := F - id$.

Lemma 2.3 *Let f be monic of degree d and $f^{q-1} =: \gamma = \gamma_0 + \gamma_1 x + \cdots + x^{d(q-1)}$, with $\gamma_j = 0$ if $j < 0$ or $j > d(q-1)$. Then Niederreiter's coefficient matrix N is given by*

$$\begin{pmatrix} \gamma_{q-1} & \gamma_{q-2} & \cdots & \gamma_{q-d+1} & \gamma_{q-d} \\ \gamma_{2q-1} & \gamma_{2q-2} & \cdots & \gamma_{2q-d+1} & \gamma_{2q-d} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \gamma_{(d-1)q-1} & \gamma_{(d-1)q-2} & \cdots & \gamma_{(d-1)q-d+1} & \gamma_{(d-1)q-d} \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

.

Proof: Use

$$\binom{nq}{i} \equiv 0 \bmod p$$

for $n \geq 0$, $i \in \{1, 2, \ldots, q-1\}$ and the product rule for the Hasse - Teichmüller - derivative to show that

$$f^q H^{(q-1)}(h/f) = \sum_{i=0}^{q-1} H^{(i)}(f^q) H^{(q-1-i)}(h/f) = H^{(q-1)}(f^{q-1}h).$$

After this, use the definition of the Hasse - Teichmüller - derivative and

$$\binom{n+q-1}{q-1} \equiv \begin{cases} 1 \bmod p & \text{for } n \equiv 0 \bmod q \\ 0 \bmod p & \text{for } n \not\equiv 0 \bmod q \end{cases}$$

to read off the coefficients of the matrix. $\circ$

For $q \leq d$ the entries of this matrix can be determined very efficiently by computing f^q/f. If $q > d$, not all coefficients of f^{q-1} are needed but at the moment it is unclear if there is an efficient way to compute only the coefficients which are needed.

In [3] it has been shown that for any field K and $g, h \in K[X]$ with $\deg h < \deg g$ one has

$$H^{(k)}(\frac{h}{g}) \cdot g^{k+1} \;\equiv\; (-1)^k \cdot h \cdot (g')^k \bmod g.$$

From this it follows immediately that $\mathbf{n} = (\cdot(-f'^q)) \circ \mathbf{b} \circ (\cdot f'^{-1})$ hence

$$\mathcal{L} = \mathcal{B} \cdot f'.$$

Hence $\mathcal{L}$ and $\mathcal{B}$ are equivalent (see also [6],[7]). Now the methods of section 2.2 can be used to extract factors using $\mathcal{L}$ or $\mathcal{B}$.

3 The implementations

The algorithms of Berlekamp and Niederreiter were implemented for prime fields and for fields of characteristic two. In addition, the algorithm of Cantor and Zassenhaus based on a distinct degree factorization was implemented. After the distinct degree factorization, only polynomials consisting of factors of the same degree have to be considered. For these polynomials the randomized approach can be applied directly. For detailed information, refer to [2]. In order to apply the costly linear algebra algorithms to matrices of formats as small as possible our implementations always reduce the factorization problem to the squarefree case.

The implementation was mainly written in C^{++} (a part of it was written in C). The prime fields and the extension fields of the binary field were both implemented as a C^{++} *class*. One field element is stored in a 32 bit word. The polynomials are arrays of field elements. The arithmetic was implemented as a *template*, which has the advantage that the same functions can be used easily for different fields. To avoid physical copies of polynomials, pointers with reference counters are used. For the polynomials over $\mathbf{F}_2$, a different (C) implementation is used. In this case polynomials are packed in 32 bit words. The memory management in this case works as follows: if a polynomial is reused and enough memory is already allocated, nothing is done. In the other case memory is reallocated. This effectively reduces overhead for memory allocation and deallocation.

In the prime fields, multiplications and divisions are done with the help of tables. One table is a discrete logarithm table with respect to a generator of $\mathbf{F}_q^*$. The other table holds the powers of this generator. The multiplication (division) is therefore reduced to an integer addition (subtraction) and three table look-ups.

