POLYNOMIAL-TIME FACTORIZATION OF MULTIVARIATE POLYNOMIALS OVER FINITE FIELDS

J. von zur Gathen and E. Kaltofen†

Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4, Canada

Abstract.

We present a probabilistic algorithm that finds the irreducible factors of a bivariate polynomial with coefficients from a finite field in time polynomial in the input size, i.e. in the degree of the polynomial and *log*(cardinality of field). The algorithm generalizes to multivariate polynomials and has polynomial running time for densely encoded inputs. Also a deterministic version of the algorithm is discussed whose running time is polynomial in the degree of the input polynomial and the size of the field.

1. Introduction and Summary of Results

Polynomials with coefficients from a finite field and their factorization properties have been considered for a long time. In 1846, Schönemann proved that univariate polynomials over $\mathbf{Z}_p$ have the unique factorization property (Schönemann [21], p.276). Since there is only a finite number of factor candidates, the factorization problem is immediately shown to be computable. However, an efficient algorithm to compute these factors was not presented until the late 1960's. Berlekamp [3] then devised an algorithm which factors univariate polynomials over a finite field F with q elements in $O(qn^3)$ field operations, where n is the degree of the polynomial (see Knuth [15], Sec.4.6.2). This running time is polynomial both in n and q. Soon after, Berlekamp [4] made the running time polynomial in the input size, i.e. using $\log q$ rather than q, at the expense of introducing a probabilistic rather than deterministic method. It seems natural to ask whether this can also be accomplished for multivariate, say bivariate polynomials, over F. In particular, given a bivariate polynomial of total degree n with coefficients in F, can one find (probabilistically) its factors in sequential running time polynomial in n and $\log q$?

† E. Kaltofen's current address: Rensselaer Polytechnic Institute, Department of Computer Science, Troy, New York 12181. Appears in *Math. Comput.*, **45**, pp. 251-261 (1985).

Older algorithms proposed for this problem (e.g. Musser [19], 2.7.2, and Davenport-Trager [8]) had an exponential worst case running time. The same was true of the Berlekamp-Zassenhaus approach to factoring integer polynomials, until Lenstra-Lenstra-Lovász [18] (for the univariate case) and Kaltofen [13, 14] (for the multivariate case) provided a polynomial-time solution. In this paper, we give a polynomial-time factorization algorithm for bivariate polynomials over a finite field, based on the methods from Kaltofen [13]. Chistov-Grigoryev [7] and Lenstra [17] have also presented polynomial-time algorithms for this problem. Both these papers are based on the short vector algorithm for lattices from Lenstra-Lenstra-Lovász [18], and are quite different from ours.

Our algorithm has two variants: a probabilistic one (Las Vegas) with running time $(nlogq)^{O(1)}$, and a deterministic one with running time $(nq)^{O(1)}$, where n is the degree of the input polynomials and q the cardinality of the coefficient field (section 4.2). In our deterministic version, q could be replaced by $logq$ if one could factor univariate polynomials over finite fields in deterministic time polynomial in $logq$. Observe that $nlogq$ is the input size in a natural "dense" encoding of polynomials. Our description concentrates on the probabilistic variant, which may be the more important one for practical purposes.

We also give a parallel variant (section 4.1) for our algorithm which runs in parallel time $O(log^2 n \; logq)$, based on the results for univariate factorization in von zur Gathen [10].

It is straightforward to generalize our algorithm for factoring multivariate polynomials (section 4.3). Again the running time is polynomial in the input size, provided the inputs are encoded as dense polynomials. Chistov-Grigoryev [7] and Lenstra [17] also present multivariate factoring algorithms of polynomial running time. Using an effective Hilbert Irreducibility Theorem and the results presented here, von zur Gathen [11] presents a polynomial-time factoring procedure for sparsely encoded multivariate polynomials.

2. Factoring a Nice Polynomial

The algorithm for factoring an arbitrary polynomial $f \in F[x,y]$ proceeds in two stages. We first preprocess f into a "nice format", and then factor the nice polynomial. We start by describing the crucial second stage.

We assume that an algorithm for factoring univariate polynomials over F is given. This algorithm will be allowed to be probabilistic (Las Vegas), so that it either returns the correct answer or "failure", the latter with small probability.

