

A Note on Gröbner Bases and Berlekamp's Algorithm

Manuel A. Insua and Manuel Ladra

Departamento de Álgebra, Universidad de Santiago, E-15782, Spain

Abstract

It will be shown that it is possible to factorize square-free polynomials over finite fields using Gröbner bases and a MAPLE implementation of this method will be described.

Key words: Gröbner bases, finite fields, factorization of polynomials, Berlekamp's algorithm, Maple.

1 Introduction

Gröbner bases show their great adaptability to solve many different problems in several areas of mathematics, in this occasion we show how they can be utilized in the factorization of square-free polynomials in one variable over finite fields.

The factorization of polynomials over finite fields, apart from its intrinsic mathematical interest, plays a crucial role in many topics of mathematics, especially in cryptography (enormously fomented since internet's popularization), number theory and coding theory.

On the other hand, knowing how to factorize polynomials over finite fields is very important, because it is the initial point to another algorithms of polynomial factorization into more complexes rings as: $Q[x]$, $Q[x_1, \dots, x_n]$, $F_q[x_1, \dots, x_n]$, $Q(\alpha)[x]$, $Q(\alpha)[x_1, \dots, x_n]$ [13].

¹ *E-mail address:* ladra@usc.es (M. Ladra)

² The second author was supported by the MCYT (Spain), BMF 2003-04686-C02 (European FEDER support included).

In general, factorization methods existing in present time can be classified in two parts perfectly distinguished: deterministic techniques and probabilistic techniques.

While deterministic techniques are based on classical algorithms, probabilistic ones are based on algorithms that use probabilistic methods to obtain the wished result. Using a deterministic method guarantees that the solution will always arrive (at least in theory, it may happen that the number of operations is so high that makes it impossible).

If we use a probabilistic method, the algorithm could fail because the probability of having a mistake is greater than zero. Certainly, this probability is always searched to be the smallest, one way to achieve it is running the algorithm several times. This little "trick" reduces the probability of the algorithm failure [13]. This little "trick" is used when Cantor-Zassenhaus algorithm is applied [3].

Berlekamp's algorithm [1], subject of this article, and Niederreiter's algorithm [10] are two examples of deterministic techniques of factorization. Cantor-Zassenhaus [3], Las Vegas type Berlekamp [2], von zur Gathen-Shoup [12] and Kaltofen-Shoup [7] algorithms are examples of factorization probabilistic techniques.

The interest of this article is centered on Berlekamp's algorithm. This algorithm provides the factorization of a square-free polynomial, which is the real difficulty when a polynomial is factorized.

The use of Gröbner bases to factorize polynomials is not novel. In fact, for bivariate polynomials over finite fields, surprisingly Gröbner bases are useful in practice [11].

The aim of this article is to use Gröbner bases to obtain the greatest common divisors needed to apply Berlekamp's algorithm, which at last, will provide the factorization of the polynomial we had in the beginning.

Next, in the second section, Berlekamp's algorithm is briefly exposed. In the third section, the theorem that allows to obtain the factorization of a polynomial using Gröbner Bases theory is exposed and proved; and in the fourth and last section, a Maple package which calculates the factorization of a square-free polynomial in one variable over a finite field using this kind of techniques is shown.

2 Brief description of Berlekamp's Algorithm

Let F_q be a finite field of q elements and characteristic p , and $a(x) \in F_q[x] - \{0\}$ of degree n , monic, greater than zero and square free.