Polynomial multiplication and division are performed with the classical algorithms and gcd's are computed with the Euclidian algorithm. Evaluation of powers is done with the left-to-right repeated squaring method, e.g. α^{13}, $(13 = (1101)_2)$, is computed as

$$(((\alpha^2)\alpha)^2)^2\alpha.$$

The run times given in the tables were obtained on an IBM RISC System/6000 with a POWER processor, rated at 62.5 Mhz. The programs were compiled with the IBM xlC compiler with optimization level -O3. All times are in seconds and represent total run-times for factoring 25 pseudorandomly selected polynomials. This was done to get some sort of average behaviour.

In the case of Berlekamp's and Niederreiter's algorithms, also two intermediate times are given. The first time (1) is taken at the point where the matrix is built. The second one (2) is taken at the point were a basis for the subspace is determined. The final one (3) represents the complete run-time of the algorithms. For the Cantor-Zassenhaus algorithm, only the total run-time is given. We just mention here that almost all of this time is spent in the distinct degree factorization. The comparatively long run times are probably due to the fact that we are dealing with small polynomial degrees here.

In Table 1 the results for $\mathbf{F}_2$ are given. It can be seen that, despite the fact that the Niederreiter matrix can be set up more efficiently than the Berlekamp matrix, the Berlekamp algorithm has the best performance. The reason for this is that the Berlekamp matrix is better suited for elimination. (Approximately half of the elements of both matrices are known to be zero in advance, but these zeros in Berlekamp's (transposed) matrix are all situated in the left half of the matrix, while those of Niederreiter's matrix are situated in the lower left and the upper right corner.) The same kind of behaviour also happens in other very small fields.

Table 2 shows that the setup of the Niederreiter matrix is relatively expensive for $q > d$, since not all coefficients of f^{q-1} are needed in this case. For $q \leq d$, the setup is very efficient and makes the Niederreiter algorithm outperform Berlekamp's. The 'small' field version refers to a deterministic final stage of the algorithm, while the 'big' field version refers to a randomized final stage of the algorithm. For this relatively small prime, the randomized versions already outperform the deterministic versions.

For the Niederreiter algorithm mentioned in Table 3, a linearization over $\mathbf{F}_2$ is used. If m is the degree of extension of the binary field, then this approach leads to a $md \times md$ matrix over $\mathbf{F}_2$, instead of a $d \times d$ matrix over $\mathbf{F}_{2^m}$in Berlekamp's case. For this linearization over $\mathbf{F}_2$ a polynomial basis is used. The setup of the large matrix is relatively fast in Niederreiter's algorithm, since at most $(d+2)m^2$ entries have to be computed.

Of course Berlekamp's algorithm can also be linearized over $\mathbf{F}_2$, but it was already shown in [5] that this runs slower than the Niederreiter algorithm. Table 4 shows that a linearization over $\mathbf{F}_2$ should not be used for very large extensions, because the computation of the nullspace of the large matrix takes too much time.

d		Berlekamp	Niederreiter	Cantor-Zass.
500	(1)	0.84	0.29	
	(2)	3.42	3.94	
	(3)	3.47	4.12	9.18
1000	(1)	3.98	0.74	
	(2)	20.0	25.0	
	(3)	20.2	25.7	52.0
1500	(1)	9.59	1.46	
	(2)	50.0	69.2	
	(3)	50.9	69.7	122
2000	(1)	17.2	2.33	
	(2)	100	151	
	(3)	101	152	360

Table 1: 25 pseudorandom polynomials of degree d in $\mathbf{F}_2[x]$.