Definition 2.1. Let F be a field, and $f \in F[x,y]$. We call f nice if the following conditions hold:

(N_1) $f(x,0) \in F[x]$ is squarefree.

(N_2) f is monic with respect to x.

Algorithm QUICK FACTORING.

Input: A nice polynomial $f \in F[x,y]$.

Output: An irreducible factor $g \in F[x,y]$ of f.

1. Compute an irreducible monic factor $h \in F[x]$ of $f(x,0)$. If $h = f(x,0)$, then return f. If the probabilistic univariate procedure returns failure, then return "failure".

2. Set $d_x = deg_x f$, $d_y = deg_y f$, and $d = 2d_x d_y$. Set $E = F[t]/(h(t))$, and $a_0 = (t \bmod h(t)) \in E$. We use the Newton iteration in steps 3 and 4 to compute $b \in E[y]$ such that

$$f(b,y) \equiv 0 \bmod y^{d+1}$$

in $E[y]$.

3. Set $t = \dfrac{1}{f_x(a_0,0)} \in E$, where $f_x = \dfrac{\partial f}{\partial x} \in F[x,y]$. (Note that $f_x(a_0,0) \neq 0$, since otherwise a_0 would be a double zero for $f(x,0)$, contradicting its squarefreeness.)

4. For $k = 1,...,d$ compute

$$a_k = a_{k-1} - tf(a_{k-1},y) \in E[y].$$

("mod y^{k+1}", i.e. truncating the powers y^l of y with $l>k$. Then $f(a_k) \equiv 0 \bmod y^{k+1}$.) Set $b=a_d$.

5. Find the minimal i, $degh \leq i \leq d_x$, for which there exist $u_0,...,u_{i-1} \in F[y]$ such that

$$deg_y u_j \leq d_y \quad \text{for } 0 \leq j < i,$$

$$b^i + \sum_{0 \leq j < i} u_j b^j \equiv 0 \bmod y^{d+1.}$$

Compute the corresponding $u_0,...,u_{i-1}$.

6. Return

$$g = x^i + \sum_{0 \leq j < i} u_j x^j \in F[x,y].$$

For the timing analysis, we assume that the factorization procedure used in step 1 to factor a univariate polynomial of degree e takes at most $\tau(e)$ operations in F. We will later allow a probabilistic procedure (Las Vegas), which either correctly returns an irreducible factor, or "failure".

Theorem 2.2. Let $f \in F[x,y]$ be nice, and assume that step 1 of algorithm QUICK FACTORING does not return "failure". Then the following hold:

(i) The output is an irreducible factor of f.

(ii) Let n be the total degree of f, and d_x the degree of f with respect to x. The algorithm can be performed in $O(n^3 d_x^4) + \tau(d_x)$ or $O(n^7) + \tau(n)$ operations in F.

Proof. The correctness claim (i) follows just as in Kaltofen [13], section 4. The output g will be the irreducible factor of f such that h divides $g(x,0)$.

In applying (ii), we will need the first estimate, which clearly implies the second one. First observe that step 3 can be performed in $O(d_x)$ operations in E. Each a_k in step 4 takes $O(d_x)$ operations in $E[y]$ (computing mod y^{k+1}). By Lemma 2.3, step 4 then takes $O(d^2 d_x \log^4 d)$ operations in E.

In step 5, we first compute $b^2, b^3, \ldots, b^{d_x}$ in $O(d_x)$ operations in $E[y]$ (again mod y^{d+1}) or $O(d_x d \log^4 d)$ operations in E. Then we have to solve a system of at most $(d+1)d_x$ linear equations in at most $d_x(d_y+1)$ unknowns over F. (Note that one equation in E corresponds to less than d_x equations in F.) Gaussian elimination solves the system in $O((d_x(d_y+1))^2 (d+1)d_x)$ or $O(d^3 d_x)$ operations in F. Noting that $[E:F] < d_x$, $d \le nd_x \le n^2$, and using Lemma 2.3, we get a total of $O(d^3 d_x + d^2 d_x^2 \log^4 d \log^4 d_x)$ or $O(n^3 d_x^4)$ operations in F. □

The following lemma gives an upper bound on the time to perform arithmetic in finite field extensions. Lempel-Seroussi-Winograd [16] give a quasi-linear bound on the number of nonscalar operations needed for multiplication.