Lemma 1 [1] *Let $W = \{[v(x)] \in \frac{F_q[x]}{\langle a(x) \rangle} \mid [v(x)]^q = [v(x)]\}$. Then W is an F_q -vector subspace of $\frac{F_q[x]}{\langle a(x) \rangle}$ whose dimension is the number of irreducible factors in the factorization of $a(x)$.*

Corollary 2 [1] *$a(x)$ is irreducible if and only if $\dim_{F_q} W = 1$.*

Definition 3 *Let Q be the matrix which has as rows (and in this order), the residues of $x^0, x^q, \dots, x^{(n-1)q}$ modulo $a(x)$. The Matrix Q representing the Frobenius map is called Berlekamp-Petr matrix.*

Theorem 4 [1]

$$W = \{(v_0, \dots, v_{n-1}) \mid (v_0, \dots, v_{n-1}) \cdot (Q - I) = 0\}.$$

Theorem 5 [1] *Let $a(x)$ be a square free polynomial in $F_q[x]$ and $v(x)$ be a non-constant polynomial in W . Then*

$$a(x) = \prod_{c \in F_q} \text{GCD}(a(x), v(x) - c).$$

Berlekamp algorithm [1] consists on:

- Calculating an F_q -basis for W ($\{v_1(x) = 1, v_2(x), \dots, v_k(x)\}$).
- Applying Theorem 5, taking $v(x) = v_2(x)$.
- If $a(x)$ factorizes in a product of k factors, then the process stops.

If it were not like this, Theorem 5 would be newly applied to each factor $a(x)$ previously obtained, taking $v(x) = v_3(x)$; and so on.

3 Berlekamp with Gröbner Bases

Calculating an F_q -basis for W is the first point to begin with Berlekamp algorithm. Thanks to Theorem 4 it is known that solving the following lineal system

$$(x_1, \dots, x_n) \cdot (Q - I) = (0, \dots, 0)$$

is enough to obtain an F_q -basis for W .

From a certain point of view, it could be said that Berlekamp's algorithm is the reiterate application of Theorem 5. The biggest difficulty in this result is calculating all the greatest common divisors.

We can obtain with a Gröbner basis the greatest common divisor of n polynomials in one variable, so it seems that there must be a connection between Gröbner bases theory and Theorem 5. Lazard in [9] presents some structural properties of bivariate Gröbner bases, and Czichowski [5], in a similar problem to the treated here but in a very different context, uses the structure theorem of [9] for the calculation of integrals of rational functions over a field of characteristic zero. Using ideas of [9] and Theorem 5 we obtain the following theorem. Some parts can also be obtained from the results of [9].

Theorem 6 *Let $a(x), v(x) \in F_q[x]$, with $a(x)$ monic square-free and $\deg(a(x)) = n \geq 1$, $v(x) \in W$, $I = \langle a(x), v(x) - z \rangle \subset F_q[x, z]$, and $G = \{P_1(x, z) = R_1(z) \cdot x^{n_1} + \dots, P_2(x, z) = R_2(z) \cdot x^{n_2} + \dots, \dots, P_m(x, z) = R_m(z) \cdot x^{n_m} + \dots\}$ be the reduced Gröbner basis of I , respect to the lexicographic monomial order with $x > z$. It is supposed that, moreover, G is ordered respect to its principal terms in ascendant order. Then*

- (1) $R_{k+1}(z) | R_k(z)$, $k = 1, \dots, m - 1$.
- (2) $R_m(z) \in F_q$.
- (3) $P_1(x, z) = R_1(z)$ and $R_1(z)$ is the radical of $\text{Resultant}_x(a(x), v(x) - z)$. Moreover, $\text{Resultant}_x(a(x), v(x) - z)$ has all its roots in F_q .
- (4) $P_k(x, z) = R_k(z) \cdot S_k(x, z)$, $k = 1, \dots, m$ being $S_k(x, z)$ a monic polynomial respect to its biggest power in x ($S_k(x, z) = x^{n_k} + \dots$).
- (5) If $Q_k(z)$, $k = 1, \dots, m - 1$, denotes the polynomial defined by $R_k(z) = Q_k(z) \cdot R_{k+1}(z)$, then

$$a(x) = \prod_{k=1, \dots, m-1} \left(\prod_{c/Q_k(c)=0} S_{k+1}(x, c) \right).$$

PROOF.

