M. Phil. Thesis

A survey of polynomial factorisation algorithms

William M. Anderson

October 7, 1990

THESIS

submitted for the degree

of

Master of Philosophy

of

The University of Edinburgh

Acknowledgements

It is a pleasure to thank Dr. Kyriakos Kalorkoti for introducing me to the topic of this thesis, for his enthusiasm and for the patience he has shown while supervising my studies. I should also like to thank the University of Strathclyde for making it possible for me to study in the Computer Science Department of the University of Edinburgh.

Table of Contents

Chapter 1

Introduction

Historical Survey

The earliest work on polynomial factorisation was on univariate polynomials with integer coefficients. The classical method for determining factors of $f \in \mathbb{Z}[x]$ is associated with Kronecker, although it seems to have been known earlier [32].

If $f(x)$ factorises as $f(x) = g(x)h(x)$ then for any $a \in \mathbb{Z}$ the integer $g(a)$ will divide $f(a)$ and hence either $g(a) = 1$ or $g(a)$ is equal to a product of prime factors of $f(a)$. If the values of $g(x)$ were known at a suitable number of points $(a_1, g(a_1))$, $(a_2, g(a_2))$,... the factor $g(x)$ could be constructed by interpolation. A fuller description of the method, which is deterministic, can be found in [57]. Unfortunately the choice of the a_i and their number must be found by trial and error and the resulting polynomial tested by division into $f(x)$. It is easy to show that in the worst case the amount of effort required is exponential in the degree of f. Furthermore the method requires the factorisation of integers, for which no polynomial time algorithm is at present known.

From the viewpoint of complexity we should like to know if it is possible to find the factors of a polynomial in a time which is polynomial in the degree and in the length of the bit string required to describe f. If $f \in \mathbb{Z}[x]$ has degree n and

$$f = a_n x^n + \cdots + a_1 x + a_0,$$

we define the *size* of f, denoted $|f|$, by

$$|f|^2 = \sum_{r=0}^{n} a_r^2.$$

The complexity question can be formulated more precisely by asking whether f can be factorised in a time which is polynomial in n and $\log(|f|)$. We mention here that there are other ways of measuring a polynomial for complexity: some remarks are made in §6.5 about dense and sparse encoding.

There are circumstances when the coefficients of the polynomial are not in $\mathbb{Z}$ but in a finite field, an algebraic extension $\mathbb{Z}[\alpha]$ of $\mathbb{Z}$ or perhaps a ring of polynomials. Each of these distinct factorisation problems has received attention. Polynomials over a finite field arise in coding theory [5]; polynomials in $\mathbb{Z}[\alpha][x]$, or with factors there, arise in the formal integration of rational functions.

The simplest situation is the one in which the polynomial has its coefficients in a finite field, and Berlekamp's algorithm of 1967 [4] for this case was the first polynomial-time factorisation algorithm. Let the finite field be $\mathbb{Z}_q$, where q is a prime power, and let $f \in \mathbb{Z}_q[x]$ be of degree n. The number of non-zero polynomials in $\mathbb{Z}_q[x]$ of degree strictly less than n is q^n, but Berlekamp proved that we need only examine at most n^3q polynomials to determine a complete factorisation of f. A description of the algorithm, for the case when q is a prime, is given in §2.5.

Berlekamp's algorithm can be used as a basis for factoring polynomials f in $\mathbb{Z}[x]$. Essentially the idea is that if we find the factors of (f mod p_i) for a number of primes $p_1, p_2, \ldots$ then examination of these factors will suggest factors of f in $\mathbb{Z}[x]$ which can be tested by division. Although this method can be successful in practice there are irreducible polynomials in $\mathbb{Z}[x]$ for which (f mod p_i) has factors modulo infinitely many primes p_i. The method cannot be guaranteed to work because we do not know which primes to use.

Another idea which was exploited in 1969 by Zassenhaus [62] is Hensel lifting which is described in §2.6. If $g \in \mathbb{Z}[x]$ is such that (g mod p) divides (f mod p), then the technique of Hensel lifting may be used to modify g to g^*, say, which has the property that (g^* mod p^k) divides (f mod p^k) for some integer $k > 1$. In a p-adic sense g^* is a better approximation to a possible factor of f in $\mathbb{Z}[x]$ than g. Hensel lifting need not lead to a true factor of f in $\mathbb{Z}[x]$, as an example at the end of §2.6 shows, so that trial factors must be tested by division in $\mathbb{Z}[x]$. The

method is useful in practice, but the testing process may need exponential time in unfavourable cases.

Modifications of Zassenhaus' method to obtain factors of elements of an algebraic number field $\mathbb{Z}[\alpha][x]$ were made in 1975–78 by a number of authors including Wang [59], [60] and Weinberger and Rothschild [61].

A new kind of algorithm for factorisation in $\mathbb{Z}[\alpha][x]$, using a type of linear space called a *lattice*, was described by A. K. Lenstra in 1982 [36]. The lattice contains the factor which is being sought, and Lenstra shows that there is a way of finding it which is usually quick but may require exponential time in unfavourable cases.

Berlekamp's algorithm is a first stage in almost all other methods, and it is rather slow for large fields. Three probabilistic algorithms have been given for this situation, one by Berlekamp in 1970 [6], one by Rabin in 1980 [51] and one by Cantor and Zassenhaus in 1981 [11].

From 1982 a sequence of theoretical papers has appeared, of which the most important is by A. K. Lenstra, H. W. Lenstra and L. Lovász [40]. (In future we will refer to this paper as LLL.) In it the authors showed for the first time that a primitive element of $\mathbb{Z}[x]$ can be factored in a time polynomial in n and $\log |f|$. It is described in detail in Chapter 3, but we outline the ideas in the next paragraph.

Berlekamp's algorithm and Hensel lifting are used to determine $h^* \in \mathbb{Z}[x]$ with the property that $(h^* \bmod p^k)$ divides $(f \bmod p^k)$ in $\mathbb{Z}_{p^k}[x]$. LLL show that there must be a factor h of f in $\mathbb{Z}[x]$ (not necessarily distinct from f) with the property that $(h \bmod p^k)$ is divisible by $(h^* \bmod p^k)$. The set of possible h can be represented as a lattice. A special ordered basis for the lattice, called a reduced basis and which has certain properties, may be constructed. It turns out that by examining the reduced basis elements in order, one can determine an irreducible factor of f. If no proper factor is found then f is known to be irreducible. The major part of the effort in the algorithm is taken up with finding the reduced basis.

The algorithm in LLL may be modified to show that factorisation of polynomials over other domains may be accomplished in polynomial time. In a series of papers (1984–1987) A. K. Lenstra has shown how to prove this, by making an

appropriate choice of lattice, for multivariate polynomials over a finite field [38], the integers [37] and an algebraic number field [39].

A different way of extending the LLL algorithm to bivariate and multivariate polynomials over $\mathbb{Z}$ was given by Kaltofen in 1982 [25], [26], [27]. If $f(y,x)$ is an element of $\mathbb{Z}[y,x]$ which has been suitably preprocessed, the LLL algorithm will determine the factors of $f(0,x)$. Now

$$f(0,x) \equiv f(y,x) \pmod{y},$$

so Hensel lifting can be used to construct a sequence of factorisations of $f(y,x)$ modulo y^r for $r = 1,2,\ldots$. It is shown that, in contrast to the case for $\mathbb{Z}[x]$, this process terminates either with a condition showing that $f(y,x)$ is irreducible or with a factorisation. The time required is polynomial.

Although the LLL algorithm and those based on it are the best from the point of view of theory, they are not recommended for practical use [39]. The practical algorithms for polynomials over infinite fields all have the property that they may require an exponential search in unfavourable cases, but experience suggests that their typical performance is fast.

Contents of the Thesis

The thesis consists mainly of a survey of papers on the factorisation of polynomials. To set the scene the second chapter contains an account of a number of basic algebraic results and notations. It also has a description of Berlekamp's algorithm. The third chapter is devoted to the paper by Lenstra, Lenstra and Lovász. It contains a brief explanation of lattices and reduced bases for them as well as an explanation of the algorithm. Chapter 4 describes Kaltofen's result for factorisation of bivariate and multivariate integer polynomials. The extensions of the LLL result by A. K. Lenstra are in Chapter 5, while Chapter 6 has a collection of results that did not seem to fit conveniently elsewhere, including the Cantor-Zassenhaus method. The practical methods—apart from those for finite fields—are described in Chapter 7.

Chapter 8 is rather different from all the previous ones. It is a short description of factorisation in certain quotient rings. At the end there is a bibliography in which the references are preceded by a short guide.

Chapter 2

Standard Tools

2.1 Introduction

This chapter consists essentially of introductory material. The second section contains some remarks about rings and polynomials. The third section explains the computational model and says something about the polynomials we shall try to factorise and what the algorithms will be required to do. The remaining sections have an account of some standard algebraic tools, namely the Chinese Remainder Theorem, Berlekamp's algorithm, Hensel lifting, subresultants and finally a few remarks on ring and field extensions. Useful references for the algebraic tools are [15] and [32].

2.2 Rings and Polynomials

All rings will be commutative and unless otherwise indicated possess a multiplicative identity denoted by 1. Almost always in this thesis a ring will be a unique factorisation domain (UFD). As usual $\mathbb{Z}$ denotes the integers, $\mathbb{Z}_r$ the ring of residues modulo r and $\mathbb{Q}$ the rationals.

An element $u \in R$ is a *unit*, or invertible element, if there is an $a \in R$ such that $au = 1$.

Definition 2.2.1 *For elements f and g of R we say that g* divides *f if there is an element $h \in R$ such that $f = gh$.*

Definition 2.2.2 *If $f = gh \in R$, g is not a unit and neither g nor h is zero then we say that f* factorises *and that g is a* factor *of f. If neither g nor h is a unit we say that the factorisation is* proper.

Note that $f \neq 0$ is a factor of itself since $f = 1f$.

Definition 2.2.3 *If f and g are both non-zero but $fg = 0$ then we say that f and g are* zerodivisors *or* divisors of zero.

For example $0 = 2 \times 3$ in the ring $\mathbb{Z}_6$ so that 2 and 3 are zerodivisors of $\mathbb{Z}_6$.

If $f = uv$ and f is a unit then there is a $g \in R$ such that $fg = 1$ and hence $u(vg) = v(ug) = 1$, so that u and v are units. Thus no unit factorises.

Definition 2.2.4 *An element f of a ring is said to be* irreducible *or* prime *if it is not a unit and has no factor other than a unit multiple of f.*

Definition 2.2.5 *Suppose an element f of a unique factorisation domain R can be written as the product $f = g_1 \cdots g_r$ of irreducible elements g_i. If for any unit u, $g_i \neq ug_j$ whenever $i \neq j$, then f is said to be* squarefree.

As usual, a univariate *polynomial* f with indeterminate x over a ring R is either zero or has the form

$$f = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0, \tag{2.1}$$

where the coefficients a_i are elements of R, n is a natural number and $a_n \neq 0$. The ring of polynomials with coefficients in R and indeterminate x is referred to as $R[x]$. If $f \neq 0$ the number n is the *degree* of f, written $\deg(f)$: the degree of the zero polynomial is often taken to be $-\infty$. The coefficient a_n is called the *leading coefficient* of f, denoted $\mathrm{lc}(f)$, and f is said to be *monic* if $a_n = 1$. The *trailing coefficient*, or *constant term*, is a_0.

If f is a polynomial whose coefficients are in a UFD then these coefficients have a greatest common factor which is known as the *content* of f, denoted $\mathrm{cont}(f)$. A

polynomial with content 1 is said to be *primitive.* A monic polynomial in $R[x]$ is necessarily primitive. Every polynomial which has coefficients in a UFD can be thought of as the product of its content and a primitive polynomial, known as the *primitive part* of f. The primitive part of f is denoted $\mathrm{pp}(f)$. We note that if R is a UFD then so is $R[x]$. For future reference we quote Gauss's Lemma and a corollary.

Lemma 2.2.1 (Gauss) *Let R be a UFD and f, g elements of $R[x]$. Then*

$$\mathrm{cont}(fg) = \mathrm{cont}(f)\,\mathrm{cont}(g).$$

Corollary 2.2.1 *Let R be a UFD and $f \in R[x]$ be primitive. Then the factors of f in $R[x]$ are primitive.*

If a and b are non-zero elements of a field then conventionally we take $\gcd(a, b) = 1$. If $f \in k[x]$ the idea of the content of f is not significant. An element of $\mathbb{Z}[x]$ will have a content but an element of $\mathbb{Q}[x]$ will not.

An element $h \in R[x]$ which divides two other elements f and g is called a common factor of f and g. The common factor of highest degree is unique up to multiplication by an element of R and is called the greatest common divisor of f and g. It is denoted by $\gcd(f, g)$. If f and g are primitive then $\gcd(f, g)$ is primitive and uniquely defined up to multiples by units. In the case when $\deg(\gcd(f, g)) = 0$ we write $\gcd(f, g) = 1$ and say that f and g are *coprime.*

The following standard results will be used in Section 2.6.

Theorem 2.2.1 *Let k be a field and f, g primitive elements of $k[x]$. Then there are unique elements a and b of $k[x]$ such that $\deg(a) < \deg(g)$, $\deg(b) < \deg(f)$ and*

$$af + bg = \gcd(f, g).$$

Corollary 2.2.2 *Let k be a field and f, g coprime elements of $k[x]$. Then there exist unique elements a and b such that $\deg(a) < \deg(g)$, $\deg(b) < \deg(f)$ and*

$$af + gb = 1.$$

The gcd of two polynomials may be found by Euclid's algorithm (see Section 7.1) and the elements a and b above by the Extended Euclidean Algorithm.

Now let there be v indeterminates $x_1, \ldots, x_v$. A term of the form $x_1^{d_1} \cdots x_v^{d_v}$ with $d_i \in \mathbb{N}$ for $1 \leq i \leq v$ is called a *monomial.* A polynomial is a finite linear combination of monomials of the form

$$g = \sum_{(d_1,\ldots,d_v)\in\mathbb{N}^v} a_{d_1\ldots d_v} x_1^{d_1} \cdots x_v^{d_v}$$

with $a_{d_1\ldots d_v} \in R$. The set of such polynomials is denoted by $R[x_1, \ldots, x_v]$. The (total) degree of g is the largest value of $\sum d_i$ occurring in any monomial with a non-zero coefficient and the degree of g in the indeterminate x_i is the maximum value of d_i which occurs. Some discussion of canonical forms for multivariate polynomials can be found in [15].

When $k = 2$ it will be convenient to use x and y for the indeterminates and in this case we write $\deg_x(g)$ and $\deg_y(g)$ for the highest powers of x and y occurring in g and $\deg(g)$ for the total degree.

It is sometimes helpful to think of an element of $R[y, x]$ as a polynomial in x with coefficients in $R[y]$, that is, as an element of $R[y][x]$. Then if $\deg_x(g) = m$ we write

$$g = \sum_{i=1}^{m} b_i x^i.$$

Here the b_i are in $R[y]$. The leading coefficient in x of g, denoted by $\mathrm{lc}_x(g)$, is b_m.