Lemma 2.3. Let F be an arbitrary field, and $h \in F[x]$ of degree d. Then an arithmetic operation (+, -, *, division by an invertible element) in $F[x]/(h)$ can be performed in $O(d \log^4 d)$ operations in F. If the cardinality $\#F$ of F is at least $2d$, then it can be performed in $O(d \log^2 d)$ operations.

Proof. Let $q = \#F$. We consider the elements of $F[x]/(h)$ as being represented by polynomials in $F[x]$ of degree less than d (i.e. by its sequence of d coefficients). The last claim is well-known (see Aho-Hopcroft-Ullman [1], 8.3). If $q < 2d$, we can (deterministically) compute an irreducible polynomial $w \in F[t]$ of degree $\lceil \log_q 2d \rceil$ (see Theorem 3.1). Setting $K = F[t]/(w)$, an operation in $F[x]/(h) \subseteq K[x]/(h)$ can be simulated in $O(d \log^2 d)$ operations in K, and an operation in K costs $O(\log^2 d)$ operations in F, giving a total of $O(d \log^4 d)$ operations in F for each operation in $F[x]/(h)$. □

Remark 2.4. Some simplifications of the algorithm may be of practical interest. Step 4 only has to be executed for $k = 1, \ldots, \bar{d}$, where

$$\bar{d} = \left\lceil \frac{d_y(2d_x - 1)}{\deg h} \right\rceil .$$

(See Kaltofen [13], Theorem 4.1.) The algorithm can also be performed without the assumption that f is monic with respect to x. If $c \in F[y]$ is the leading coefficient, then step 4 has to be executed for $k = 1,...,\delta$, where

$$\delta = \left\lceil \frac{d_y(2d_x - 1) + deg\ c\,(d_x + 1)}{degh} \right\rceil .$$

In step 5, we then have to consider

$$deg_y u_j \le d_y\ deg\ c \quad \text{for } 0 \le j < i ,$$

$$cb^i + \sum_{0 \le j < i} u_j b^j \equiv 0 \bmod y^{\delta+1} ,$$

and in step 6, we have to compute

$$v = gcd(c, u_0, ..., u_{i-1}) \in F[y]$$

and return

$$g = \frac{c}{v} x^i + \sum_{0 \le j < i} \frac{u_j}{v} x^j .$$

3. The Preprocessing Stage

In this section we describe the algorithm for factoring an arbitrary bivariate polynomial over a finite field. It converts the input polynomial into a nice polynomial, calls QUICK FACTORING, and then determines a factor of the input polynomial.

We first need an algorithm for the *gcd* of two bivariate polynomials. We use a modular approach for this; see Brown [5].

Algorithm BIVARIATE GCD.

Input: Two polynomials $f, g \in F[x,y]$, where f is monic with respect to x, and F is an arbitrary field.

Output: The monic (with respect to x) *gcd* $h \in F[x,y]$ of f and g.

1. Set $d_x = max(deg_x f, deg_x g)$, $d_y = max(deg_y f, deg_y g)$, and $d = 2d_x d_y$. If $d = 0$, use a procedure for univariate *gcd*'s. If $q = \#F < 3d$, then do the following. Choose an irreducible monic polynomial $w \in F[t]$ of degree $\lceil log_q 3d \rceil$, and replace F by the extension field $F[t]/(w)$.

2. Choose any pairwise distinct $a_1, ..., a_{2d} \in F$ such that $g(x, a_i)$ has the same degree in x as g. (We need at most $2d + d_y < 3d$ elements in F to locate such evaluation points.)

3. For all i, $1 \le i \le 2d$, compute the monic

$$h_i = gcd(f(x,a_i), g(x,a_i)) = \sum_{0 \le j} h_{ij} x^j \in F[x].$$

4. Set $m = min\{deg h_i : 1 \le i \le 2d\}$, and choose some $M \subseteq \{1,...,2d\}$ with $\#M = d_y + 1$ and $deg\ h_i = m$ for all $i \in M$.