d		Berlekamp (small-big)	Niederreiter (small-big)	Cantor-Zass.
25	(1)	0.58	0.98	
	(2)	0.78	1.07	
	(3)	1.89-1.20	2.72-1.64	1.27
50	(1)	3.90	3.51	
	(2)	4.76	4.37	
	(3)	7.38-6.78	10.5-6.84	8.36
75	(1)	12.6	7.60	
	(2)	15.5	10.3	
	(3)	22.5-19.8	19.9-16.7	25.9
100	(1)	24.6	13.4	
	(2)	32.1	20.2	
	(3)	42.4-39.5	37.1-30.0	59.9
150	(1)	56.0	29.7	
	(2)	79.3	52.4	
	(3)	98.2-95.9	82.6-76.6	222
200	(1)	98.6	52.3	
	(2)	153	107	
	(3)	184-181	152-147	504

Table 2: 25 pseudorandom polynomials of degree d in $\mathbf{F}_{97}[x]$.

d		Berlekamp	Niederreiter	Cantor-Zass.
10	(1)	0.48	0.97	
	(2)	0.51	1.26	
	(3)	1.50	1.37	1.29
20	(1)	2.10	2.58	
	(2)	2.35	4.36	
	(3)	5.97	4.74	8.50
30	(1)	5.63	5.25	
	(2)	6.79	11.9	
	(3)	12.9	12.8	16.5
50	(1)	22.9	12.6	
	(2)	28.3	37.1	
	(3)	50.7	38.7	68.4
75	(1)	73.3	27.2	
	(2)	88.8	94.7	
	(3)	135	98.4	186
100	(1)	168	45.9	
	(2)	206	194	
	(3)	280	201	498

Table 3: 25 pseudorandom polynomials of degree d in $\mathbf{F}_{2^{20}}[x]$.

m		Berlekamp	Niederreiter	Cantor-Zass.
10	(1)	11.4	1.90	
	(2)	14.0	5.66	
	(3)	20.0	6.07	21.6
20	(1)	22.9	12.6	
	(2)	28.3	37.1	
	(3)	50.7	38.7	68.4
30	(1)	36.0	40.9	
	(2)	43.6	109	
	(3)	77.8	110	159

Table 4: 25 pseudorandom polynomials of degree 50 in $\mathbf{F}_{2^m}[x]$.

4 Concluding remarks

The advantages of Niederreiter's algorithm mentioned in the introduction can be realized most efficiently in the following situations:
(i) Factorization over $\mathbf{F}_q$ for $q \leq d$. Here the faster setup of the matrix pays off most. But notice that if q is very small then Berlekamp's matrix is better suited for elimination which may lead to shorter total running times.
(ii) Factorization over $\mathbf{F}_{2^m}$ using a linearization over $\mathbf{F}_2$. However, due to the size of the matrices it is only recommendable for relatively small extension degrees.

References

[1] E.R. Berlekamp, *Factoring polynomials over finite fields*, Bell System Tech. J. **46**, (1967), 1853 - 1859

[2] D.G. Cantor, H. Zassenhaus, *A new algorithm for factoring polynomials over finite fields*, Math. of Comp. **36**(154), (1981), 587 - 592

[3] P. Fleischmann, *Connections between the algorithms of Berlekamp and Niederreiter for factoring polynomials over finite fields*, Linear Algebra Appl. **192** (1993), 101-108

[4] S. Gao, J. von zur Gathen, *Berlekamp's and Niederreiter's polynomial factorization algorithms*, Contemporary Math. **168**, (1994), 101-116

[5] J. Gerhard, *Faktorisieren von Polynomen über $\mathbf{F}_q$: ein Vergleich neuerer Verfahren*, Diplomarbeit, Erlangen, (1994)

[6] T.C.Y. Lee, Scott A. Vanstone, *Subspaces and polynomial factorizations over finite fields*, AAECC **6**, (1995), 147-157

[7] V.S. Miller *On the factorization method of Niederreiter*, preprint

[8] H. Niederreiter, *A new efficient factorization algorithm for polynomials over small finite fields* AAECC **4**, (1993), 81-87

[9] H. Niederreiter, *Factorization of polynomials and some linear algebra problems over finite fields* Linear Algebra Appl. **192**, (1993), 301-328

[10] H. Niederreiter, *New deterministic factorization algorithms for polynomials over finite fields*, Contemporary Math. **168**, (1994), 251-268