2.3 The Computational Model

Many of the polynomials we shall look at will have coefficients either in $\mathbb{Z}$, $\mathbb{Z}_q$ for some prime power q, or $\mathbb{Q}$. We shall suppose that the target polynomial to be factorised is input as a degree n followed by $n + 1$ coefficients (known as dense encoding).

The time needed to carry out an integer arithmetic operation in $\mathbb{Z}$ is dependent on the sizes of the operands. It follows that the time required to factorise a

polynomial in $\mathbb{Z}[x]$ will depend on the sizes of its coefficients, that is to say on the length of the input which is

$$O\left(\log(n) + \sum_{r=0}^{n}\left[1 + \log(a_r')\right]\right),$$

where a_r' is the greater of $|a_r|$ and 1. The sum dominates the $\log(n)$ term. There are two ways in which we measure the coefficients of a polynomial f which we refer to as the size of f and the height $\mathrm{ht}(f)$. To make the next definition precise we shall adopt the convention that all logarithms are to the base 2.

Definition 2.3.1 *The* size *of an element of* $\mathbb{Z}[x]$ *in the form (2.1) is* $\log(|f|)$ *where*

$$|f| = \left(\sum_{r=0}^{n} a_r^2\right)^{1/2}. \tag{2.2}$$

Definition 2.3.2 *The* height *of an element of* $\mathbb{Z}[x]$ *in the form (2.1) is*

$$\mathrm{ht}(f) = \max_{0\le i\le n} |a_i|. \tag{2.3}$$

The size of the elements of $\mathbb{Z}_q$ is bounded above by $\log q$, so that arithmetic operations in this field can be regarded as taking a constant time.

As indicated in the first chapter, the classical factorisation algorithms require a time which is exponential in the degree and we shall be looking at algorithms which are of polynomial time complexity in the degree and the size of the input. In the remainder of this section we want to show that the essential ideas of factorisation algorithms can be found by looking at a slightly restricted class of problem.

Let $F(X)$ be an element of $\mathbb{Z}[X]$ with degree n and leading coefficient A_n. If we define $x = A_nX$ and $f(x) = A_n^{n-1}F(X)$ then f is monic in x. We therefore assume that our input polynomial is monic. The substitution of x/A_n for X changes the size of the polynomial, but a simple calculation shows that $|f| < (n+1)|F|$, so that the growth is not exponential.

Now let $\tilde{f}$ be in $\mathbb{Z}[x]$ and let g be the gcd of $\tilde{f}$ and its derivative. Then $f = \tilde{f}/g$ is squarefree. If h is a factor of f we may find the multiplicity of h as factor of $\tilde{f}$ by trial divisions. Thus it is sufficient to confine ourselves to squarefree polynomials.

It should be noted in passing that it is not a trivial exercise to find the gcd of two polynomials in $\mathbb{Q}[x]$ efficiently in practice. Some remarks are made about this in Chapter 7.

Finally let $\mathcal{A}$ be an algorithm which given as input a monic squarefree element f of $\mathbb{Z}[x]$ either finds an irreducible factor h_1 of f or shows that f itself is irreducible. If f is irreducible then $\mathcal{A}$ terminates and if not it is applied recursively to $h_2 = f/h_1$ to obtain a complete factorisation of f. Algorithms like $\mathcal{A}$ are the subject of this thesis.

In the case when the polynomial of interest comes from a different ring (such as $\mathbb{Z}[y, x]$) similar considerations apply to repeated factors. The derivation from a given polynomial of a polynomial monic in x can be done as follows. Suppose that $F \in \mathbb{Z}[X, Y]$ has degree n in X and d in Y:

$$f(Y, X) = \sum_{i=0}^{n} \sum_{j=0}^{d} a_{ij} X^i Y^j,$$

and that $\mathrm{lc}_X(F)$ is a polynomial in Y. Introduce a new indeterminate $y = Y - mX$. Then we have

$$\hat{F}(y, X) = F(y + mX, X) = \sum_{i=0}^{n} \sum_{j=0}^{d} X^i (y + mX)^j.$$

Clearly

$$\mathrm{lc}_y\left(X^i (y + mX)^j\right)$$

is independent of y. Hence $\mathrm{lc}_X(\hat{F})$ is a polynomial in m and we can find $m \in \mathbb{Z}$ such that it is non-zero. Thus the new choice of indeterminate transforms $F(Y, X)$ to $\hat{F}(y, X)$ which has the same total degree, $n + d$, as $F(Y, X)$ (but increased degree in X) and whose leading term in X has a coefficient in $\mathbb{Z}$. We can now proceed as we did in the univariate case to obtain a polynomial which is monic in x.

It should be noted that the growth in the number of the coefficients resulting from the substitution of $mX + y$ for Y may be exponential. To see this suppose that $F(Y, X)$ has the form

$$F(Y, X) = 1 + YX + Y^2X^2 + \cdots + Y^nX^n.$$

The sum of the coefficients of F is $n+1$ and $|F| = \sqrt{n+1}$. The term Y^pX^p becomes

$$Y^p(X+y)^p = Y^p \sum_{r=0}^{p} \binom{p}{r} X^{p-r} y^r. \tag{2.4}$$

The sum of the coefficients on the right of (2.4) is 2^p so that if $F(X+y, X) = \hat{F}(y, X)$ we find $|\hat{F}| > 2^{n/2}$. The complexity of our algorithms is measured in $\log(|F|)$, so that this results in at worst a blow-up which is polynomial (actually linear) in the total degree. However a polynomial with few non-zero coefficients (sparse) will in general be transformed to one with most coefficients non-zero (dense).

2.4 The Chinese Remainder Theorem

The Chinese Remainder Theorem is concerned with the solution of simultaneous congruences and originally (classically) was concerned with integer congruences. It now appears in the literature in a number of different forms: below we derive a version for univariate polynomials with rational coefficients which is suitable for our purposes.

From Corollary 2.2.2, with $k = \mathbb{Q}$, $f = q_1$ and $g = q_2$ we have the following lemma.

Lemma 2.4.1 *Let q_1 and q_2 be coprime elements of $\mathbb{Q}[x]$. Then there is a unique element $a \in \mathbb{Q}[x]$ of degree less than $\deg(q_2)$ such that $aq_1 \equiv 1 \pmod{q_2}$.*

This result is valid if $\mathbb{Q}$ is replaced by any other field and in particular holds for polynomials in $\mathbb{Z}_p[x]$, where p is a prime number.

Theorem 2.4.1 (Chinese Remainder Theorem) *Let $q \in \mathbb{Q}[x]$ be the product of r pairwise coprime polynomials $q_1, \ldots, q_r$; and let $a_j \in \mathbb{Q}[x]$ for $j = 1, \ldots, r$. Then the simultaneous congruences*

$$a \equiv a_j \pmod{q_j}, \quad \textit{for} \quad 1 \le j \le r, \tag{2.5}$$

have a solution

$$a = \sum_{j=1}^{r} a_j b_j (q/q_j) \pmod{q}, \tag{2.6}$$

where b_j is the element of $\mathbb{Q}[x]$, unique modulo q_j, such that

$$b_j(q/q_j) \equiv 1 \pmod{q_j}. \tag{2.7}$$

Furthermore the solution given by (2.6) is unique modulo q.

Proof We know that b_j exists and is unique from Lemma 2.4.1 because q_j and q/q_j are coprime. We also know that if $i \neq j$ then $q/q_j \equiv 0 \pmod{q_i}$ because q_i is a factor of q/q_j. It follows that the expression on the right of equation (2.6) satisfies each of the equations (2.5).

If equations (2.5) had two solutions a and a' then we should have

$$a - a' \equiv 0 \pmod{q_j}, \quad \text{for } 1 \leq j \leq r,$$

and hence $a \equiv a' \pmod{q}$, because the q_j are coprime. □

The CRT holds if $\mathbb{Q}[x]$ is replaced by $\mathbb{Z}_p[x]$.

2.5 Berlekamp's Algorithm for factoring $f \in \mathbb{Z}_p[x]$

2.5.1 The theoretical background

Suppose $f \in \mathbb{Z}_p[x]$ is squarefree, has degree n and factorizes into a product of r irreducible elements $f_1, \ldots, f_r$. We want a way to determine the f_k. If the leading coefficient a_n of f is not zero then it has a multiplicative inverse a_n^{-1} and so f may be rescaled to have leading coefficient one. This means that it is sufficient to find an algorithm which factorises monic elements of $\mathbb{Z}_p[x]$. We quote

Theorem 2.5.1 (Fermat) *If p is prime and $a \in \mathbb{Z}_p$ then $a^p = a$.*

Fermat's theorem can be generalised to the following lemma which we shall need later in this section.

Lemma 2.5.1 *If* $v(x) \in \mathbb{Z}_p[x]$ *then* $v(x)^p = v(x^p)$.

Proof Consider an element v of $\mathbb{Z}_p[x]$ of the form $\sum_{k=0}^{m} b_k x^k$. Because the binomial coefficient $\binom{p}{r}$ divides by p if $r \notin \{0, p\}$, it follows by induction on m that

$$v(x)^p = (\sum_{k=o}^{m} b_k x^k)^p = \sum_{k=0}^{m} b_k^p x^{pk}.$$

Then by Fermat's Theorem we have $b_k^p = b_k$ so that

$$v(x)^p = \sum_{k=o}^{m} b_k x^{pk} = v(x^p). \tag{2.8}$$

□

Since the number of polynomials of degree less than n is finite, this means that all the irreducible factors of f can be found by a search. However a brute-force search may require examining all polynomials whose degree does not exceed $\lfloor n/2 \rfloor$ and there are $O(p^{n/2})$ of these. Berlekamp's algorithm reduces the size of this search by determining a small set G of polynomials from which to choose the g. In addition it determines how many irreducible factors f has before the search begins.

If g is any polynomial in $\mathbb{Z}_p[x]$ then $\gcd(f, g)$ divides f. If it is also the case that

$$1 \leq \deg(\gcd(f, g)) < n = \deg(f)$$

then $\gcd(f, g)$ will be a proper factor of f. We therefore want to find polynomials $g \in \mathbb{Z}_p[x]$ which have the property that $\gcd(g, f) = f_k$ for some factor f_k of f. Since the f_k are unknown at this stage we look instead for polynomials v which are divisible by f and which can be factorized. The condition that v is divisible by f is that

$$v \equiv 0 \pmod{f},$$

and this motivates the method.

If u is an indeterminate, Fermat's theorem shows that the polynomial $u^p - u \in \mathbb{Z}_p[u]$ has every element of $\mathbb{Z}_p$ as a zero so:

$$u^p - u = (u - 0)(u - 1)(u - 2) \cdots (u - (p - 1)). \tag{2.9}$$

In particular this identity holds if we replace the indeterminate u by any polynomial $v \in \mathbb{Z}_p[x]$.

Berlekamp's idea is to seek solutions in $\mathbb{Z}_p[x]$ of

$$v^p - v \equiv 0 \pmod{f}. \tag{2.10}$$

Each solution v_j of (2.10) will satisfy

$$(v_j - 0)(v_j - 1)\cdots(v_j - (p-1)) \equiv 0 \pmod{f}. \tag{2.11}$$

A way of solving (2.10) will be explained below: for the moment we suppose that solutions $v_1 \ldots v_t$ have been found. The terms $(v_j - a) \in \mathbb{Z}_p[x]$ which appear on the left side of (2.11) are the elements of the set G mentioned above.

All the factors of f divide the left hand side of (2.11). Further, if one of the factors f_k of f divides both $(v_j - a)$ and $(v_j - b)$ then $v_j \equiv a \pmod{f_k}$ and $v_j \equiv b \pmod{f_k}$ which implies that $a \equiv b \pmod{f_k}$ i.e.,that $a = b$. So for a particular solution v_j of (2.10) each f_k divides $(v_j - a)$ in (2.11) for exactly one value of a. It follows that there exist elements $a_i \in \mathbb{Z}_p$ for $1 \leq i \leq r$ such that

$$v_j \equiv a_i \pmod{f_i} \quad \text{for} \quad 1 \leq i \leq r. \tag{2.12}$$

Since the f_i are co-prime the Chinese Remainder Theorem shows that equations (2.12) have a unique solution modulo f for each set of values $a_1, \ldots, a_r$. Furthermore for each solution v_j of (2.12)

$$v_j^p \equiv a_i^p \equiv a_i \equiv v_j \pmod{f_k} \quad \text{for} \quad 1 \leq i \leq r,$$

so that

$$v_j^p \equiv v_j \pmod{f},$$

which is equivalent to (2.10). This proves that the solutions of (2.12), with the a_i allowed to range over $\mathbb{Z}_p$, are precisely the solutions of (2.10), a result known as Berlekamp's Theorem. Note that (2.12) leads to p^r systems each with a different solution set $\{v_j\}$. Thus (2.10)has p^r solutions. We shall see below that the set of all these solutions is a vector space over $\mathbb{Z}_p$ and so it must have dimension r.

The factors of f are determined by examining $\gcd(f, v_j - a)$ for each solution v_j of (2.10) and each $a \in \mathbb{Z}_p$. Now we have to show that all the factors f_k can be distinguished in this way. We shall prove this by establishing that if $f_j \neq f_i$ then there is a solution v^* of (2.10) such that f_k, f_l do not divide the same factor $(v^* - a)$ in (2.11).

If we regard (2.12) as a set of modular equations, they have a solution v^* which is unique modulo f and is also a solution of (2.10). Suppose we choose $a_k \neq a_l$. Since f_k divides $(v^* - a_i)$ only if $i = k$ we must have $(v^* - a_k) \equiv 0 \pmod{f_k}$ and $(v^* - a_l) \not\equiv 0 \pmod{f_k}$; while at the same time $(v^* - a_l) \equiv 0 \pmod{f_l}$ and $(v^* - a_k) \not\equiv 0 \pmod{f_k}$. Thus there is a solution v^* of (2.10) which distinguishes f_i from f_j. This means that the equations $\gcd(f, v^* - a_k) = f_k$ and $\gcd(f, v^* - a_l) = f_l$ hold.

Berlekamp's Theorem says that the solutions of the non-linear equation (2.10) form a linear space of dimension r. This superficially surprising result is a consequence of doing arithmetic in $\mathbb{Z}_p$ as the following shows. Let $w = au_1 + bu_2$ where u_1, u_2 are two of the v_j and a, b are elements of $\mathbb{Z}_p$. Then

$$w^p = (au_1 + bu_2)^p = a^p u_1^p + b^p u_2^p = au_1 + bu_2 = w,$$

so that w is also a solution of (2.10).