5. For $0 \le j \le m$, interpolate the h_{ij}'s: compute $b_j \in F[y]$ of degree at most d_y with $b_j(a_i) = h_{ij}$ for all $i \in M$. (In particular, $b_m = 1$.)

6. Return $h = \sum_{0 \le j \le m} b_j x^j$.

Theorem 3.1. Let $f, g \in F[x,y]$, where f is monic with respect to x, and let d be as in step 1 above. Then algorithm BIVARIATE GCD has the following properties:

(i) It correctly computes a gcd of f and g,

(ii) It can be performed in $O(d^2 log^4 d)$ operations in F. If $\#F \ge 3d$, then it takes $O(d^2 log^2 d)$ operations.

Proof. Let $h_0 = gcd(f,g) \in F[x,y]$ be monic with respect to x, and $f = uh_0$, $g = vh_0$ with $u, v \in F[x,y]$. Then the resultant

$$r = res_x(u,v) \in F[y]$$

is a polynomial of degree less than d, and for any $a \in F$ with $r(a) \ne 0$ and $deg\ g(x,a) = deg_x g$ we have

$$gcd(f(x,a), g(x,a)) = h_0(x,a).$$

Thus for at least d among $h_1, ..., h_{2d}$ we have

$$h_i = h_0(x, a_i),$$

and $deg\ h_i \ge deg_x h_0$ for all i. Therefore some M as in step 4 can be found, and steps 5 and 6 correctly compute $h = h_0$.

If $q < 3d$, then we can find w as in step 1 deterministically by testing each monic polynomial $w \in F[t]$ of degree $l = \lceil log_q 3d \rceil$ for irreducibility. There are at most $q^l \le 3dq < 9d^2$ such polynomials, and each irreducibility test takes $O(log^2 d\ log^2 logd\ logloglogd\ logq)$ or $O(log^4 d)$ operations in F (Rabin [20]). Any operation in $F[t]/(w)$ can be simulated by $O(log^2 d)$ operations in F. This factor $log^2 d$ has to be multiplied to the estimates for steps 3 to 6 only if $q < 3d$.

In step 3, the number of operations is $O(d)$ for each $f(x,a_i)$ and $g(x,a_i)$, and $O(d_x log^2 d_x)$ for each h_i (Aho-Hopcroft-Ullman [1], 8.9), for a total of $O(d(d + d_x log^2 d_x))$ operations. Obviously $m \le d_x$, and the interpolations in step 5 take $O(d_x(d_y log^2 d_y))$ operations (Aho-Hopcroft-Ullman [1], 8.7). The total is $O(d^2 log^2 d\ log^2 d)$ operations, and $O(d^2 log^2 d)$ if $q \ge 3d$. □

We now describe the algorithm for computing a factor of a bivariate polynomial over a finite field.

Algorithm BIVARIATE FACTORING

Input: A polynomial $f \in F[x,y]$, where F is a finite field with q elements, and $p = charF$, the characteristic of F.

Output: Either a non-constant factor $g \in F[x,y]$ of f, or "failure".

1. (Check primitivity) Set $d_x = deg_x f$, and write $f = \sum_{0 \le i \le d_x} f_i x^i$ with $f_i \in F[y]$. Compute the content

$$c = cont_x(f) = gcd(f_0,\ldots,f_{d_x}) \in F[y].$$

If c is non-constant, then return c.

2. (Check squarefreeness) Compute $f_x = \frac{\partial f}{\partial x}$ and $f_y = \frac{\partial f}{\partial y}$. If $f_x = f_y = 0$, then write $f = \sum_{0 \le i,j} f_{ij} x^{ip} y^{jp}$, set $g = \sum_{0 \le i,j} f_{ij}^{q/p} x^i y^j$ and return g. (We have $g^p = f$.) If $f_x = 0$ and $f_y \ne 0$, then interchange the role of x and y and goto step 1. Now we have $f_x \ne 0$. Compute $g = gcd(f, f_x)$. If $g \ne 1$, then return g.