(1) Let $k \in \{1, 2, \dots, m - 1\}$, as $n_k < n_{k+1}$ and G is the reduced Gröbner basis of the ideal I , then $\deg(R_k(z)) > \deg(R_{k+1}(z))$. There are $A(z), B(z) \in F_q[z]$ such that $A(z) \cdot R_k(z) + B(z) \cdot R_{k+1}(z) = \text{GCD}(R_k(z), R_{k+1}(z))$.

Let $P(x, z) = A(z) \cdot x^{n_{k+1}-n_k} \cdot P_k(x, z) + B(z) \cdot P_{k+1}(x, z) = \text{GCD}(R_k(z), R_{k+1}(z)) \cdot x^{n_k+1} + \dots$. Since $P(x, z) \in I$, then $P(x, z) \xrightarrow{G} 0$. The principal term of

$P(x, z)$ is $GCD(R_k(z), R_{k+1}(z)) \cdot x^{n_{k+1}}$ and it can only be divisible by the principal term of $P_{k+1}(x, z)$. Then, it is verified that $GCD(R_k(z), R_{k+1}(z)) = R_{k+1}(z)$, and so $R_{k+1}(z) | R_k(z)$ for any $k, k \in \{1, 2, \dots, m-1\}$.

(2) Since $a(x) \rightarrow_G 0$ and the principal term of $a(x)$ is x^n , then $R_m(z)$ must be constant.

(3) I is an ideal zero-dimensional and it is radical since $a(x)$ is square free and $v(x) - z$ is linear in z [8]. Taking the order of G and the Elimination Theorem [4] into account, we can conclude that $P_1(x, z) = R_1(z)$ and $I_1 = \langle R_1(z) \rangle$, where $I_1 = I \cap F_q[z]$. $\text{Resultant}_x(a(x), v(x) - z) \in I \cap F_q[z]$ and so $R_1(z) | \text{Resultant}_x(a(x), v(x) - z)$.

Let $x_1, \dots, x_n$ be the n different roots of $a(x)$ (possibly in the algebraic closure of F_q). Since $v(x) \in W$, i. e., $v^q(x) \equiv v(x) \pmod{a(x)}$ then $v(x_i) \in F_q, i \in \{1, \dots, n\}$, and so $\text{Resultant}_x(a(x), v(x) - z) = \prod_{i=1, \dots, n} (v(x_i) - z)$ [6] has all its roots in F_q .

(4) We will check that $P_k(x, z) = R_k(z) \cdot S_k(x, z)$ for $k = 1, \dots, m$.

This result will be proved by induction.

For $k = 1$, $P_1(x, z) = R_1(z) \cdot S_1(x, z)$, being $S_1(x, z) = 1$ (see 3).

It is supposed that the result is true for k . Let $Q_k(z) \in F_q[z]$ such that $R_k(z) = Q_k(z) \cdot R_{k+1}(z)$.

It is defined

$$P(x, z) = Q_k(z) \cdot P_{k+1}(x, z) - x^{n_{k+1}-n_k} \cdot P_k(x, z) \in I$$

$$P(x, z) \rightarrow_G 0, \text{ and even more, } P(x, z) \rightarrow_{\{P_1, \dots, P_k\}} 0$$

Then

$$Q_k(z) \cdot P_{k+1}(x, z) = \sum_{i=1, \dots, k} c_i(x, z) \cdot P_i(x, z)$$

$$P_{k+1}(x, z) = \underbrace{\frac{R_k(x, z)}{Q_k(z)}}_{R_{k+1}(z)} \cdot \sum_{i=1, \dots, k} c_i(x, z) \cdot \underbrace{\frac{P_i(x, z)}{R_k(x, z)}}_{R_k(z) | R_i(z) \quad i = 1, \dots, k-1}$$

and so $R_{k+1}(z) | P_{k+1}(x, z)$.