It remains to find the solutions of (2.10). To do this we represent a polynomial by a column c of its coefficients. If $v(x) \in \mathbb{Z}_p[x]$ is any polynomial then $(v(x) \bmod f)$ has at most n non-zero coefficients and can be represented by an n-vector. Let P be the matrix whose $(k+1)$st column represents x^{pk} modulo f and (abusing notation) let v be the vector of the polynomial v in $\mathbb{Z}[x]$. Then Pv represents $(v(x^p) \bmod f)$ or, by Lemma 2.5.1, $(v(x)^p \bmod f)$. So in our notation equation (2.10) becomes

$$(P - I)v = 0.$$

The solutions are the vectors which span the null-space of $P - I$ and the dimension of the null space is equal to r, the number of irreducible factors of f, by the arguments above. The v_j and the value of r can be found by triangulating $P - I$,

```
begin
  form the matrix P;
  triangulate P;
  determine r;
  determine v_1, ..., v_r;
  count := 1;  {count equals the number of factors found}
  for j from 2 to r do
    for a from 0 to p - 1 do
      begin
        examine gcd(f, v_j - a) for a factor;
        if a factor is found then count := count + 1;
        if count = r then return
      end
end
```

Figure 2–1: Berlekamp's Algorithm

which verifies the assertion that the number of distinct factors f_k is determined before searching the values of $\gcd(f, v_j - a)$ to find them.

Finally one last observation needs to be made. Every element of $\mathbb{Z}_p$ solves equation (2.10) and this means that one of the v_j is $(1, 0, \ldots, 0)^T$, which we label v_1. Since v_1 is a polynomial of degree zero, only $v_2, \ldots, v_r$ need be considered in the $v_j - a$.

2.5.2 Complexity

The complexity of the algorithm is determined by the triangulation of $P - I$ and the gcd calculations. The measure of complexity is the number of arithmetic operations which are carried out in $\mathbb{Z}_p$. The size of the operands is bounded by $\log(p)$, which does not depend on f.

That triangulation of a matrix can be carried out in $O(n^3)$ operations and a gcd calculation in $O(n^2)$ operations are standard results. The maximum number of gcd calculations is rp so that the total complexity is bounded by $O(n^3)+O(n^2pr)$. Since f could have n factors this gives a worst case complexity $O(n^3p)$, which depends not only on the degree of f but also on p.

2.6 Hensel Lifting

Suppose we are given a polynomial $f \in \mathbb{Z}[x]$ which is monic and squarefree and some prime p such that $(f \bmod p)$ is squarefree. We can regard $(f \bmod p)$ as an element of $\mathbb{Z}_p[x]$ and apply Berlekamp's algorithm. Suppose this algorithm finds two coprime factors g_1 and h_1 such that

$$(f \bmod p) \equiv g_1 h_1. \tag{2.13}$$

If, by an abuse of notation, we regard g_1 and h_1 as elements of $\mathbb{Z}[x]$ whose coefficients lie in the interval $[0, p-1]$, equation (2.13) can be written as

$$f \equiv g_1 h_1 \pmod{p}. \tag{2.14}$$

The Hensel lifting method gives us a way of going from (2.14) to obtain new polynomials g_2 and h_2 in $\mathbb{Z}[x]$ such that

$$f \equiv g_2 h_2 \pmod{p^2}, \tag{2.15}$$

and in general to find g_k and h_k such that

$$f \equiv g_k h_k \pmod{p^k}. \tag{2.16}$$

Since f is monic then $f \not\equiv 0 \pmod{p^k}$ and (2.16) imply that if g_k and h_k exist then $g_k \not\equiv 0 \pmod{p^k}$ and $h_k \not\equiv 0 \pmod{p^k}$ for all $k \in \mathbb{Z}^+$. Further we can assume that g_k and h_k are monic. Let $g_2 = g_1 + \gamma$ and $h_2 = h_1 + \kappa$, where f_1, g_1 satisfy (2.14). Since (2.15) implies that $f \equiv g_2 h_2 \pmod{p}$ we find

$$g_1\kappa + h_1\gamma + \gamma\kappa \equiv 0 \pmod{p}.$$

This equation is satisfied if $\gamma \equiv 0 \pmod{p}$ and $\kappa \equiv 0 \pmod{p}$. We therefore write $g_2 = g_1 + p\hat{g}_2$ and $h_2 = h_1 + p\hat{h}_2$. Substituting these expressions in (2.15) yields

$$\hat{g}_2 h_1 + \hat{h}_2 g_1 \equiv (f - g_1 h_1)/p \pmod{p}. \tag{2.17}$$

The division on the right of (2.17) is exact by virtue of (2.14). The degrees of $\hat{g}_2$ and $\hat{h}_2$ do not exceed those of g_1 and h_1 respectively.

Since g_1 and h_1 are coprime Corollary 2.2.2 shows that the equation

$$ag_1 + bh_1 \equiv 1 \pmod{p} \tag{2.18}$$

has a solution (a, b), with $\deg(a) < \deg(h_1)$ and $\deg(b) < \deg(g_1)$, which is unique modulo p. The quantities $\hat{g}_2$ and $\hat{h}_2$ are obtained from it by multiplication:

$$\begin{aligned} \hat{g}_2 &= b(f - g_1 h_1)/p \pmod{p} \\ \hat{h}_2 &= a(f - g_1 h_1)/p \pmod{p} \end{aligned} \tag{2.19}$$

In the general case, if $f \equiv g_{k-1} h_{k-1} \pmod{p^{k-1}}$, the polynomials $g_k = g_{k-1} + p\hat{g}_k$ and $h_k = h_{k-1} + p\hat{h}_k$ will satisfy $f \equiv g_k h_k \pmod{p^k}$ provided $\hat{g}_k$ and $\hat{h}_k$ are chosen so that they satisfy

$$(f - g_{k-1} h_{k-1})/p^{k-1} \equiv \hat{g}_k h_{k-1} + \hat{h}_k g_{k-1} \equiv \hat{g}_k h_1 + \hat{h}_k g_1 \pmod{p}. \tag{2.20}$$

This is the same as (2.17) with a different right hand side, so that the solution is of the same form as (2.19). Thus the Hensel lifting method requires that the solution (a, b) of (2.18) be found once and then for $k = 2, 3, 4 \ldots$, $\hat{g}_k$ and $\hat{h}_k$ can be found by a single multiplication:

$$\begin{aligned} \hat{g}_k &= b(f - g_{k-1} h_{k-1})/p^{k-1} \pmod{p} \\ \hat{h}_k &= a(f - g_{k-1} h_{k-1})/p^{k-1} \pmod{p} \end{aligned} \tag{2.21}$$

Note that $\deg(g_k) \leq \deg(g_1)$ and $\deg(h_k) \leq \deg(h_1)$.

Hensel lifting need not lead to a factorisation of f in $\mathbb{Z}[x]$ as k increases. For example $f = x^2 + 1$ is irreducible in $\mathbb{Z}[x]$ but has a factorisation of the form

$$x^2 + 1 \equiv (x + a)(x + b) \pmod{5^k}$$

for all $k \in \mathbb{Z}^+$. The first three are $(x+2)(x+3)$, $(x+7)(x+18)$ and $(x+57)(x+68)$.

2.7 Resultants and Subresultants

2.7.1 Resultants

Let J be an integral domain and $f, g \in J[x]$. Suppose that $f = \sum_{i=0}^{n} \phi_i x^i$ and $g = \sum_{i=0}^{m} \gamma_i x^i$. The resultant $R(f,g)$ of f and g is the determinant of the matrix

$$\begin{pmatrix}
\phi_n & \phi_{n-1} & \cdots & \phi_0 & & & \\
 & \phi_n & \phi_{n-1} & \cdots & \phi_0 & & \\
 & & \ddots & \ddots & \ddots & \ddots & \\
 & & & \phi_n & \phi_{n-1} & \cdots & \phi_0 \\
\gamma_m & \gamma_{m-1} & \cdots & \gamma_0 & & & \\
 & \gamma_m & \gamma_{m-1} & \cdots & \gamma_0 & & \\
 & & \ddots & \ddots & \ddots & \ddots & \\
 & & & \gamma_m & \gamma_{m-1} & \cdots & \gamma_0
\end{pmatrix},$$

in which there are m rows of ϕ_i and n rows of γ_i. (The alignments in the centre column are accidental and should be ignored.) The blank entries are zero. It can be shown [57] that if f and g have a factor of positive degree then $R(f,g) = 0$.

Suppose that $f, g \in \mathbb{Q}[y][x]$ and write their resultant as $R(f,g)(y) \in \mathbb{Q}[y]$. If $R(f,g)(y)$ vanishes when evaluated at $y = b$ and $\phi_n(b)\gamma_m(b) \neq 0$, then $f(b,x)$ and $g(b,x)$ (which are in $\mathbb{Q}[x]$) have a common factor. The number of elements $b \in \mathbb{Q}$ for which this can happen is finite.

If $f(b,x)$ has a repeated factor then $f(b,x)$ and $f_x(b,x)$ will have a common factor of positive degree and so $R(f,f_x)(b) = 0$. The argument in the previous paragraph shows that we can find an element $c \in \mathbb{Q}$ (or $c \in \mathbb{Z}$) such that $R(f,f_x)(c) \neq 0$, so that $f(c,x)$ is squarefree. We assume, by making a change of variable if necessary, that $c = 0$.

2.7.2 Subresultants

We are often faced with the question of whether two polynomials are coprime. The standard algebraic test for this is that their resultant will be non-zero if and only if they are coprime [57]. In this section we look at a generalisation of the resultant, which is explained with the aid of an example.

Let k be a field, $f = \sum_{r=0}^{6} \phi_r x^r$ and $g = \sum_{r=0}^{4} \gamma_r x^r$ be elements of $k[x]$. We shall show how to tell if f and g have a common factor of degree three or more. Intuitively the argument is that if f and g have a common factor then their coefficients will have some kind of linear dependence.

If d divides both f and g then d must also divide $af + bg$ for any $a, b \in k[x]$. In particular if $\deg(af + bg) \leq 2$ and $\deg(d) \geq 3$ we must have $af + bg = 0$. We shall see that to determine whether f and g have a common factor of degree at least three what we examine are the terms in xf, f, x^3g, x^2g, xg and g whose degree is at least three. These are

$$\begin{array}{rccccccccc}
xf: & \phi_6 x^7 & + & \phi_5 x^6 & + & \phi_4 x^5 & + & \phi_3 x^4 & + & \phi_2 x^3 \\
f: & & & \phi_6 x^6 & + & \phi_5 x^5 & + & \phi_4 x^4 & + & \phi_3 x^3 \\
x^3 g: & \gamma_4 x^7 & + & \gamma_3 x^6 & + & \gamma_2 x^5 & + & \gamma_1 x^4 & + & \gamma_0 x^3 \\
x^2 g: & & & \gamma_4 x^6 & + & \gamma_3 x^5 & + & \gamma_2 x^4 & + & \gamma_1 x^3 \\
xg: & & & & & \gamma_4 x^5 & + & \gamma_3 x^4 & + & \gamma_2 x^3 \\
g: & & & & & & & \gamma_4 x^4 & + & \gamma_3 x^3.
\end{array}$$

Now form the matrix M:

$$M = \begin{pmatrix}
\phi_6 & & \gamma_4 & & & \\
\phi_5 & \phi_6 & \gamma_3 & \gamma_4 & & \\
\phi_4 & \phi_5 & \gamma_2 & \gamma_3 & \gamma_4 & \\
\phi_3 & \phi_4 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\
\phi_2 & \phi_3 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 \\
xf & f & x^3 g & x^2 g & xg & g
\end{pmatrix}.$$

The blank entries denote zeros. Rows one to five of M contain the coefficients of $x^7, x^6, \ldots, x^3$ of the entries of the elements in the corresponding columns of the sixth row. The determinant of M can be written $\sum_{r=0}^{7} \det(\mu_r) x^r$, μ_r being the

6×6 matrix which is identical to M in its first five rows and whose sixth row contains the coefficients of x^r in the sixth row of M. Thus by construction μ_r has two identical rows and $\det(\mu_r) = 0$ for $3 \leq r \leq 7$. The determinant of M is therefore a polynomial of degree at most two and

$$\det(M) = \sum_{r=0}^{2} \det(\mu_r) x^r. \tag{2.22}$$

The greatest common divisor of f and g divides every element on the sixth row of M and hence divides $\det(M)$. This implies that if $\deg(\gcd(f,g)) > \deg(\det(M)) = 2$ then $\det(M) = 0$.

In this example $\det(M)$ is the second order subresultant of f and g and this subresultant vanishes if f and g have a common factor of degree more than two.

We now generalise to the case in which f and g are allowed to have arbitrary degree. First we shall construct a vector corresponding to the last row of M and then we shall define matrices corresponding to the μ_r.

Let $f = \sum_{r=0}^{n} \phi_r x^r$ and $g = \sum_{r=0}^{m} \gamma_r x^r$ where $n \geq m > 0$, $\phi_n \neq 0$ and $\gamma_m \neq 0$. Then for all j satisfying $0 \leq j < m$ we have

$$\deg(x^{m-j-1} f) = \deg(x^{n-j-1} g) = n + m - j - 1$$

and we define an $(n + m - 2j)$-vector b_j by

$$b_j = (x^{m-j-1} f, x^{m-j-2} f, \ldots, xf, f, x^{n-j-1} g, x^{n-j-2} g, \ldots, xg, g).$$

In the example above the sixth row of M was b_2 with $n = 6$, $m = 4$ and $j = 2$.

Next denote by $T_j(f,g)$ a square matrix of order $(n + m - 2j)$ whose i-th row consists of the row of coefficients of $x^{n+m-j-i}$ in b_j, for $1 \leq i \leq n + m - 2j - 1$, and whose last row is b_j

Definition 2.7.1 *The j-th subresultant of f and g is $S_j(f,g) = \det(T_j(f,g))$.*

The top row of T_j contains the coefficients in b_j of $x^{n+m-j-1}$ and the second last row those of x^{j+1}, so arguing as in the example we deduce that $S_j(f,g)$ is a polynomial of degree at most j.

If the last row b_j of T_j is replaced by the row of coefficients of x^r in b_j we obtain a constant matrix $t_{j,j-r}(f,g)$ whose determinant we denote by $s_{j,j-r}(f,g)$. A line of argument like the one that led to equation (2.22) shows that

$$S_j(f,g) = \sum_{r=0}^{j} s_{j,j-r}(f,g)x^r. \tag{2.23}$$

In the case $j = 0$ it is clear that $S_0(f,g)$ is an element of k and from (2.23) equals $s_{0,0}(f,g)$. This is the classical resultant.

Since $\gcd(f,g)$ divides every element of b_j it follows that it is a factor of the j-th subresultant which will therefore be zero whenever $\deg(\gcd(f,g)) > \deg(S_j(f,g)) = j$. It follows that

$$\deg(\gcd(f,g)) \leq \min\{j \mid S_j(f,g) \neq 0\}. \tag{2.24}$$

In fact, as shown in [8], the inequality in 2.24 can be replaced by equality.