3. (Monic version of f) Let $f_0 \in F[y]$ be the leading coefficient of f with respect to x. Set

$$v = f_0^{d_x - 1} f\left(\frac{x}{f_0}, y\right) \in F[x,y].$$

(v is monic of degree d_x with respect to x.)

4. (Extend F) Set $d_y = deg_y v$, $m = max\{d_x, d_y\}$, and $d = 2 d_x d_y$. If $q = \#F > d$, then set $F^* = F$. Otherwise choose a prime number l with $m < l \le 2m$. Choose monic polynomials $w_1,\ldots,w_{8ln} \in F[t]$ of degree l at random, and test them for irreducibility. If none is irreducible, return "failure". Otherwise choose an irreducible w_i, and set $F^* = F[t]/(w_i)$.

5. (Good evaluation point) Set

$$r = disc_x(v) = res_x\left(v, \frac{\partial v}{\partial x}\right) \in F[y].$$

(r is a nonzero polynomial of degree $\le (2d_y - 1)d_x < d$.) Choose $c \in F^*$ such that $r(c) \ne 0$, and set

$$f^* = v(x, y-c) \in F^*[x,y].$$

(f^* is nice.)

6. Call procedure QUICK FACTORING with input $f^* \in F^*[x,y]$, to return $g^* \in F^*[x,y]$.

7. Set

$$e = deg_x g^*, \quad g_1 = f_0^{-e+1} \, g^*(xf_0, y+c) \in F^*[x,y],$$
$$g_0 = cont_x(g_1) \in F^*[y], \quad g = g_1/g_0 \in F[x,y],$$

and return g.

For a concrete estimate of the running time, we have to implement step 1 of the procedure QUICK FACTORING. The probabilistic version of Berlekamp's univariate algorithm due to Cantor-Zassenhaus [6] (see also Knuth [15], 4.6.2) factors a polynomial of degree e in

$$O(e^3 + e^2 \, loge \, logq)$$

operations in F, where $q = \#F$. Other algorithms for this problem are due to Berlekamp [4], Rabin [20], Ben-Or [2]. This algorithm can be written as a Las Vegas procedure, so that it either returns an irreducible factor or "failure" - the latter with probability at most ½. The algorithm requires $O(e \, loge)$ random choices from F, and we assume that they can be performed in $O(e \, loge \, logq)$ random bit choices. The cost of the Las Vegas univariate factoring procedure in step 1 of QUICK FACTORING is dominated by the cost of other steps. So we can apply that procedure several times, say $n+1$ times, to obtain failure probability at most 2^{-n-1}, where n is the total degree of f.

Theorem 3.2. Let F be a finite field with q elements, and $f \in F[x,y]$ of total degree n. Algorithm BIVARIATE FACTORING with input f has the following properties.

(i) If f is irreducible, it either returns f or "failure".

(ii) If f is reducible, it either returns a proper factor of f or "failure".

(iii) Failure occurs with probability at most 2^{-n}.

(iv) The algorithm can be performed with

$$O(n^7 log^4 n \; log^2 q (n^5 + logn \; logq))$$

bit operations, and $O(n^5 \, logq)$ random bit choices.

Proof. It is well-known how the factorization of f and v in $F[x,y]$ are related; see Knuth [15], exercise 4.6.2-18, using the coefficient domain $F[y]$.

By a result of von zur Gathen [11], section 5, the factorizations of v in $F[x,y]$ and $F^*[x,y]$ are the same. The relation between factors of v and f^* in $F^*[x,y]$ is obvious. Also, f^* is nice.

Note that an l as in step 4 exists by Bertrand's Postulate (see Hardy-Wright [12]). In order to see that some c as in step 5 can be found, it is sufficient to show that $\#F^* = q^l \geq d$:

$$q^l \geq 2^{m+1} \geq 2m^2 \geq d.$$

(The second inequality holds for all $m \geq 1$ with $m \neq 3$. But for $m = 3$ we have $l \geq 5$ and $q^l \geq d$.) We have now proven (i) and (ii) in the case where no failure occurs.