(5) Let $k \in \{1, \dots, m-1\}, c \in F_q$ such that $Q_k(c) = 0$. We will check first that $S_{k+1}(x, c) = GCD(a(x), v(x) - c)$.

Let $j > k$, it is defined

$$P(x, z) = Q_j(z) \cdot P_{j+1}(x, z) - x^{n_{j+1}-n_j} \cdot P_j(x, z) \longrightarrow_G 0$$

Then

$$Q_j(z) \cdot P_{j+1}(x, z) = \sum_{i=1, \dots, j} c_i(x, z) \cdot P_i(x, z),$$

evaluated in c is

$$P_{j+1}(x, c) = \sum_{i=k+1, \dots, j} \frac{c_i(x, c)}{Q_j(c)} \cdot P_i(x, c)$$

And so it can be concluded that

$$P_{k+1}(x, c) | P_j(x, c), \quad j \in \{k+1, \dots, m-1\}.$$

Therefore

$$P_{k+1}(x, c) = GCD(\{P_j(x, c)\}_{j=k+1}^m),$$

or if a monic polynomial is chosen

$$S_{k+1}(x, c) = GCD(\{P_j(x, c)\}_{j=k+1}^m).$$

As $\{P_1(x, z), \dots, P_m(x, z)\}$ is a Gröbner basis of $I = \langle a(x), v(x) - z \rangle$, it is verified that

$$\langle P_1(x, c), \dots, P_m(x, c) \rangle = \langle a(x), v(x) - c \rangle.$$

This fact, together with Theorem 5, provides the searched result. $\square$

After this theoretical development, we need to explain the power of this result with an example.

Example 7 Factorize the square-free polynomial $a(x) = x^4 + 2 \in Z_3[x]$.

Note: Take into account that we are working in the quotient ring $\frac{Z_3[x]}{\langle a(x) \rangle}$, which is also an F_3 -vector space with basis $\{1, x, x^2, x^3\}$; and that is why we will indistinctly talk about vector (a, b, c, d) or polynomial $a + bx + cx^2 + dx^3$.

Berlekamp-Petr Matrix

<i>Monomial</i>	<i>Residue modulo $a(x)$</i>
x^0	$(1, 0, 0, 0)$
x^3	$(0, 0, 0, 1)$
x^6	$(0, 0, 1, 0)$
x^9	$(0, 1, 0, 0)$

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

A Z_3 -basis for W is $\{(1, 0, 0, 0), (0, 0, 1, 0), (0, 1, 0, 1)\}$, and so the number of irreducible factors in which $a(x)$ is decomposed is 3.

Theorem 6 is applied to vector $(0, 0, 1, 0) = x^2$.

So ideal I is

$$I = \langle x^4 + 2, x^2 - z \rangle.$$

The reduced Gröbner basis of I respect to the lexicographic monomial order, with $x > z$, is

$$G = \{2 + z^2, x^2 - z\}.$$

So

$$P_1(z) = 2 + z^2 = R_1(z)$$

$$P_2(x, z) = x^2 - z$$

$$R_2(z) = 1$$

$$S_2(x, z) = x^2 - z$$

Following Theorem 6

$$a(x) = \prod_{k=1, \dots, 2-1} \left(\prod_{c/Q_k(c)=0} S_{k+1}(x, c) \right) = \prod_{c/c^2+2=0} S_2(x, c) =$$

$$= S_2(x, 1) \cdot S_2(x, 2) = (x^2 - 1) \cdot (x^2 - 2)$$

As it discomposes just in two factors (and it must be three), we have to go on applying Theorem 6. To do it, we take the next element of the base and one of the factors.

Theorem 6 is applied to vector $(0, 1, 0, 1) = x + x^3$ and to the first factor.