For a specific example, let $k = \mathbb{Z}$, $f = x^4 - 1 = (x^2+1)(x^2-1)$ and $g = x^4+x^3+2x^2+x+1 = (x^2+x+1)(x^2+1)$. Then $n = m = 4$ and $\gcd(f,g) = x^2+1$. It is readily verified that

$$\begin{aligned} S_3(f,g) &= (x^2+1)(x+2), \\ S_2(f,g) &= 3(x^2+1), \\ S_1(f,g) &= 0(x^2+1). \end{aligned}$$

It will be useful later to have expressions for the $s_{j,j-r}$ and to see how these can be found we return to the example matrix M. In terms of the general definition M is $T_2(f,g)$ and

$$\det(M) = S_2(f,g) = \sum_{r=0}^{2} s_{2,2-r}(f,g)x^r.$$

The last row of M is b_2 and the rows of $t_{2,0}$ contain the coefficients in b_2 of x^7 down to x^2:

$$t_{2,0} = \begin{pmatrix} \phi_6 & & \gamma_4 & & & \\ \phi_5 & \phi_6 & \gamma_3 & \gamma_4 & & \\ \phi_4 & \phi_5 & \gamma_2 & \gamma_3 & \gamma_4 & \\ \phi_3 & \phi_4 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \phi_2 & \phi_3 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 \\ \phi_1 & \phi_2 & & \gamma_0 & \gamma_1 & \gamma_2 \end{pmatrix}.$$

Suppose we extend the matrix $t_{2,0}$ with a seventh and eighth row , containing respectively the coefficients of x and the constant terms in b_2, to obtain a matrix $t'_{2,0}$:

$$t'_{2,0} = \begin{pmatrix} \phi_6 & & \gamma_4 & & & \\ \phi_5 & \phi_6 & \gamma_3 & \gamma_4 & & \\ \phi_4 & \phi_5 & \gamma_2 & \gamma_3 & \gamma_4 & \\ \phi_3 & \phi_4 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \phi_2 & \phi_3 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 \\ \phi_1 & \phi_2 & & \gamma_0 & \gamma_1 & \gamma_2 \\ \phi_0 & \phi_1 & & & \gamma_0 & \gamma_1 \\ & \phi_0 & & & & \gamma_0 \end{pmatrix}.$$

Note that the columns are just the coefficients of xf, f, x^3g, x^2g, xg and g. The column operations which reduce $t_{2,0}$ to a lower triangular form $\hat{t}_{2,0}$ also reduce $t'_{2,0}$ to a lower triangular form $\hat{t}'_{2,0}$. The columns of $t'_{2,0}$ may be interpreted as the corresponding polynomials x^pf and x^qg which occur as elements of b_2. Thus the triangulation process for $t'_{2,0}$ may be thought of as constructing polynomials λ_i and μ_i in $k[x]$ such that

$$\lambda_i f + \mu_i g = h_i, \quad \text{for } 2 \leq i \leq 7, \tag{2.25}$$

where h_i is the polynomial of degree i corresponding to the diagonalised $(8-i)$-th column of $\hat{t}'_{2,0}$ of degree i. Since the first two columns of $t'_{2,0}$ are already in lower triangular form we have $\mu_7 = \mu_6 = 0$, $\lambda_7 = x$ and $\lambda_6 = 1$.

Diagonalisation by column operations does not change the value of the determinant so that $s_{2,0}$ is the product of the diagonal elements of $\hat{t}'_{2,0}$, which are the leading coefficients of the h_i:

$$s_{2,0} = \prod_{i=2}^{7} \mathrm{lc}(h_i).$$

In the general case given by equation (2.23) $t_{j,0}$ is a square matrix of order $(n+m-2j)$ and the equation corresponding to (2.25) is

$$\lambda_i f + \mu_i g = h_i \text{ for } j \leq i \leq n+m-j-1. \tag{2.26}$$

Here h_i is the polynomial of degree i corresponding to the diagonalised $(n+m-j-i)$-th column. The values of μ_i and λ_i are given by

$$\mu_i = 0 \text{ and } \lambda_i = x^{m-j-i} \text{ for } n \leq i \leq n+m-j-1; \tag{2.27}$$

and

$$s_{j,0} = \prod_{i=j}^{m+n-j+1} \mathrm{lc}(h_i). \tag{2.28}$$

In the particular case when f is monic, $\mathrm{lc}(f) = 1$ so that $\mathrm{lc}(h_i) = 1$ for $n \leq i \leq n+m-j-1$ and

$$s_{j,0} = \prod_{i=j}^{n-1} \mathrm{lc}(h_i). \tag{2.29}$$

We remark finally that the results in this section up to (2.24) are unchanged if the field k is replaced by a commutative ring R, but triangulation of a matrix is not usually possible in R without changing the value of its determinant.

2.8 Algebraic Extensions of $\mathbb{Z}$ and $\mathbb{Q}$

Let R and S be UFDs such that $R \subset S$ and let $G \in R[t]$ be irreducible. Suppose there is an element $\alpha \in S$ such that $G(\alpha) = 0$ in S. Then the set $R \cup \{\alpha\}$ with the operations of S generates a subring of S usually written $R[\alpha]$ and called an *(algebraic) extension* of R. The root α is called an *algebraic element* over R and in the case when G is monic α is said to be *integral* over R. The elements of $R[\alpha]$ are polynomials in α of degree less than $\deg(G)$.

The primitive part of G is called the *minimal polynomial*. If the roots of G are $\alpha = \alpha_1, \alpha_2, \ldots, \alpha_m$ then $\alpha_2, \ldots, \alpha_m$ are called the *conjugates* of α.

If k, k' are the fields of quotients of R, S then $\alpha \in k'$ and we can form an extension field of k denoted by $k(\alpha)$. The elements of $k(\alpha)$ are rational functions of α.

The element α is called an *algebraic number* in the case when $R = \mathbb{Z}$ and if G is monic it is called an *algebraic integer*. The algebraic integers of $K = \mathbb{Q}(\alpha)$ form a ring denoted O_K.

Example 2.8.1 Since $\sqrt{5}$ satisfies the equation $x^2 - 5 = 0$ it is an algebraic integer. Furthermore the polynomial $x^2 - x - 1$ factorises as

$$\left(x - \frac{\sqrt{5}+1}{2}\right)\left(x + \frac{\sqrt{5}-1}{2}\right)$$

so that $(\sqrt{5}+1)/2$ and $(\sqrt{5}-1)/2$ are both algebraic integers.

The extensions of $\mathbb{Z}$ by algebraic integers are a proper subset of the algebraic extensions of $\mathbb{Z}$, for an extension by an algebraic number which is the root of a non-monic polynomial is not in general equivalent to an extension by an algebraic integer.

Let α be an algebraic integer over $\mathbb{Z}$ with minimal polynomial g. The discriminant of g, denoted $\mathrm{discr}(g)$, is the quantity

$$\mathrm{discr}(g) = \prod_{i<j}(\alpha_i - \alpha_j)^2.$$

It may be shown (see [33], Proposition 2.1) that there is a $d \in \mathbb{N}$ such that d^2 divides $\mathrm{discr}(g)$ and such that the ring of algebraic integers of $\mathbb{Q}(\alpha)$ is contained in $(1/d)\mathbb{Z}[\alpha]$.

Definition 2.8.1 *Let α be an algebraic integer and let d be the smallest element of $\mathbb{N}$ for which the ring of algebraic integers of $\mathbb{Q}(\alpha)$ is contained in $(1/d)\mathbb{Z}[\alpha]$. We call d the* defect *of α.*

We shall need the defect in later chapters.

For a field k which is the field of quotients of a ring R, however, an extension by an algebraic number is always isomorphic to an extension by an algebraic integer. To see why this is we recollect from section 2.3 that if we define a new indeterminate $u = A_n t$, the polynomial $f(u) = A_n^{n-1}F(t)$ is monic and $\beta = A_n\alpha$ is a root of f. Then since division is possible in a field we find

$$k(\alpha) = k(\beta/A_n) = k(\beta).$$

It follows that the minimal polynomial of an extension $k(\beta)$ may, without loss of generality, be chosen to be a monic polynomial in $R[t]$. In particular if $\mathbb{Q}(\beta)$ is an extension of the rationals, the minimal polynomial is a monic element of $\mathbb{Z}[t]$.

Now suppose that $f(x) \in \mathbb{Q}(\beta)[x]$. Then

$$f(x) = \left(\frac{1}{n(\beta)}\right) F(x)$$

for some $n(\beta) \in \mathbb{Z}[\beta]$ and some $F(x) \in \mathbb{Z}[\beta][x]$, so that the problem of finding factors in $\mathbb{Q}(\beta)[x]$ of $F(x)$ and of $f(x)$ are in this sense equivalent, the minimum polynomial of β in $\mathbb{Z}[t]$ being assumed monic. Furthermore an extension of Gauss's lemma states that if $f \in O_k[x]$ is monic then f factorises in $k[x]$ if and only if it factorises in $O_k[x]$.

Chapter 3

Lenstra, Lenstra and Lovász

3.1 Introduction

The most important paper on the factorisation of polynomials in this survey is [40], subsequently referred to as LLL, in which the authors proved for the first time that the factors of a primitive polynomial $f \in \mathbb{Z}[x]$ can be determined in a time which is polynomial in the degree of f and in $\log |f|$. The precise result is given in §3.5.

Factorisation of a primitive polynomial in $\mathbb{Z}[x]$ is equivalent to factorisation of a polynomial in $\mathbb{Q}[x]$. To see why this is, suppose that $f \in \mathbb{Z}[x]$ is primitive and has a factorisation

$$f(x) = f_1(x)f_2(x)\cdots f_r(x),$$

in which each f_j is in $\mathbb{Q}[x]$. We can write

$$f_j(x) = \sum_{i=0}^{d_j} \frac{a_{ij}}{b_{ij}} x^i,$$

where we assume that $\gcd(a_{ij}, b_{ij}) = 1$. Then if

$$A_j = \gcd(a_{0j}, \ldots, a_{d_j j}) \text{ and } B_j = \operatorname{lcm}(b_{0j}, \ldots, b_{d_j j})$$

we have

$$f_j(x) = \frac{A_j}{B_j} \sum_{i=0}^{d_j} c_{ij} x^i = \frac{A_j}{B_j} F_j,$$

where each F_j is in $\mathbb{Z}[x]$ and is primitive. By Corollary 2.2.1 to Gauss's Lemma it follows that

$$f(x) = F_1(x)F_2(x)\cdots F_r(x),$$

so that each f_j is a product of F_j and a (non-zero) element of $\mathbb{Q}$. The preceding discussion shows how the F_j can be determined from the f_j.

Given a non-primitive $\hat{f}$ in $\mathbb{Z}[x]$ we may determine its content γ by gcd calculations to obtain $\hat{f} = \gamma f$ and a factorisation of f will lead to a factorisation of $\hat{f}$, up to multiplication by an element of $\mathbb{Q}$.

The overall idea of the LLL algorithm is that factors of (f mod p) should give some information about factors of f in $\mathbb{Z}[x]$, and that factors of (f mod p^k) should give more. However, as the example in §2.6 makes clear, increasing k will not lead to factors of f itself in general. The factors in $\mathbb{Z}[x]$ may be determined by a finite search (as was known to Kronecker), but the new aspect of the LLL algorithm is that this search can be made quick (actually trivial) by a suitable preconditioning which can be done in polynomial time.

In order to explain the algorithm it is necessary to describe some theoretical ideas and state some results from the paper. As details can be found in the original paper we omit formal proofs and instead try to give the motivation. We describe *lattices*, an idea from geometrical number theory, and an associated algorithm known as the basis reduction algorithm in §3.2. The basis reduction is the preconditioning mentioned in the previous paragraph. The connection between lattices and the factorisation algorithm is explained in §3.3. The factorisation algorithms are shown in §3.4 and their complexity in §3.5.

3.2 Lattices and the basis reduction algorithm

3.2.1 Definitions and properties

The following definition of a lattice is taken from [12].

Definition 3.2.1 *Let $b_1, b_2, \ldots, b_n$ be a real basis for the vector space $\mathbb{R}^n$. The set of all sums of integral multiples of the basis vectors is called a* lattice.

In the application of lattices to factorisation in $\mathbb{Z}[x]$ the basis vectors will be in $\mathbb{Z}^n$.

Example 3.2.1 If the standard unit vectors $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are used as a basis for the vector space $\mathbb{R}^3$ then the corresponding lattice consists of all points with integral coordinates.

Example 3.2.2 The vectors $(2,0)$ and $(2,1)$ form a basis for $\mathbb{R}^2$ and the lattice L with this basis contains all the points with integral coordinates the first of which is even. In this case $(2,0)$ and $(0,1)$ form another basis for $\mathbb{R}^2$ and L while the pair (1,0) and (0,1) do for $\mathbb{R}^2$ but not for L. The basis of a lattice is not unique but the choice is restricted.

If a lattice L has basis $b_1, \ldots, b_n$ then the *determinant* of L is defined by

$$d(L) = |\det(b_1, \ldots, b_n)|,$$

the b_i being written as column vectors. The value of $d(L)$ is independent of the choice of basis [12]. A geometrical interpretation of $d(L)$ is that it is the volume of the smallest n-dimensional parallel-sided box whose edges are lattice vectors. If the lattice vectors were orthogonal this box would have volume $\prod_{i=1}^{n} |b_i|$ where $|u|$ denotes the Euclidean length of u. In the general case the volume is less than this and we have Hadamard's inequality:

$$d(L) \leq \prod_{i=1}^{n} |b_i|.$$

Among the various bases for a lattice L in $\mathbb{R}^n$ some may have the property of being *reduced.* An explanation of this is given after the formal definition. A point to note is that the order of the elements in a reduced basis is significant and we shall suppose throughout that all bases have their elements prescribed in a given order.

Definition 3.2.2 *Let* $u = (u_1, \ldots, u_n)$ *and* $v = (v_1, \ldots, v_n)$ *be elements of* $\mathbb{R}^n$. *The* inner product $\langle u, v \rangle$ *of* u *and* v *is*

$$\langle u, v \rangle = \sum_{i=1}^{n} u_i v_i.$$

Definition 3.2.3 *Given an ordered basis $b_1, b_2, \ldots, b_n$ of a lattice L in $\mathbb{R}^n$ we can obtain a new ordered basis $b_1^*, b_2^*, \ldots, b_n^*$ for $\mathbb{R}^n$ by the Gram-Schmidt orthogonalisation process:*

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^*,$$

where

$$\mu_{ij} = \langle b_i, b_j^* \rangle / \langle b_j^*, b_j^* \rangle.$$

The original basis is said to be reduced *if the following conditions hold.*

1. $|\mu_{ij}| \leq 1/2$ *for* $1 \leq j < i \leq n$; *and*
2. $|b_l^* + \mu_{l,l-1} b_{l-1}^*|^2 \geq \frac{3}{4} |b_{l-1}^*|^2$ *for* $1 < l \leq n$.

Note that because b_{l-1}^* and b_l^* are orthogonal the second condition can be written

$$|b_l^*|^2 \geq (3/4 - \mu_{l,l-1}^2) |b_{l-1}^*|^2, \tag{3.1}$$

a form which will be used in the basis reduction algorithm. The factor 3/4 may be replaced by any number $y \in (\frac{1}{4}, 1)$, but then the factor 2^{i-j} appearing in (3.2) and Theorem 3.2.1 is replaced by the same power of $4/(4y-1)$. The choice of $y = 3/4$ simplifies the results slightly.