Failure can either occur in step 6 - with probability at most 2^{-n-1} by the remark before theorem 3.2 - or in the computation of w_i in step 4. There are exactly $(q^l - q)/l$ many irreducible monic polynomials of degree l in $F[x]$ (Schönemann [21], section 46, p. 317), so that step 4 has failure probability

$$\leq (1 - \frac{q^l - q}{lq^l})^{8ln} \leq (1 - \frac{1}{2l})^{8ln} \leq e^{-2n} \leq 2^{-n-1}.$$

Therefore the total failure probability is at most 2^{-n}.

For the timing estimate, first note that $d_x \leq n$, $d_y \leq n^2$, $d = 2d_x d_y \leq 2n^3$, $l \leq 2m \leq 2n^2$, and the total degree n^* of f^* is not more than n^2. Step 1 takes $O(n^3)$ operations, and step 3 $O(n^4)$ operations. In step 2, the gcd can be computed in $O(d^2 log^4 d)$ operations in F by Theorem 3.1, and the p-th root in $O(d\ log\ \frac{q}{p})$ operations in F, since $deg_y f \leq d_y$ with d_y from step 4. The prime number l can be found deterministically in $O(m^{3/2} log^2 m)$ bit operations, and w in $O(8ln\ m^2\ log^2 m\ loglogm\ logq)$ or $O(n^7 log^3 n\ logq)$ operations in F (Rabin [20]). Steps 5 and 7 both take $O(d_x d^2)$ operations. The cost of the algorithm is dominated by the running time for step 6, which is

$$O((n^*)^3 d_x^4 + n(d_x^3 + d_x^2\ logd_x\ log(q^l))$$

or $O(n^{10} + n^5\ logn\ logq)$ operations in F^*. Each operation in F^* can be simulated by $O(l\ log^4 l)$ operations in F by Lemma 2.3, and $O(l\ log^4 l\ log^2 q)$ bit operations. Thus the total cost is

$$O(n^7\ log^4 n\ log^2 q\ (n^5 + logn\ logq))$$

bit operations.

The number of random bit choices is $O(8ln\ l\ logq)$ or $O(n^5 logq)$ in step 4, and $O(nd_x\ logd_x\ logq)$ or $O(n^2\ logn\ logq)$ in step 5. □

We note that if $q > d$, then the algorithm uses $F^* = F$ and runs in $O(n^3\ log^2 q(n^7 + logn\ logq))$ bit operations.

Once we have found one nontrivial factor using BIVARIATE FACTORING, we can of course apply the algorithm to this partial factorization. Repeating this yields a probabilistic algorithm which returns either the complete factorization of the input polynomial, or "failure". The total number of bit operations is

$$O(n^8 log^4 n \;\; log^2 q (n^5 + logn \;\; logq)),$$

and the number of random bit choices is $O(n^6 logq)$. The failure probability can be made as small as $n 2^{-2n} \le 2^{-n}$ by repeating the algorithm twice at each stage, in fact less than any prescribed $\varepsilon > 0$ by repeating the whole procedure at least $(log\,\varepsilon)/n$ times. So we have

Corollary 3.3. Let F be a finite field with q elements. Polynomials in $F[x,y]$ of total degree n can probabilistically (Las Vegas) be factored completely in time polynomial in n and $logq$.

4. Some Variants

4.1. A parallel version. The basic subroutines for algorithm BIVARIATE FACTORING are a univariate factoring procedure over finite fields, computing univariate gcd's, and solving systems of linear equations over a finite field (which also solves the interpolation step in BIVARIATE GCD). In von zur Gathen [10], all these tasks have been shown to be probabilistically solvable in parallel with $O(log^2 n)$ operations in F (respectively $O(log^2 n \;\; log^2 k \;\; logp)$ for factoring). Here n is the total degree of the input polynomial, $p = charF$, and $q = p^k = \#F$. For a complete factorization, one would lift all irreducible factors of $f(x,0)$ from step 1 of QUICK FACTORING in parallel, using a quadratic Newton procedure (see e.g. von zur Gathen [9]), and then discard duplicate ones. As our model of parallel computation we can take Boolean circuits. Also a prime number l as in step 4 of BIVARIATE FACTORING can be found in parallel with $O(log^2 n)$ bit operations.