Ideal I is now

$$I = \langle x^2 - 1, x^3 + x - z \rangle$$

The reduced Gröbner basis of I respect to the lexicographic monomial order, with $x > z$, is

$$G = \{z^2 + 2, x + z\}$$

And so

$$P_1(x, z) = z^2 + 2 = (z - 1) \cdot (z - 2) = R_1(z)$$

$$P_2(x, z) = x + z$$

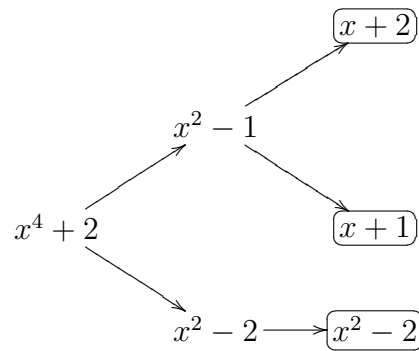
$$R_2(z) = 1$$

$$S_2(x, z) = x + z$$

Then

$$x^2 - 1 = S_2(x, 2) \cdot S_2(x, 1) = (x + 2) \cdot (x + 1)$$

We can summarize all the process in the following diagram:



4 Maple program

Once a theoretical result with a clearly practical implication has been proven, it results nice to apply it to concrete cases.

The version with Gröbner Bases of the Berlekamp's algorithm, which is deduced from the theorem exposed in this article has been developed in five functions in Maple 9.5. The Maple code is given below and it is also available in <http://web.usc.es/~mladra>. To run this code right, Andreas Pirklbauer's package GB must be loaded before. This package is in the Share Library and performs the Gröbner basis computation over a finite field.

A pair of examples of utilization of this functions package are shown.

Example 8 *To calculate the factorization of $x^4 + 2 \in Z_3[x]$, it should be written:*

```
> GBerlekamp(x^4+2,x,3,alpha,alpha);
{2 + x, x^2 + 1, 1 + x}
```

Example 9 *To calculate the factorization of $(1 + \alpha + \alpha^3) + \alpha x + (1 + \alpha^2 + \alpha^3)x^3 + (1 + \alpha + \alpha^3)x^4 + x^5 \in \frac{Z_2[\alpha,x]}{\langle \alpha^4 + \alpha + 1 \rangle} = F_{16}$, it should be written:*

```
> GBerlekamp(1+alpha+alpha^3+alpha*x+(1+alpha^2+alpha^3)*x^3+
(1+alpha+alpha^3)*x^4+x^5,x,2,alpha^4+alpha+1,alpha);
{alpha^3 + 1 + (alpha^3 + alpha)x + x^3 + x^2(alpha^2 + alpha + 1),
alpha^2 + 1 + (alpha^3 + alpha^2)x + x^2}
```

Maple code

```
GBerlekamp:=proc(apoly,var,p::prime,irredpoly,alpha)

# This function returns the factorization of a monic squarefree polynomial

local GB_apolydeg,GB_card::integer,GB_W,proceslist,aux,icont,hcont,dimW::integer,
bInContinue,cardproceslist::integer,domloop, apolycan,irredpolycan;

apolycan:=CanonicalForm(apoly,irredpoly,alpha,p);

irredpolycan:=Normal(irredpoly) mod p;

proceslist:={apolycan};
```

```

blnContinue:=true;

GB_apolydeg:=degree(apoly,var);

GB_card:=p^degree(irredpoly,alpha);

# We calculate a basis for W.

GB_W:= WBasis(apolycan,var,GB_apolydeg,p,GB_card,irredpolycan,alpha);

GB_W:=evalm([seq(var^icont,icont=0..GB_apolydeg-1)] &* GB_W);

GB_W:=convert(GB_W,set);

dimW:=nops(GB_W);

if dimW > 1 then

    for hcont in GB_W[2..-1] while blnContinue

        do

            domloop:=proceslist;

            for icont in domloop while blnContinue

                do

                    aux:=GroebnerKernel(icont,hcont,var,p,irredpolycan,alpha);

                    if nops(aux) > 1 then

                        proceslist:=proceslist union {op(aux)};

                        proceslist:=proceslist minus {icont};

                    fi;

                    if dimW=nops(proceslist) then

                        blnContinue:=false;

                    fi;

                end do;

            end do;

        end do;

    fi;