If the the first condition holds the basis is said to be *reduced in size*: informally the vectors $b_1, b_2, \ldots, b_n$ should not be too far from orthogonality. The second condition says that the projection of b_l on the orthogonal complement of $b_1, \ldots, b_{l-2}$ must not be too much smaller than the projection of b_{l-1} on the same subspace. Conditions 1 and 2 together can be shown to imply that

$$|b_j^*|^2 \leq 2^{i-j} |b_i^*|^2 \text{ for } 1 \leq j \leq i \leq n, \tag{3.2}$$

that is, the vectors of the orthogonal basis must not become small too quickly. A basis satisfying (3.2) is said to be 2-*reduced*. Inequality (3.2) also shows that the property of being reduced depends on the the order in which the basis vectors are labelled. An elementary example illustrates this.

Example 3.2.3 Let L have basis $b_1 = (2, 0)$ and $b_2 = (0, 1)$. Then we have that $b_1^* = b_1$, $b_2^* = b_2$ and $\mu_{21} = 0$. It is readily checked that inequality (3.2) is not

satisfied. An exchange of subscripts however gives the reduced basis $b_1 = b_1^* = (0,1)$ and $b_2 = b_2^* = (2,0)$.

The following theorem is used in the proof of Theorem 3.3.4 in §3.3

Theorem 3.2.1 *Let $L \subset \mathbb{R}^n$ be a lattice with reduced basis $b_1, b_2, \ldots, b_n$ and $x_1, x_2, \ldots, x_t \in L$ be linearly independent. Then we have*

$$|b_j|^2 \leq 2^{n-1} \max\{|x_1|^2, |x_2|^2, \ldots, |x_t|^2\}$$

for $j = 1, 2, \ldots, t$. In particular we have that for any non-zero vector $x \in L$

$$|b_1|^2 \leq 2^{n-1}|x|^2. \tag{3.3}$$

This result is derived using (3.2).

LLL present an algorithm which, given as input a basis $b_1, \ldots, b_n$ of vectors in $\mathbb{Z}^n$ for a lattice L, outputs a reduced basis with vectors also in $\mathbb{Z}^n$. The new basis vectors are integral linear combinations of the old ones. One feature of the reduction algorithm is the exchange of subscripts illustrated in Example 3.2.3. The other principal process is illustrated in the next example.

Example 3.2.4 Let $b_1 = (2,0)$ and $b_2 = (4,1)$ be a basis of L. The Gram-Schmidt process gives $b_1^* = (1,0)$, $b_2^* = (0,1)$ and $\mu_{21} = 4 > 1/2$, so the first condition for reduction is violated. A new basis can be obtained by replacing b_2 by $b_2 - rb_1$ where r is the integer closest to μ_{21}, namely $r = 2$. Then the basis becomes $b_1 = (2,0)$ and $b_2 = (0,1)$. This is the basis treated in Example 3.2.3.

Consider now the reduction of a basis with a larger number of vectors. If condition 1 of Definition 3.2.3 is not satisfied, that is if $|\mu_{ij}|$ is found to be too large, then b_i is replaced by $b_i - rb_j$ where r is the integer nearest to μ_{ij} and consequent modifications are made to μ_{is} for $1 \leq s \leq j$.

If condition 2 is not satisfied b_{l-1} and b_l are interchanged in the basis ordering and as a result $\mu_{l-1,j}$ and $\mu_{l,j}$ are interchanged for $1 \leq j \leq l-2$; $\mu_{l-1,l}$ is redefined; and $\mu_{i,l-1}$ and $\mu_{i,l}$ are modified for $l+1 \leq i \leq n$. These changes may result in

condition 2 of the definition no longer holding for b_{l-1} and b_{l-2}; and may also result in some values $\mu_{l-1,j}$ being too large in positions which have already been checked. This forces the algorithm to retrace its steps. The overall strategy is to increase the index l, but if an exchange of vectors is made when l is greater than 2 then it is reduced by 1.

It is not obvious that the algorithm will terminate. The proof of termination given by LLL depends on showing that there is a function of the basis vectors which has a lower bound $d(L)$ and that the value of this function is reduced by a factor of at least 3/4 each time two basis vectors have their order reversed. This limits the number of exchanges possible.

A more efficient basis reduction algorithm has been given by Schönhage [53], but as this does not affect the essential complexity result in LLL we do not describe it.

Finally we remark that there are several types of basis reduction not described here. An account of these and further references may be found in [53], [54] and [55].

3.2.2 The complexity of the basis reduction algorithm

The complexity of the algorithm is determined in LLL for the case when the basis vectors are elements of $\mathbb{Z}^n$. If the input vectors satisfy

$$|b_i|^2 \leq B, \qquad \text{for } 1 \leq i \leq n,$$

then the number of iterations is bounded by a function of n and B. LLL show that the number of arithmetic operations required is $O(n^4 \log B)$ and that the integer operands have binary length bounded by $O(n \log B)$. With classical algorithms this gives a total of $O(n^6 (\log B)^3)$ bit operations or, with fast multiplication techniques, $O(n^{5+\epsilon} (\log B)^{2+\epsilon})$ for each $\epsilon > 0$.

3.2.3 Reduction Algorithm

The algorithm is described with the aid of three procedures. The first is $mu(i,j)$ which describes the action taken when μ_{ij} is too large. The function *round* appearing in *mu* is the nearest integer function $\langle \cdot \rangle$ described formally by

$$\langle r \rangle = \lfloor r + 0.5 \rfloor.$$

```
procedure mu(i,j);
  begin
    r := round(μ_ij);
    b_i := b_i − r * b_j;
    for 1 ≤ s ≤ j do μ_is := μ_is − r * μ_js;
    μ_ij := μ_ij − r;
  end; {of mu}
```

The procedure $swap(b_l, b_{l-1})$ describes the action taken when two basis vectors b_l and b_{l-1} are exchanged. The inner products $\langle b_i^*, b_i^* \rangle$ of the vectors b_i^* generated

```
procedure swap(l, l − 1);
  begin
    μ := μ_{l,l−1}; B := B_l + μ^2 * B_{l−1}; μ_{l,l−1} := μ * B_{l−1}/B;
    B_l := B_{l−1} * B_l/B; B_{l−1} := B;
    exchange b_l and b_{l−1};
    for r := 1 to l − 2 do exchange μ_{l−1,r} and μ_{l,r};
    for s := 1 to n do
      begin
        t_{l−1} := μ_{s,l−1}; t_l := μ_{s,l};
        ( μ_{s,l−1} )   ( 1   μ_{l,l−1} ) ( 0    1 ) ( t_{l−1} )
        ( μ_{s,l}   ) := ( 0   1        ) ( 1   −μ ) ( t_l     )
      end
  end; {of swap}
```

```
procedure reduction
begin
  input b_1, ..., b_n;
  determine the {b_i^*} and the {μ_ij} from the Gram-Schmidt procedure;
  for i := 1 to n do B_i := ⟨b_i, b_i⟩;
  l := 2;
  repeat
    if |μ_{l,l-1}| > 1/2 then mu(l, l-1);
    if |b_l^*|^2 < (3/4 - μ_{l,l-1}^2)|b_{l-1}^*|^2 then
      begin
        swap(l, l-1);
        if k > 2 then l := l-1;
      end
    else
      begin
        for l-2 ≥ s ≥ 1 do mu(l, s);
        l := l+1
      end
  until l = n+1
end {of reduction}
```

Figure 3–1: The basis reduction algorithm

by the Gram-Schmidt process are denoted by B_i. The values of B_i are used by *swap* and altered by it. After swapping two vectors *swap* calculates the resultant changes to the b_i^* and μ_{ij} without re-executing the Gram-Schmidt process.

Finally *reduction* describes the overall process. Observe that the Gram-Schmidt process is carried out only once at the beginning of *reduction*.

3.3 Derivation of the factorisation algorithm

We assume that the given input polynomial $f \in \mathbb{Z}[x]$ is both primitive—hence monic—and squarefree. For the rest of this section p is a prime and k is a positive integer. The reader should note that we have denoted the factor of f which is being sought by h and its approximation mod p^k by h^*. In LLL these are denoted by h_0 and h.

With f we associate for $k = 1, 2, 3, \ldots$ and some prime p the element $(f \bmod p^k) \in \mathbb{Z}_{p^k}[x]$ by the rule

$$(f \bmod p^k) = \sum_{r=0}^{n} (a_r \bmod p^k) x^r.$$

Since f is monic $(f \bmod p)$ will have the same degree as f. If we choose p to be the smallest prime which does not divide $R(f, f')$, the resultant of f and its derivative, then $(f \bmod p)$ will also be squarefree.

Now $(f \bmod p)$ is factorised using Berlekamp's algorithm. If $(f \bmod p)$ is irreducible then f is necessarily irreducible and the algorithm terminates. Otherwise we choose an irreducible factor which may be assumed monic and which will not be a repeated factor since $(f \bmod p)$ is squarefree. This factor is raised by Hensel's method to a (monic) factor $\hat{h}$ of $(f \bmod p^k)$ in $\mathbb{Z}_{p^k}[x]$. To each member of $\mathbb{Z}_{p^k}[x]$ there corresponds a "natural" element $h^* = \sum h_r^* x^r$ of $\mathbb{Z}[x]$ such that $0 \leq h_r^* \leq p^k - 1$. These remarks furnish the proof of the following lemma.

Lemma 3.3.1 *Let a monic, squarefree and primitive $f \in \mathbb{Z}[x]$ be given and let p be a prime which does not divide $R(f, f')$. Then we can determine $h^* \in \mathbb{Z}[x]$ such that*

1. *h^* is monic;*
2. *$(h^* \bmod p^k)$ divides $(f \bmod p^k)$ in $\mathbb{Z}_{p^k}[x]$;*
3. *$(h^* \bmod p)$ is irreducible in $\mathbb{Z}_p[x]$;*
4. *$(h^* \bmod p)^2$ does not divide $(f \bmod p)$ in $\mathbb{Z}_p[x]$.*

The degree of h^* is denoted by l and satisfies $0 < l \leq n$.

Arguments about divisibility now prove the following theorem.

Theorem 3.3.1 *Let f be given and h^* determined as described in Lemma 3.3.1. Then f has an irreducible factor h in $\mathbb{Z}[x]$ such that $(h^* \bmod p)$ divides $(h \bmod p)$ in $\mathbb{Z}_p[x]$. Further if $g \in \mathbb{Z}[x]$ divides f then the following three assertions are equivalent:*

- *$(h^* \bmod p)$ divides $(g \bmod p)$ in $\mathbb{Z}_p[x]$*
- *$(h^* \bmod p^k)$ divides $(g \bmod p^k)$ in $\mathbb{Z}_{p^k}[x]$*
- *h divides g in $\mathbb{Z}[x]$*

In particular $(h^* \bmod p^k)$ divides $(h \bmod p^k)$ in $\mathbb{Z}_{p^k}[x]$.

Condition 4. of Lemma 3.3.1 shows that the factor h in the theorem must be unique. Note that since we may have $\deg(h) = \deg(h^*) = n$ this theorem does not assert that f has a factor other than itself.

Theorem 3.3.1 tells us that we may restrict our attention to those polynomials h such that $(h^* \bmod p^k)$ divides $(h \bmod p^k)$ in $\mathbb{Z}_{p^k}[x]$. If we could establish an upper bound $m < n$ for the degree of h we should know that f had a factor. The rest of this section is concerned with establishing the bound m.

As was remarked in section 2.5.1, we can associate with a polynomial $f = a_0 + a_1 x + \cdots + a_n x^n$ an $(n+1)$-vector $(a_0, a_1, \ldots, a_n)$, and vice versa. A set of polynomials may be thought of as spanning a vector space. We shall be interested in the lattice generated by a basis of this space.

Let m be a fixed integer and define L to be the lattice whose basis is

$$\{p^k x^i \mid 0 \leq i < l\} \cup \{h^* x^j \mid 0 \leq j \leq m - l\}. \tag{3.4}$$

L consists of those polynomials which have degree at most m and which, when taken modulo p^k, are divisible by $(h^* \bmod p^k)$. If it can be established that there is an element $b \in L$ such that h divides b then m will be an upper bound for the degree of h. Before we show a condition for this to happen we give a rather technical lemma.

Lemma 3.3.2 *Let* $b \in L$; $\gcd(f,b) = g$; $r = \lambda f + \mu b$ *for some* $\lambda, \mu \in \mathbb{Z}[x]$; $\deg(g) = e$; *and* $\deg(r) < e + l$. *If* $(h^{\alpha} \bmod p)$ *does not divide* $(g \bmod p)$ *then* $r \equiv 0 \pmod{p^k}$.

The proof is by divisibility arguments and is given at the end of Proposition 2.7 of [40].

Let b and g be as in the Lemma and let $\deg(b) = m_1$. If h divides b then it divides anything of the form

$$\lambda f + \mu b \text{ with } \lambda, \mu \in \mathbb{Z}[x].$$

The $(e-1)^{th}$ subresultant of f and b is zero, so this suggests that we look at linear combinations of

$$M := \{x^i f \mid 0 \le i \le m_1 - e - 1\} \cup \{x^j b \mid 0 \le j \le n - e - 1\}. \tag{3.5}$$

We also have

$$\lambda f + \mu b \equiv 0 \pmod{g}$$

so that the coefficients of f and b are linearly dependent. We confine our attention to the set M' of projections of M onto

$$\mathbb{Z}x^e + \mathbb{Z}x^{e+1} + \ldots + \mathbb{Z}x^{n+m_1-e-1}. \tag{3.6}$$

LLL show that M' is a lattice and the projections of the elements of (3.5) onto (3.6) are a basis for it. An upper and lower bound for the determinant of M' are now used to obtain a condition that an element of L contains a divisor of f.

Hadamard's inequality gives

$$d(M') \le |f|^{m_1-e}|b|^{n-e} \le |f|^m|b|^n, \tag{3.7}$$

which is the upper bound. The lower bound is harder to find: the proof is indicated below.

Theorem 3.3.1 shows that proving $(h^{\alpha} \bmod p)$ divides $(g \bmod p)$ is equivalent to proving that h divides b, so let us assume that $(h^{\alpha} \bmod p)$ does not divide $(g \bmod p)$

and see what would lead to a contradiction. The lattice M' must have a basis $B = \{b_e, b_{e+1}, \ldots, b_{n+m_1-e-1}\}$ such that $\deg(b_j) = j$ (see [12]) and our assumption lets us invoke Lemma 3.3.2 to see that p^k divides $\mathrm{lc}(b_j)$ for $e \leq j \leq e+l-1$. So a lower bound for the determinant of M' is

$$d(M') \geq p^{kl}, \tag{3.8}$$

provided

$$e + l - 1 \leq n + m_1 - e - 1. \tag{3.9}$$

The result (3.8) follows from the facts that g divides b and $(h^* \bmod p)$ divides $(f/g \bmod p)$. Hence (3.7) and (3.8) give the required contradiction if

$$p^{kl} > |f|^m |b|^n. \tag{3.10}$$

We sum up this discussion in a theorem which gives a condition for h to be a proper factor.