The resulting Las Vegas algorithm returns either the complete factorization of the input polynomial, or "failure"; the latter with probability no more than 2^{-n}. The number of processors required is polynomial in n and $logq$. Thus we have

Theorem 4.1. Let F be a finite field with $q = p^k$ elements, where $p = charF$. Polynomials in $F[x,y]$ of total degree n can probabilistically be factored completely in parallel time $O(log^2 n \;\; log^2(kn) \; logp + logn \; logq)$.

The second summand comes from the computation of p-th roots in step 2 of BIVARIATE FACTORING, and the first summand from step 1 of QUICK FACTORING, where a univariate polynomial of degree at most n over a field with not more than p^{kn^2} elements has to be factored. In step 4 of QUICK FACTORING, each step of the quadratic Newton iteration has to compute $t \in E[y]$ such that $t\, f_k(a_k,y) \equiv 1 \bmod y^{2k}$. This congruence can be considered as a system of linear equations over the ground field, and solved in parallel time $O(log^2 n)$.

4.2. A deterministic version. Algorithm BIVARIATE FACTORING can be viewed as a reduction from bivariate factoring to univariate factoring over finite fields. All steps of this reduction are deterministic, except the choice of $w_i \in F[t]$ in step 4. We need w_i in order to construct F^* with $\#F^* \ge d$, so that step 5 can be executed. But it is sufficient to have $w^+ \in F[t]$ with $l = degw^+ \ge log_q d$, and use $F^+ = F[t]/(w^+)$. Such an w^+ can be found deterministically in time polynomial in d. The problem is that we are not guaranteed that an irreducible factor of f is irreducible in $F^+[x,y]$. Our choice for the degree of w was motivated by the fact that then irreducible factors remain irreducible in $F^*[x,y]$ (von zur Gathen [11]), and we can avoid the costly norm computation below.

However, the case of w^+ as above can be salvaged by introducing the norm

$$N(g) = N_{F^+(x,y)/F(x,y)}(g) = (-1)^{il} res_t(w, \bar{g})$$

for $g \in F^+[x,y]$, where we choose $\bar{g} \in F[x,y,t]$ of degree $i<l$ in t such that $g \equiv \bar{g} \bmod w$ (see van der Waerden [22], p.89). It is well-known that if $g \in F^+[x,y]$ is an irreducible factor of f, then $N(g) \in F[x,y]$ is a power of an irreducible factor of f (Weyl [23], I.5). This irreducible factor is easily found as the gcd of f and $N(g)$. Thus we have

Theorem 4.2. Let F be a finite field with q elements.

(i) Factoring bivariate polynomials over F of total degree n is deterministically reducible to factoring univariate polynomials of degree at most n (over a small finite extension field of F). The number of operations for the reduction is polynomial in n and $\log q$.

(ii) Bivariate polynomials over F of degree n can be factored deterministically with a number of operations that is polynomial in n and q.

Proof. The above discussion has proven (i); we have to factor a univariate polynomial over a finite extension field F^+ of F. For (ii), we use any of the deterministic variants of Berlekamp's algorithm. □

4.3. A multivariate version. The algorithm can easily be modified for factoring multivariate polynomials over a finite field with q elements. One variable is selected as the main variable, and constants are substituted for the remaining variables. The resulting univariate polynomial is then factored and this factorization lifted. See Kaltofen [14] for details.

The running time of the resulting probabilistic algorithm is polynomial in the input size, and polynomial in the input size and q for the deterministic version. The input size for a polynomial $f \in F[x_1,...,x_k]$ of degree d is $O(d^k \log q)$ in a "dense encoding".

Another measure of size - of greater practical relevance - is the length of a "sparse encoding" of a multivariate polynomial, which is proportional to the number of nonzero terms in the polynomial. Multivariate polynomials can be factored in polynomial time also under this measure, taking input and output size into account (von zur Gathen [11]).

4.4 Remark. Let F be a finite field with q elements. We have (deterministically) reduced the factorization of a bivariate polynomial $f \in F[x,y]$ of total degree n to factoring univariate polynomials of degree at most n over a finite extension of F. This problem in turn can be reduced to factoring univariate polynomials over F itself, using the method in Berlekamp [4], section 6. All reductions are polynomial in n and $\log q$.