```

```

    return(proceslist);
end proc;

BerlekampPetrMatrix:=proc(apoly,var,apolydeg,p,cardfinitefield,irredpoly,alpha)
# This function returns Berlekamp-Petr Matrix

    local mat::list,apolydegm1::integer,aux,icont::integer,jcont::integer;

    mat:=array(1..apolydeg,1..apolydeg);

    apolydegm1:=apolydeg-1;

    for icont from 0 to apolydegm1
    do
        aux:=Rem(var^(icont*cardfinitefield),apoly,var) mod p;
        aux:=CanonicalForm(aux,irredpoly,alpha,p);
        for jcont from 0 to apolydegm1
        do
            mat[icont+1,jcont+1]:=coeff(aux,var,jcont);
        end do;
    end do;

    return evalm(mat);
end proc;

WBasis:=proc(apoly,var,apolydeg,p,cardfinitefield,irredpoly,alpha)
# This procedure obtains a basis for W #

    local sysmat,tagvars,syspoly,indepvars,aux::integer,hbasis,icont::integer,
        jcont::integer,scont::integer,tcont::integer,equats,remvars,dimKer::integer;

    tagvars:=[seq(tagvar[icont],icont=1..apolydeg)];

    indepvars:=[];

    # System Matrix to Solve #

```

```

sysmat:=evalm(BerlekampPetrMatrix(apoly,var,apolydeg,p,cardfinitefield,
                                irredpoly,alpha)-array(identity,1..apolydeg,1..apolydeg));
syspoly:=evalm(tagvars&*sysmat);
syspoly:=GB(convert(syspoly,set) union {irredpoly},
            [op(tagvars),alpha],plex) mod p;
syspoly:=convert(syspoly,set) minus {irredpoly};
# Calculus of independent variables #
for icont in tagvars
do
    aux:=nops(select(has,syspoly,icont));
    if aux<>1 then
        # Independent variable found #
        indepvars:=[op(indepvars),icont];
    fi;
end do;
dimKer:=apolydeg-nops(syspoly);
if nops(indepvars) < dimKer then
    remvars:=convert(tagvars,set) minus convert(indepvars,set);
    for icont in remvars while nops(indepvars) < dimKer
    do
        aux:=select(has,syspoly,icont);
        if nops(indets(aux[1]) minus {alpha})-\
            nops(indets(aux[1]) intersect convert(indepvars,set)) >1
        then
            # Independent variable found #

```

```

        indepvars:=[op(indepvars),icont];

    fi;

end do;

fi;

hbasis:=array(1..apolydeg,1..nops(indepvars));

scont:=0;

for icont in indepvars
do
    scont:=scont+1;

    equats:=map(x->if x=icont then x=1 else x=0 fi,indepvars);

    aux:=subs(equats,sypoly);

    tcont:=0;

    for jcont in tagvars
    do

        tcont:=tcont+1;

        if member(jcont,indepvars) then

            if jcont=icont then

                hbasis[tcont,scont]:=1;

            else

                hbasis[tcont,scont]:=0;

            fi;

        else

            hbasis[tcont,scont]:=CanonicalForm(op(2,solve(

                {select( has,aux,jcont)[1]=0},jcont)[1]),irredpoly, alpha, p);

        fi;
    end for;