Theorem 3.3.2 *With the notation given, if $b \in L$ satisfies $|b|^n < p^{kl}/|f|^m$, then h has degree at most m and divides f in $\mathbb{Z}[x]$. In particular h divides b and* $\gcd(f, b) \neq 1$.

If b is sufficiently small then it is divisible by h, so we want to look at small vectors in L. It is now that the reduced basis proves useful. To find a bound for m the only vector of L we need examine is the first element of its reduced basis.

To prove the next theorem we need a bound on the size of the factors of a polynomial, and this can be deduced from the Landau-Mignotte Theorem [44] which we quote.

Theorem 3.3.3 *Let $f \in \mathbb{Z}[x]$ have the form*

$$f = \sum_{i=0}^{n} a_i x^i$$

and let $g \in \mathbb{Z}[x]$ be a factor of f of the form

$$g = \sum_{i=0}^{m} b_i x^i.$$

Then in the case when f and g are monic we have

$$\sum_{i=0}^{m} |b_i| \leq 2^m \sqrt{\sum_{i=0}^{n} a_i^2}. \tag{3.11}$$

We call (3.11) the Landau-Mignotte inequality. We denote the right hand side by Λ and call it the Landau-Mignotte factor-bound of f. From (3.11) we can deduce [32, Example 4.6.2.20] that if $g \in \mathbb{Z}[x]$ of degree m divides $f \in \mathbb{Z}[x]$ of degree n then

$$|g|^2 \leq \binom{2m}{m} |f|^2. \tag{3.12}$$

Now 3.12 and Theorem 3.2.1 lead to a result which allows us to find the bound m on the degree of h.

Theorem 3.3.4 *With the same notation as above suppose that $b_1, b_2, \ldots, b_{m+1}$ is a reduced basis for L and that*

$$p^{kl} > 2^{mn/2} \binom{2m}{m}^{n/2} |f|^{m+n}. \tag{3.13}$$

Then $\deg(h) \leq m$ *if and only if*

$$|b_1| < (p^{kl}/|f|^m)^{1/n}. \tag{3.14}$$

The last result shows how to find h itself. It turns out that with the reduced basis Theorem 3.2.1 implies that if b_j satisfies

$$|b_j| < (p^{kl}/|f|^m)^{1/n} \tag{3.15}$$

then so do $b_1, \ldots, b_{j-1}$; and h is a factor of just those b_j which satisfy (3.15). The exact result is the last theorem of this section.

Theorem 3.3.5 *Let the notation and hypothesis be the same as for Theorem 3.3.4. In addition let there be an integer $j \in \{1, 2, \ldots, m+1\}$ for which*

$$|b_j| < (p^{kl}/|f|^m)^{1/n}, \tag{3.16}$$

and let t be the largest such j. Then

$$\begin{aligned} \deg(h) &= m + 1 - t, \\ h &= \gcd(b_1, b_2, \ldots, b_t), \end{aligned}$$

and (3.16) holds for all j such that $1 \leq j \leq t$.

This shows that once we have obtained a bound m on the degree of h the reduced basis enables us to find h by a gcd calculation. This justifies the remark made earlier that the search is trivial. In fact no search is needed.

3.4 The Algorithms

3.4.1 Introduction

The factorisation algorithm has been divided into two procedures called *outer* and *inner*. The outer procedure finds an h^* such that the preconditions for Theorem 3.3.1 are satisfied, while the inner procedure looks for a factor h of f. The correct value of the parameter m is not known and this must be determined. There is some freedom in the overall structure of the algorithm: we follow LLL.

The first thing is to determine a suitable prime p, that is, one which does not divide $R(f, f')$, the resultant of f and f'. This can be done by generating the sequence of primes starting at 2 and testing each to see if it is a divisor of $R(f, f')$. If it is not we can stop, while if r is a prime divisor of t then $t = r^e t_1$ so that later division tests can be done on t_1. Each failure reduces the dividend by at least one half, so the number of tests needed is $O(\log |R(f, f')|)$. Since the resultant has $O((2n)!)$ terms whose size is bounded by $O(n^n |f|^{2n})$ we need at most $O(n \log n + n \log |f|)$ trials.

Next we look for a factor of $(f \bmod p)$ in $\mathbb{Z}_p[x]$ by Berlekamp's algorithm. If $(f \bmod p)$ is irreducible then we can stop; otherwise the factor is raised by Hensel lifting to a factor of $(f \bmod p^k)$ in $\mathbb{Z}_{p^k}[x]$. The index k is chosen so that (3.13) is satisfied for the maximum possible value of m, namely $(n-1)$. These things are done by the procedure *outer*.

The procedure *inner* finds, for the current value of m, a reduced basis for the lattice L. The reduced basis vectors are then used in the test of Theorem 3.3.4 to see whether L contains a factor. If so it is calculated by the result of Theorem

```
procedure outer;
begin
  determine a prime p such that R(f, f') ≢ 0  (mod p);
  find h* ∈ Z[x] such that (h* mod p) divides (f mod p) in Z_p[x];
  l := deg(h*);
  if l < n then
    begin h = f; exit end
  else
    begin
      k := 1;
      while p^{kl} ≤ 2^{n(n-1)/2} C(2n-2, n-1)^{n/2} |f|^{2n-1} do
        begin
          k := k + 1;
          use Hensel lifting to modify h* so that (h* mod p^k)
            divides (f mod p^k) in Z_{p^k}[x]
        end
    end
end
```

```
procedure inner;
begin
  find the basis (3.4) of the lattice L of Theorem 3.3.2;
  determine a reduced basis b_0, ..., b_m for L;
  if |b_1| < (p^{kl}/|f|^m)^{1/n} then
    begin
      t := 1;
      while |b_t| < (p^{kl}/|f|^m)^{1/n} do t := t + 1;
      deg(h) := m + 1 - t;
      h := gcd(b_1, ..., b_{t-1}); {h divides g}
      exit
    end
end
```

3.3.5. This gcd calculation is carried out using the subresultant algorithm of [7] mentioned in section 2.7.2.

The LLL algorithm executes *outer* once and determines a minimum value for m (which must be at least l). Then *outer* attempts to find a factor. If *outer* is succesful the algorithm terminates; if it is not successful *outer* is repeated with m doubled, unless m exceeds $n-1$, when we know that f is irreducible.

It must be possible for m to achieve the value $(n-1)$ so that a factor will not be missed. It is not necessary to start at the lowest possible value, however, as the factor found by the algorithm will always be irreducible. It would be possible to omit the loop and and simply use $m = n-1$.

The overall algorithm is so structured that it terminates as soon as either a factor is found or it is established that f is irreducible.

Here is an example of the algorithm. The working has been simplified by choosing *a priori* a value of m.

Example 3.4.1 Let

$$f = x^4 + x^3 + 2x^2 + x + 1 \tag{3.17}$$

```
begin
  input f; n := deg(f);
  execute outer;
  {calculate the minimum value of m}
  u := 1;
  while (n − 1)/2^u > l do u := u + 1;
  m := ⌊(n − 1)/2^u⌋;
  {look for a factor}
  repeat
    execute inner;
    m := 2m
  until m > n − 1; {if this is true f is irreducible}
  h := f
end
```

Figure 3–2: The LLL algorithm

and choose for a prime number $p = 41$. Then we find that

$$f \equiv (x-9)(x+9)(x^2+x+1) \pmod{41} \tag{3.18}$$

Now if we identify h^{∞} with $(x-9)$ we have $l = 1$ and

$$2^{n(n-1)/2}\binom{2n-2}{n-1}^{n/2}|f|^{2n-1} = 37072760.$$

Since

$$41^5 > 37072760 > 41^4$$

we choose $k = 5$ and raise the factorisation 3.18 to

$$f \equiv (x+46464143)(x^3-46464142x^2-46464142x-46464143) \pmod{41^5}, \tag{3.19}$$

in which h^{∞} has been modified to $(x+46464143)$. The irreducible factor which we seek has degree at most three if f is not irreducible, so let $m = 3$ and choose as a basis for the lattice L

$$41^5, h^{\infty}, h^{\infty}x \text{ and } h^{\infty}x^2.$$

This has as a reduced basis

$$\begin{gathered}(1,0,1,0),\\(-5237,-2476,5238,0),\\(1238,-10475,-1238,0),\\(-1107107,-522435,1103118,1).\end{gathered}$$

Here the vector (a_0,a_1,a_2,a_3,a_4) corresponds to the polynomial $a_0+a_1x+a_2x^2+a_3x^3+a_4x^4$. Only the first element of the reduced basis satisfies the condition that its length is less than

$$(p^{kl}/|f|^m)^{1/n} = (41^5/16\sqrt{2})^{1/4} \simeq 59.44$$

so that $t = 2$, $\deg(h) = 2$ and $h = b-1 = x^2+1$. Hence

$$f = (x^2+1)(x^2+x+1).$$

3.5 Complexity

The complexity of the LLL algorithm depends on the degree n of f and the size of the coefficients. The coefficient size is described by $|f|$ which is defined for $f = \sum_{i=0}^{n} a_i x^i$ by

$$|f| = \left(\sum_{i=0}^{n} a_i^2 \right)^{1/2}.$$

The algorithm requires $O(n^6 + n^5 \log |f|)$ arithmetic operations on integers of binary length $O(n^3 + n^2 \log |f|)$. We can sum up this chapter in a theorem.

Theorem 3.5.1 *A polynomial $f \in \mathbb{Z}[x]$ can be completely factored into the product of an integer and irreducible elements of $\mathbb{Z}[x]$ in a time $O(n^{12} + n^9(\log |f|)^3)$, or, if fast multiplication methods are used, $O(n^{9+\epsilon} + n^{7+\epsilon}(\log |f|)^{2+\epsilon})$ for each $\epsilon > 0$.*

Chapter 4

Kaltofen's Reduction

4.1 Introduction

The general theme of this chapter is that factorisation of a polynomial f in $v+1$ indeterminates can be reduced in polynomial time to the problem of factorising a polynomial in one indeterminate. For simplicity we shall let $v = 1$ and $f \in \mathbb{Z}[y, x]$. The algorithm described by Kaltofen in [26] seeks a factor of f by first finding a factor of $f(0,x)$ using the LLL algorithm and then using Hensel lifting modulo powers of y.

We discussed at the end of section 2.3 how to change an arbitrary given bivariate polynomial into one which is monic in a specified indeterminate. If f is not squarefree, we can determine from it a squarefree polynomial with the same factors by gcd calculations. Finally if $f(0,x)$ is not squarefree then a simple change to the indeterminate y will make it so. To be precise, we can find, in a small number of trials, $b \in \mathbb{Z}$ such that if $\bar{y} = y + b$ and $\bar{f}(\bar{y}, x) = f(\bar{y} - b, x)$ then $\bar{f}(0,x)$ is squarefree. We therefore assume that

1. $f(y,x)$ is monic in x and hence primitive;

2. $f(y,x)$ is squarefree; and

3. $f(0,x)$ is squarefree.

If $f(y,x)$ is primitive, monic in x and factorises in $\mathbb{Z}[y,x]$ then we may have either $f(y,x) = g(y,x)h(y,x)$ or $f(y,x) = g(y,x)h(x)$, where $g(y,x)$, $h(y,x)$ are

monic in x, but not $f(y,x) = g(y,x)h(y)$. It follows that $f(0,x)$ must factorise in $\mathbb{Z}[x]$. The converse is false — $x^2 + y - 1$ is irreducible — but we might hope to use information about factors of $f(0,x)$ to search for factors of $f(y,x)$. We first describe the general idea.

Suppose to begin with that we have determined a linear factor $x - a_0$ of $f(0,x)$. This will not in general be a factor of $f(y,x)$, but we can look for a possible factor of the form $x - \alpha_k$ where

$$\alpha_k = a_0 + a_1 y + \cdots + a_k y^k, \tag{4.1}$$

and the a_i are to be determined. Kaltofen's method shows how a sequence of α_k may be constructed using Hensel lifting.

If $x - a_0$ is a linear factor of $f(0,x)$ then a_0 is called a *root* of $f(0,x)$. The factors of $f(0,x)$ in $\mathbb{Q}[x]$ need not be linear and in general the roots of $f(0,x)$ must be sought in an extension field of $\mathbb{Q}$. To be more precise, if $t(x)$ is an irreducible factor of $f(0,x)$ then a root is any $\beta \in \mathbb{Q}[x]/(t)$ such that $t(\beta) = 0$. The a_i in (4.1) are elements of $\mathbb{Q}[x]/(t)$ and the calculations for the Hensel lifting are carried out in this field. We shall follow standard practice and denote $\mathbb{Q}[x]/(t)$ by $\mathbb{Q}(\beta)$.

In order to find a factor of $f(y,x)$ in $\mathbb{Q}[y,x]$ we need to find the minimum polynomial in $\mathbb{Q}[y][x]$ of the form

$$x^M + u_{M-1}(y)x^{M-1} + \cdots + u_1(y)x + u_0(y)$$

satisfied by $x = \alpha_k(y)$. If it can be found this polynomial will be the desired factor.

Some changes in notation have been made from the original paper [26]. The factor of f which we are looking for is called h rather that g, and the polynomials which are called h_r by Kaltofen we call l_r.

4.2 Approximating a root

Factors of $f \in \mathbb{Z}[y,x]$ are to be sought in $\mathbb{Q}[y,x]$. We let $\deg_x(f) = n$, $\deg_y(f) = d$ and define polynomials $f_r \in \mathbb{Q}[x]$ by

$$f(y,x) = \sum_{r=0}^{d} f_r(x) y^r, \tag{4.2}$$

so that $f_0(x) = f(0,x)$.

We shall attempt first of all to write the input polynomial in the form

$$f(x,y) = \left(x - \sum_{r=0}^{k} a_r y^r\right) \sum_{r=0}^{d-k} l_r(x) y^r = \sum_{r=0}^{d} f_r(x) y^r \tag{4.3}$$

The value of k is so far undetermined, the a_r lie in an extension field $\mathbb{Q}(\beta)$ to be determined below, and the $l_r(x)$ are in $\mathbb{Q}(\beta)[x]$.

Putting $y = 0$ in (4.3) gives

$$(x - a_0) l_0(x) = f_0(x) = f(0,x) \in \mathbb{Q}[x], \tag{4.4}$$

so that the first requirement is to factor a univariate polynomial. This can be done using the LLL algorithm. If f_0 is irreducible then so is f and we can stop. As indicated in the previous section there is no guarantee that f_0 has a root in $\mathbb{Q}$ (i.e. has a linear factor) and so we suppose that the LLL algorithm outputs $t \in \mathbb{Q}[x]$ such that

- t divides f_0 in $\mathbb{Q}[x]$;
- t is irreducible in $\mathbb{Q}[x]$; and
- $\deg(t) = m > 0$.