4.5. Remark. Our techniques do not allow to reduce the exponent 7 in the estimate for QUICK FACTORING in Theorem 2.2(ii). However, it would be easy to improve the running time of algorithm BIVARIATE FACTORING. In remark 2.4 we have indicated how to avoid the necessity of monic inputs. This would result in an $O(n^7 log^4 n \; log^2 q)$ probabilistic algorithm for factoring a bivariate polynomial of degree n over a finite field with q elements.

Open Question. Given a polynomial $f \in \mathbf{Z}_p[x,y]$, can one decide the irreducibility of f deterministically in time polynomial in $deg f$ and $log p$?

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The design and analysis of computer algorithms*. Addison-Wesley, Reading MA, 1974.

[2] M. Ben-Or, "Probabilistic algorithms in finite fields," Proc. 22nd Symp. Foundations Comp. Sci. IEEE, 1981, pp. 394-398.

[3] E.R. Berlekamp, "Factoring polynomials over finite fields," *Bell System Tech. J.* 46, 1967, pp. 1853-1859.

[4] E.R. Berlekamp, "Factoring polynomials over large finite fields," *Math. Comp.* 24, l970, pp. 713-735.

[5] W.S. Brown, "On Euclid's algorithm and the computation of polynomial Greatest Common Divisors," *J. ACM* 18, 1971, pp. 478-504.

[6] D.G. Cantor and H. Zassenhaus, "On algorithms for factoring polynomials over finite fields," *Math. Comp.* 36, 1981, pp. 587-592.

[7] A.L. Chistov and D.Yu. Grigoryev, "Polynomial-time factoring of the multivariable polynomials over a global field," LOMI preprint E-5-82, Leningrad, 1982.

[8] J.H. Davenport and B.M. Trager, "Factorization over finitely generated fields," Proc. 1981 ACM Symp. Symbolic and Algebraic Computation, ed. by P. Wang, 1981, pp. 200-205.

[9] J. von zur Gathen, "Hensel and Newton methods in valuation rings," *Math. Comp.* 42, 1984, pp. 637-661.

[10] J. von zur Gathen, "Parallel algorithms for algebraic problems," M J. Comp., to appear.

[11] J. von zur Gathen, "Factoring sparse multivariate polynomials," Proc. 24th Symp. Foundations Comp. Sci., IEEE, 1983, pp. 172-179.

[12] G.H. Hardy and E.M. Wright, *An introduction to the theory of numbers*, Clarendon Press, Oxford, 1962.

[13] E. Kaltofen, "A polynomial-time reduction from bivariate to univariate integral polynomial factorization," Proc. 23rd Symp. Foundations of Comp. Sci., IEEE, 1982, pp. 57-64.

[14] E. Kaltofen, "Polynomial-time reduction from multivariate to bivariate and univariate integer polynomial factorization," *SIAM J. Comp.* 15-2, 1985, to appear.

[15] D.E. Knuth, *The Art of Computer Programming*, Vol.2, 2nd Ed. Addison-Wesley, Reading MA, 1981.

[16] A. Lempel, G. Seroussi and S. Winograd, "On the complexity of multiplication in finite fields," *Theor. Comp. Science* 22, 1983, pp. 285-296.

[17] A.K. Lenstra, "Factoring multivariate polynomials over finite fields," Proc. 15th ACM Symp. Theory of Computing, 1983, 189-192.

[18] A.K. Lenstra, H.W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann. 261, 1982, pp. 515-534.*

[19] D.R. Musser, *Algorithms for Polynomial Factorization*, Ph.D. thesis and TR 134, Univ. of Wisconsin, 1971.

[20] M.O. Rabin, "Probabilistic algorithms in finite fields," *SIAM J. Comp.* 9, 1980, pp. 273-280.

[21] T. Schönemann, "Grundzüge einer allgemeinen Theorie der höheren Congruenzen, deren Modul eine reelle Primzahl ist," *J. f. d. reine u. angew. Math.* 31, 1846, pp. 269-325.

[22] B.L. van der Waerden, *Modern Algebra*, vol. 1, Ungar, New York, 1953.

[23] H. Weyl, *Algebraic Theory of Numbers*, Princeton University Press, 1940.