```

```

        end do;

    end do;

    return(hbasis);

end proc;

GroebnerKernel:=proc(apoly,hbasispoly,var,p,irredpoly,alpha)

# This procedure calculates the factorization of a monic squarefree
# polynomial.

    local factorslist,GB_varz,Rk,Rkp1,GB_G,indset,icont::integer,Qroots,Pkp1,aux;

    factorslist:=[];

    GB_G:=GB({apoly,hbasispoly-GB_varz,irredpoly},
               [var,GB_varz,alpha],plex) mod p;

    GB_G:=convert(GB_G,set);

    GB_G:=GB_G minus {irredpoly};

    indset:=map(t->degree(t,var),GB_G);

# Resultant of apoly and hbasispoly-GB_varz #

    Rk:=select(t->if degree(t,var)=0 then true else false fi,GB_G)[1];

# We delete zero in indset #

    indset:=indset[2..-1];

# GB does not return ordered polynomials by their degree in var #

    for icont in indset

    do

        Pkp1:=select(t->if degree(t,var)=icont then true else false fi,GB_G)[1];

        Rkp1:=coeff(Pkp1,var,icont);

        Pkp1:=Quo(Pkp1,Rkp1,var) mod p;

        aux:=Quo(Rk,Rkp1,GB_varz) mod p;

```

```

    aux:=subs(alpha=RootOf(irredpoly),aux);

    if degree(irredpoly,alpha)=1 then

        Qroots:=Roots(aux) mod p;

    else

        Qroots:=Roots(aux,RootOf(irredpoly)) mod p;

        Qroots:=subs(RootOf(irredpoly)=alpha,Qroots);

    fi;

    factorslist:=[op(map(t->CanonicalForm(subs(GB_varz=t[1],
    Pkp1),irredpoly,alpha,p),Qroots)), op(factorslist)];

    Rk:=Rkp1;

end do;

return(factorslist);

end proc;

CanonicalForm:=proc(expr,irredpoly,alpha,p)

# This procedure calculates the canonical form of expr in a finite field #

    local aux;

    if degree(irredpoly,alpha)>1 then

        aux:=subs(alpha=RootOf(irredpoly),expr);

        aux:=Normal(aux) mod p;

        aux:=subs(RootOf(irredpoly)=alpha,aux);

    else

        aux:=Normal(expr) mod p;

    end if;

    return(aux);

end proc;

```

References

- [1] E. Berlekamp, Factoring polynomials over finite fields. *Bell System Tech. J.* 46:1853–1859 (1967).
- [2] E. Berlekamp, Factoring polynomials over large finite fields. *Math. Comp.* 24:713–735 (1970).
- [3] D. Cantor, H. Zassenhaus, A new algorithm for factoring polynomials over finite fields. *Math. Comp.* 36:587–592 (1981).
- [4] D. Cox, J. Little, D. O’Shea, *Ideals, Varieties and Algorithms*. Springer-Verlag, New York, 1992, p. 513.
- [5] G. Czichowski, A Note on Gröbner Bases and Integration of Rational Functions. *J. Symb. Comput.* 20:163–167 (1995).
- [6] K. Geddes, S. Czapor, G. Labahn, *Algorithms for Computer Algebra*. Kluwer Academic Publishers, Boston, MA, 1992, p. 585.
- [7] E. Kaltofen, V. Shoup, Subquadratic-time factoring of polynomials over finite fields. *Math. Comp.* 67:1179–1197 (1998).
- [8] M. Kreuzer, L. Robbiano, *Computational Commutative Algebra 1*. Springer-Verlag, Berlin, 2000, p. 321.
- [9] D. Lazard, Ideal Bases and Primary Decomposition: Case of Two Variables *J. Symb. Comput.* 1:261–270 (1985).
- [10] H. Niederreiter, Factoring polynomials over finite fields using differential equations and Normal Bases. *Math. Comp.* 62:819–830 (1994).
- [11] M. Noro, K. Yokoyama, Yet another practical implementation of polynomial factorization over finite fields. Proc. ISAAC’2002, ACM Press, New York, pp. 200–206 (2002).
- [12] J. von zur Gathen, V. Shoup, Computing Frobenius maps and factoring polynomials. *Comput. Complexity* 2:187–224 (1992).
- [13] J. von zur Gathen, J. Gerhard, *Modern Computer Algebra*. Cambridge University Press, Cambridge, UK, 1999, p. 753.