We denote by β an element of $\mathbb{Q}[x]/(t)$ such that $t(\beta) = 0$ and write $\mathbb{Q}(\beta)$ for $\mathbb{Q}[x]/(t)$.

Looking back at equation (4.4) we see that $a_0 = \beta$ and $l_0(x) = f_0(x)/(x - \beta)$, the division being exact by the choice of β. Because f_0 is squarefree, $l_0(\beta) \neq 0$.

Picking out the coefficients of y^r on both sides of (4.3) gives

$$(x-\beta)l_r(x) - a_r l_0(x) = f_r(x) + \sum_{s=1}^{r-1} a_s l_{r-s}(x). \tag{4.5}$$

Provided $a_0, \ldots, a_{r-1}$ and $l_0, \ldots, l_{r-1}$ have already been calculated then a_r is found by evaluating (4.5) at $x = \beta$:

$$a_r = -\frac{1}{l_0(\beta)} \left(f_r(\beta) + \sum_{s=1}^{r-1} a_s l_{r-s}(\beta) \right), \tag{4.6}$$

and $l_r(x)$ is then given by

$$l_r(x) = \frac{1}{(x-\beta)} \left(a_r l_0(x) + f_r(x) + \sum_{s=1}^{r-1} a_s l_{r-s}(x) \right), \tag{4.7}$$

the division being exact by the choice of a_r.

We now define a k-th order approximate root of $f(y,x)$ to be

$$\alpha_k(y) = \sum_{r=0}^{k} a_r y^r. \tag{4.8}$$

Because (4.5) has been satisfied for $r = 0, \ldots, k$ it follows that

$$f(y, \alpha_k(y)) \equiv 0 \pmod{y^{k+1}} \quad \text{for} \quad k = 0, 1, 2, \ldots . \tag{4.9}$$

If equations (4.6) and (4.7) are regarded as assignments they carry out the Hensel lifting of the approximations to α modulo powers of y.

It will be shown later that either this process finds a root of $f(y,x)$ or it may be stopped after a finite number of steps with the assertion that f is irreducible in $\mathbb{Q}[y,x]$. In fact the maximum value of k needed is $\lceil (2n-1)d/m \rceil$.

4.3 Constructing a minimal polynomial

The process of the previous section attempts to construct a factor $(x - \alpha(y)) \in \mathbb{Q}(\beta)[y, x]$. We are actually interested in a factor of $f(x, y)$ in $\mathbb{Q}[y, x]$ so the next step is to determine if possible the minimal polynomial in $\mathbb{Q}[y][x]$ satisfied by $\alpha(y)$. Call this (monic) polynomial H and its degree M. Then $M \geq m$ and, if H is to lead to a factor, $M < n$. Put

$$H(\alpha_k(y)) = \alpha_k(y)^M + \sum_{r=0}^{M-1} u_r(y)\alpha_k(y)^r, \tag{4.10}$$

where the $u_r \in \mathbb{Q}[y]$ have the form

$$u_r(y) = \sum_{i=0}^{d} u_{r,i} y^i, \tag{4.11}$$

since $\deg(u_r) \leq \deg_y(f) = d$.

We know neither M nor the u_r at this stage. To see how we can find the u_r and test whether some choice of M and $\alpha_k(y)$ is satisfactory we quote the key result of [26], in which L is the smallest value of the running index k which leads to a factor. The proof will be indicated in section 4.5.

Theorem 4.3.1 *If, in the notation given, there is an integer M, $m \leq M < n$, and an integer L such that*

$$H(\alpha_L(y)) \equiv 0 \pmod{y^{L+1}}. \tag{4.12}$$

can be satisfied for some choice of $u_r(y)$ then the u_r are unique and

$$h(y, x) = x^M + \sum_{r=0}^{M-1} u_r(y)x^r \tag{4.13}$$

is an irreducible factor of f in $\mathbb{Q}[y, x]$. Furthermore $L \leq \lceil (n + M)d/m \rceil$.

In short, if the minimal polynomial of $\alpha(y)$ has degree less than n it is an irreducible factor of $f(y, x)$. Since M will not exceed $n - 1$ a global bound for L is given by $\lceil (2n - 1)d/m \rceil$, the result quoted at the end of the previous section.

To test whether some choice of α_L and M is satisfactory, we proceed as follows. Each coefficient a_i in $\alpha_L = \sum_{i=0}^{L} a_i y^i$ is a polynomial of degree at most $m-1$ in β so

$$\begin{aligned}(\alpha_L(y))^r &= \left(\sum_{i=0}^{L}\left(\sum_{j=0}^{m-1} a_{ij}\beta^j\right) y^i\right)^r \\ &\equiv \sum_{i=0}^{L}\left(\sum_{j=0}^{m-1} a_{i,j}^{(r)}\beta^j\right) y^i \pmod{y^{L+1}}. \end{aligned} \tag{4.14}$$

The $a_{ij}^{(r)}$ in (4.14) are calculated using the fact that $t(\beta) = 0$. Substituting these expressions for α_L and u_r into (4.12) and equating the coefficient of $\beta^j y^i$ to zero we obtain

$$a_{i,j}^{(M)} + \sum_{r=0}^{M-1}\sum_{s=0}^{d} a_{i-s,j}^{(r)} u_{r,s} = 0, \text{ for } 0 \le i \le L \text{ and } 0 \le j \le m-1, \tag{4.15}$$

a system of $m \times (L+1)$ equations for the $(d+1)M$ unknowns $u_{r,s}$ with $0 \le r \le M-1$ and $0 \le s \le d$. Kaltofen [26] shows that if (4.15) have a solution then it is unique and the $u_{r,s}$ are integers, so that the factor found is in $\mathbb{Z}[y,x]$ even though intermediate calculations have been carried out in $\mathbb{Q}(\beta)[x]$.

If (4.15) have a solution then we are done, so the test is carried out starting at $M = m$ and increasing the value of M from m either until a factor is found or until $M = n$. The algorithm is given schematically in the next section.

4.4 The Algorithm

The input polynomial $f(y,x)$ has the properties described in Section 4.1. Either a factor is found and is output in h or f is irreducible and a suitable output message is generated.

begin

input $f(y,x)$; $n := \deg_x(f)$; $d := \deg_y(f)$;

{find the starting approximation using the LLL algorithm}

find an irreducible factor $t(x)$ of $f(0,x)$; $m := \deg(t)$;

let β satisfy $t(\beta) = 0$;

{raise the approximations to α by Hensel's method}

$K := \lceil (2n-1)d/m \rceil$;

define $f_r(x)$ by equation (4.2);

$a_0 := \beta$; $h_0(x) := f_0(x)/(x-\beta)$;

for $r := 1$ to $K-1$ do

begin

assign a_r according to equation (4.6);

assign $\ell_r(x)$ according to equation (4.7)

end;

assign a_K according to equation (4.6);

for $k := 0$ to K do $\alpha_k(y) := \sum_{r=o}^{k} a_r y^r$;

{look for a factor}

for $M := m$ to $n-1$ do

begin

$L := \lceil d(N+M)/m \rceil$;

for $r := 0$ to $n-1$ do $\alpha_L^{(r)}(y) := [\alpha_L(y)]^r \pmod{y^{L+1}}$;

try to find $u_r(y)$, $0 \le r \le M-1$ to solve

$$\alpha_L^{(M)} + \sum_{r=0}^{M-1} u_r \alpha_L^{(r)} \equiv 0 \pmod{y^{L+1}};$$

if a solution is found

then

begin

$h(y,x) := x^M + \sum_{r=0}^{M-1} u_r(y) x^r$;

exit

end

end

{if we reach here f is irreducible}

end

4.5 Correctness of the Algorithm

We are interested in the polynomial $h(y,x)$ constructed by the algorithm of the previous section so we assume here that, for some $L \in \mathbb{Z}^+$,

1. $f(0,x)$ is squarefree;
2. $f(y,x)$ is monic in x;
3. the minimal polynomial in $\mathbb{Q}[x]$ of the root β of $f(0,x)$ has degree m;
4. α_k satisfies $f(y,\alpha_k(y)) \equiv 0 \pmod{y^{k+1}}$ for $0 \leq k \leq L$; and
5. $h(y,x)$ is a polynomial of minimum degree in x such that

$$h(y,\alpha_L(y)) \equiv 0 \pmod{y^{L+1}}.$$

We show first that if $g(y,x)$ is any factor of f and $(0,\beta)$ is a root of g, then $g(y,\alpha_k(y))$ is divisible by y^{k+1}.

Lemma 4.5.1 *Let $g(y,x)$ divide $f(y,x)$ in $\mathbb{Q}[y,x]$ and satisfy $g(0,\beta) = 0$. Then*

$$g(y,\alpha_k(y)) \equiv 0 \pmod{y^{k+1}} \text{ for } 0 \leq k \leq L.$$

Proof Since $g(0,\beta) = 0$ then y divides $g(y,\beta) = g(y,\alpha_0(y))$. It follows that

$$g(y,\alpha_0(y)) \equiv 0 \pmod{y}.$$

Now suppose that

$$g(y,\alpha_{r-1}(y)) \equiv 0 \pmod{y^r}$$

for some $r < L$. This means that we may write

$$g(y,\alpha_{r-1}(y)) \equiv \gamma y^r \pmod{y^{r+1}}.$$

If $f(y,x) = g(y,x)\bar{g}(y,x)$ then

$$f(y,\alpha_r(y)) = \gamma y^r \bar{g}(0,\beta) \underset{\neq 0}{} \pmod{y^{r+1}}$$

by assumption 4. Since $f(0,x)$ is squarefree, $\bar{g}(0,\beta) \neq 0$ and hence $\gamma \equiv 0 \pmod{y}$ □

Lemma 4.5.1 and condition 5 immediately give

Lemma 4.5.2 *If* $c \in \mathbb{Q}[y,x]$ *and* $\deg_x(c) = j < \deg_x(h)$ *then* $c(0,\beta) \neq 0$.

We next show how to characterise low degree polynomials which satisfy a condition of the type 4. Suppose that $q \in \mathbb{Q}[y,x]$, $\deg_x(q) < m$ and $q(y,\alpha_k(y)) \equiv 0 \pmod{y^{k+1}}$ for $0 \leq k \leq L$. In the case $k = 0$ this gives $q(y,\beta) \equiv 0 \pmod{y}$ so that $q(0,\beta) = 0$ and hence, by condition 3, $q(0,x) = 0$. Thus y divides q. One can now argue in a similar way for $k \leq L$ that if y^k divides q and $q(y,\alpha_k(y)) \equiv 0 \pmod{y^{k+1}}$ then y^{k+1} divides q. This proves

Lemma 4.5.3 *Let* $q \in \mathbb{Q}[y,x]$ *satisfy* $\deg_x(q) < m$ *and*

$$q(y,\alpha_k(y)) \equiv 0 \pmod{y^{k+1}} \text{ for } 0 \leq k \leq L.$$

Then y^{L+1} *divides* q *and in particular divides* $\mathrm{lc}_x(q)$

We have not yet shown that the polynomial $h(y,x)$ constructed by the algorithm is a factor of f. Before we do this we require a final lemma and we now particularise g to be $\gcd(f,h)$.

Lemma 4.5.4 *Let* $g = \gcd(f,h)$ *and* $\deg_x(g) = j < \deg_x(h)$. *Then* $n \geq j + m$.

Proof Write $f(y,x) = g(y,x)\bar{g}(y,x)$. It is immediate that

$$\deg_x(f) = n = \deg_x(g) + \deg_x(\bar{g}) = j + \deg_x(\bar{g}).$$

Put $(y,x) = (0,\beta)$. Then $f(0,\beta) = 0$ and $g(0,\beta) \neq 0$ by Lemma 4.5.2. Hence $\bar{g}(0,\beta) = 0$ and $\deg_x(\bar{g}) \geq m$. □

We now come to the main theorem which proves that h divides f. The proof uses facts on subresultants described in section 2.7.2.

Theorem 4.5.1 *Let h be the polynomial determined by the algorithm of the previous section. Then $\deg_x(\gcd(f,h)) = \deg_x(h)$ so that h divides f.*

Proof Let $g = \gcd(f,h)$, $j = \deg_x(g)$ and $I = \deg_x(h)$. We shall assume that $j < I$ and derive a contradiction. If f and h are regarded as polynomials in x with coefficients in $\mathbb{Q}[y]$ then their j-th subresultant is

$$S_j(f,h) = \sum_{r=0}^{j} s_{j,j-r}(f,h)x^r,$$

where the $s_{j,j-r}$ are in $\mathbb{Q}[y]$. The contradiction will be obtained by finding lower and upper bounds for $\deg(s_{j,0})$.

Lower Bound

Column operations on $t'_{j,0}(f,h)$ give rise to relations of the form—see (2.26)—

$$c_i(y)(x) = \lambda_i(x)f(x) + \mu_i(x)h(x) \quad \text{for } j \le i \le n+I-j-1, \tag{4.16}$$

where $\deg_x(c_i) = i$ and the polynomials in x have coefficients in $\mathbb{Q}[y]$. By the properties of $\alpha_k(y)$ and Lemma 4.5.1 we have

$$f(y,\alpha_k(y)) \equiv h(y,\alpha_k(y)) \equiv 0 \pmod{y^{k+1}} \quad \text{for } 0 \le k \le L.$$

Hence for $j \le i \le n+I-j-1$ we have

$$c_i(y,\alpha_k(y)) \equiv 0 \pmod{y^{k+1}} \quad \text{for } 0 \le k \le L. \tag{4.17}$$

Since g must divide c_i we also have

$$c_i(y,x) = g(y,x)\bar{g}_i(y,x),$$

where $\deg_x(\bar{g}_i) = i-j$. Lemma 4.5.2 tells us that $g(y,\alpha_k(y)) \not\equiv 0 \pmod{y\quad}$ and so from (4.17) and Lemma 4.5.3 it follows that if $i-j < m$ then y^{L+1} divides $\mathrm{lc}_x(\bar{g}_i)$. We deduce that

$$\text{if } i-j < m \text{ then } y^{L+1} \text{ divides } \mathrm{lc}_x(c_i). \tag{4.18}$$

By the assumption $j < I$ and Lemma 4.5.4 we have $m \le n-j$ so that

$$m+j-1 \le n-1 < n+I-j-1$$

and

$$y^{L+1} \text{ divides } \mathrm{lc}_x(c_i) \text{ for } j \le i \le m+j-1. \tag{4.19}$$

Furthermore the first $I-j$ rows of the matrix $t_{j,0}$ are in lower triangular form so that $\mu_i = 0$, $\lambda_i = x^{I-j-1}$ and $\mathrm{lc}_x(c_i) = \mathrm{lc}_x(f) = 1$ for $n+1 \leq i \leq n+I-j-1$. Therefore

$$s_{j,0} = \prod_{i=j}^{n-1} \mathrm{lc}_x(c_i). \tag{4.20}$$

Result (4.20) and equation (4.19) imply that y^{mL} divides $s_{j,0}$ and we obtain the lower bound for $\deg_y(s_{j,0})$:

$$\deg_y(s_{j,0}) \geq mL. \tag{4.21}$$

Upper Bound

Each entry of $t_{j,0}$ is a polynomial of degree at most d in y. Since f is monic in x, $\deg_y(\mathrm{lc}_x(g)) = 0$ and the entries in the first column are elements of $\mathbb{Q}$. It follows that

$$\deg_y(s_{j,0}) \leq (I+n-1)d \quad \text{for} \quad j \geq 0.$$

If L is chosen to be $\lceil (I+n)d/m \rceil$ then we have

$$mL > (I+n-1)d,$$

that is, the lower bound of $\deg_y(s_{j,0})$ exceeds the upper bound. This is the required contradiction.

The maximum degree in x of a factor of f is $n-1$, so the algorithm is guaranteed to find a factor when one exists provided it iterates up to

$$L = K = \lceil (2n-1)d/m \rceil.$$

□

4.6 Extension to Multivariate Polynomials

Kaltofen [25], [27] has also shown how factorisation of

$$f(y_1, \ldots, y_v, x) \in \mathbb{Z}[y_1, \ldots, y_v, x],$$

with v arbitrary but fixed, may be obtained from a factorisation of $f(0, \ldots, 0, x)$. As in the bivariate case $f(y_1, \ldots, y_v, x)$ must be squarefree and monic in x and $f(0, \ldots, 0, x)$ must be also be squarefree. As before $n = \deg_x(f)$ but now d denotes the highest total degree of any monomial in $y_1, \ldots, y_v$ occurring in f.

Assuming $f(0, \ldots, 0, x)$ is not irreducible we denote by $t(x)$ an irreducible factor of $f(0, \ldots, 0, x)$; by β a root of t; and by J the ideal of $\mathbb{Q}(\beta)[y_1, \ldots, y_v, x]$ generated by $\{y_1, \ldots, y_v\}$. We construct a sequence of approximations $\alpha_0, \alpha_1, \ldots$ to a root $x = \alpha$ in the form

$$\alpha_k(y_1, \ldots, y_v) = \sum_{i=0}^{k} \sum_{j=i} a_{j_1 \ldots j_v} y_1^{j_1} \cdots y_v^{j_v}, \quad \text{for } k = 0, 1, 2, \ldots ,$$

where $j = j_1 + \cdots + j_v$ and $a_{j_1 \ldots j_v} \in \mathbb{Q}(\beta)$. Corresponding to equation (4.9) we now require that

$$f(y_1, \ldots, y_v, \alpha_k(y_1, \ldots, y_v)) \equiv 0 \pmod{J^{k+1}}, \tag{4.22}$$

that is, we require that the left hand side of (4.22) has no monomial in $y_1, \ldots, y_v$ of total degree less than $k + 1$.

Once again this leads to a set of linear equations. If these have a solution (which must be integral and unique) then a factor $h(y_1, \ldots, y_v, x) \in \mathbb{Z}[y_1, \ldots, y_v, x]$ is obtained. The maximum value of k required in the multivariate case is $k = L = d(2n - 1)$. If this value of k is reached and no solution of the linear equations has been found then $f(y_1, \ldots, y_v, x)$ is irreducible. The required maximum for k is larger than it was in the bivariate case because there is no result corresponding to (4.20).

4.7 Complexity

In the case of multivarite polynomials we specify that the number of indeterminates is fixed, by which we mean that the complexity is a function only of the total degree of f and of the size of the coefficients which as before are measured by $\log(|f|)$. Provided all the coefficients including the zero coefficients are listed (dense encoding) the algorithms described in sections 4.2, 4.3 and 4.6 require a time which is polynomial in the total degree and $\log(|f|)$. Kaltofen's results therefore extends the result of LLL to polynomials in any number of variables with integer coefficients.

Chapter 5

Direct extensions of the LLL algorithm

5.1 Introduction

The extensions of the LLL algorithm by A. K. Lenstra which are described here are the factorisations of multivariate polynomials with coefficients in $\mathbb{Z}$ [37]; in $\mathbb{Q}(\beta)$, an algebraic extension of $\mathbb{Q}$ [39]; and in the a finite field $\mathbb{Z}_q$, where q is a power of a prime [38]. To keep the presentation simple the descriptions are for bivariate polynomials. A. K. Lenstra has also made an extension to the factorisation of univariate polynomials over $\mathbb{Q}(\beta)$ [36], but this is not covered in the thesis.

The general approach in all cases is the same. First of all we regard f as an element of $R[y][x]$ so that $f = \sum a_i x^i$ for $a_i \in R[y]$, where R is one of $\mathbb{Z}$, $\mathbb{Q}(\beta)$ or $\mathbb{Z}_q$. We assume that f is squarefree and primitive with respect to x, that is, the coefficients in $R[y]$ have greatest common divisor 1. A suitable ideal J of $R[y]$ is identified, corresponding to the ideal (p) of $\mathbb{Z}$ in the LLL algorithm, such that $R[y]/J$ is a finite field. In addition we use a second ideal J' of $R[y]$ which corresponds to (p^k) in the LLL algorithm. We define $(f \bmod J)$ to be $\sum(a_i \bmod J)x^i$ with a similar notation for other ideals. Berlekamp's algorithm can be used to find $\hat{h} \in R[y][x]$ such that $(\hat{h} \bmod J)$ divides $(f \bmod J)$; and Hensel lifting may be used to modify $\hat{h}$ to h^* with the properties that $(h^* \bmod J)$ is a simple factor of $(f \bmod J)$ and that $(h^* \bmod J')$ divides $(f \bmod J')$. At this stage we have a lemma corresponding to Lemma 3.3.1 and this leads to a theorem corresponding to Theorem 3.3.1.

Now we are able to identify a set of polynomials which must contain a factor unless f is irreducible. This set may be regarded as a lattice L, and a basis for L is identified. We need to know if a particular element b of the lattice will divide f, and for each context a condition corresponding to (3.14) of Theorem 3.3.4 is established. This condition depends *inter alia* on the degree of b, and the strategy of the algorithm is to start with a lattice of lowest dimension and look for a factor. If there is none, the lattice is changed to one of higher dimension and the search is continued until either a factor is found or f is shown to be irreducible.

The search is simplified by the use of a basis reduction algorithm. Once a reduced basis has been determined it turns out that if the lattice contains a factor of f then some element of the reduced basis is a factor; and the algorithm is structured in such a way that this factor is irreducible.

For the factorisation of elements of $\mathbb{Z}[y,x]$ and $\mathbb{Q}(\beta)[y,x]$ the desired factor is in some sense a short element in the lattice. When $f \in \mathbb{Z}_q[y,x]$ however there are some distinctive features, essentially due to the fact that the underlying field is finite. In this case the idea of reduction centres on the degree of an element rather than its size: a brief description is given in section 5.4. The factorisation condition is correspondingly different.

5.2 Factorisation in $\mathbb{Z}[y,x]$

We keep the notation of the previous chapter by denoting $\deg_x(f)$ and $\deg_y(f)$ by n and d respectively.

To begin with we need to choose a suitable prime p and a suitable integer s. First s is found such that

- $f(x,s)$ is squarefree; and

- $|s|$ exceeds a number which depends on f, and is given below in (5.6)—(5.8).

The prime p is the smallest prime not dividing $R(f(x,s), f_x(x,s))$, the resultant of $f(x,s)$ and its derivative with repect to x.

Bounds for p and s can be obtained in terms of n, d and the height of f, $\mathrm{ht}(f)$ (see Definition 2.3.2). To be precise, Lenstra shows that $p = O(n^3 d + n^2 d \log(\mathrm{ht}(f)))$ and that the size of s can be bounded by the relation

$$\log(|s|) = O(n^2 + n \log(\mathrm{ht}(f))).$$

It follows that neither p nor s involves exponential growth, nor exponential search time since the integer s can be found in $O(nd)$ trials.

The ideal J of the introduction is generated by p and $y - s$. If we denote by $(p^i, (y-s)^j)$ the ideal generated by p^i and $(y-s)^j$ then J' is $(p^k, (y-s)^{d+1})$. We shall see below how k is determined. We now have the lemma corresponding to Lemma 3.3.1.

Lemma 5.2.1 *Let f be a given element of $\mathbb{Z}[y,x]$ which is squarefree and monic in x. Let s be an integer such that $f(x,s)$ is squarefree and let p be a prime which does not divide $R(f(x,s), f_x(x,s))$. Then we can determine $h^* \in \mathbb{Z}[y,x]$ such that*

1. $\mathrm{lc}_x(h^*) = 1$;
2. $(h^* \bmod (p^k, (y-s)^{d+1}))$ *divides* $(f \bmod (p^k, (y-s)^{d+1}))$ *in* $\mathbb{Z}[y,x]/(p^k, (y-s)^{d+1})$;
3. $(h^* \bmod (p, y-s))$ *is irreducible in* $\mathbb{Z}_p[x]$;
4. $(h^* \bmod (p, y-s))^2$ *does not divide* $(f \bmod (p, y-s))$ *in* $\mathbb{Z}_p[x]$.

We denote $\deg_x(h^*)$ by l.

At this stage a generalisation of Theorem 3.3.1 tells us that f has an irreducible factor h such that $(h^* \bmod (p^k, (y-s)^{d+1}))$ divides $(h \bmod (p^k, (y-s)^{d+1}))$ in $\mathbb{Z}[y,x]/(p^k, (y-s)^{d+1})$.

Now we need a suitable lattice in which to look for a factor. In the univariate case we denoted by m the maximum degree of any polynomial in the lattice.

Correspondingly we now choose two fixed integers m_x and m_y satisfying the inequalities $l \leq m_x \leq n$ and $0 \leq m_y \leq \deg_y(\mathrm{lc}_x(f))$. The lattice $L_{x,y}$ consists of those polynomials $g \in \mathbb{Z}[y,x]$ such that

- $\deg_x(g) \leq m_x$,
- $\deg_y(g) \leq d$,
- $\deg_y(\mathrm{lc}_x(g)) \leq m_y$,
- $(h \bmod (p^k,(y-s)^{d+1}))$ divides $(g \bmod (p^k,(y-s)^{d+1}))$ in $\mathbb{Z}[y,x]/(p^k,(y-s)^{d+1})$.

The dimension of $L_{x,y}$ is M where

$$M = m_x(d+1) + m_y + 1. \tag{5.1}$$

The next result corresponds to Theorem 3.3.2. In it a condition is imposed on k and also an additional condition on s.

Theorem 5.2.1 *Define B by*

$$B = \left(e^{n+d}\,\mathrm{ht}(f)\sqrt{(n+1)(d+1)}\right)^{m_x}\left(\mathrm{ht}(g)\sqrt{(m_x+1)(d+1)}\right)^{n}. \tag{5.2}$$

Suppose that $g \in L_{x,y}$ satisfies

$$|s|^{d+1} > B \tag{5.3}$$

and k is chosen so that

$$p^k > B(1+(1+|s|)^{d+1})^{d(n+m_x-1)}. \tag{5.4}$$

Then h divides g and in particular $\gcd(f,g) \neq 1$ *in* $\mathbb{Z}[y,x]$.

The inequalities in (5.3) and (5.4) are obtained with the aid of results in [44].

Now we can describe how the values of s and k should be chosen. The polynomial g can be thought of as a variable and h as its required value. We want to

choose values of k and s at the start of the algorithm which will be satisfactory whatever the degrees in x and y of the factor h.

First we modify the bound for $|s|$ in (5.3) by substituting the largest values that can occur for m_x, m_y and M:

1. $m_x := n - 1$;

2. $m_y := d_y = \deg_y(\mathrm{lc}_x(f))$;

3. $M := (n-1)(d+1) + d_y + 1$.

In addition we know from a result in [44] that if g divides f then

$$\mathrm{ht}(g) \leq 2^{(M-1)/2}\sqrt{M}e^{n+d}\,\mathrm{ht}(f). \tag{5.5}$$

If $\mathrm{ht}(g)$ is replaced by the right side of (5.5) then we obtain from (5.2) and (5.3) the following bound:

$$|s| > C, \tag{5.6}$$

where

$$C = (e^{n+d}\,\mathrm{ht}(f)\sqrt{(n+1)(d+1)})^{n-1}(2^{(M-1)/2}\sqrt{M}e^{n+d}\sqrt{n(d+1)})^{n}, \tag{5.7}$$

and

$$M = (n-1)(d+1) + \deg_y(\mathrm{lc}_x(f)). \tag{5.8}$$

Next k is chosen to be the least positive integer for which (5.4) holds when the substitutions 1–3 are made:

$$p^k > C(1 + (1 + |s|)^{d+1})^{2d(n-1)}. \tag{5.9}$$

The factor h^* is obtained from $\hat{h}$ by Hensel lifting so that Lemma 5.2.1 holds for this value of k.

If the basis reduction algorithm is applied to the lattice L_{xy} to obtain a reduced basis $(b_1, \ldots, b_M)$ then it follows from Theorem 3.2.1 that

$$|b_M|^2 \leq 2^{M-1}|x|^2 \quad \text{for all } x \in \mathrm{span}(b_1, \ldots, b_M). \tag{5.10}$$

This leads to the following theorem.

Theorem 5.2.2 *Let s and k be chosen as indicated above, let b_M be an element of the reduced basis for L_{xy} (obtained from the reduced basis algorithm of section 3.2) and let $|b|_{\mathcal{M}}$ satisfy (5.10). Then $h \in L_{xy}$ if and only if (5.3) and (5.4) are satisfied with g replaced by b_M.*

This says in effect that if f has a factor it will be contained in the lattice L_{xy} of dimension M which satisfies

$$l(d+1)+1 \leq M \leq (n-1)(d+1)+d_y+1. \tag{5.11}$$

In fact the vector b_M represents the factor h and it may be found by trying each value of M satisfying (5.11) in turn starting with the smallest value and increasing by increments of 1 up to the largest. If h is found we stop; and if h is not found for any M we know that f is irreducible.

Complexity

Lenstra shows that to factorise f completely requires $O(n^7d^6+n^6d^6\log(\mathrm{ht}(f)))$ arithmetic operations on integers of binary length $O(n^4d^3+n^3d^3\log(\mathrm{ht}(f)))$.

Multivariate Case

For an element of $\mathbb{Z}[x_1,x_2,\ldots,x_v]$ the evaluation at $y=s$ is replaced by evaluation at $(x_2,\ldots,x_v)=(s_2,\ldots,s_v)$; and the lifting is done modulo ideals generated by p^{i_1}, $(x_2-s_2)^{i_2},\ldots(x_v-s_v)^{i_v}$. Lenstra describes the estimates corresponding to (5.3) etc. as 'rather complicated'.

5.3 Factorisation in $\mathbb{Z}[\alpha][y,x]$

In [39] Lenstra describes an algorithm for factorising a multivariate polynomial of $t \geq 2$ indeterminates with coefficients in an algebraic number field. A good deal of *ad hoc* notation is required for the exposition. In order to simplify matters as much as possible we give a skeletal description of the bivariate case.

We suppose initially that the field in question is an extension of $\mathbb{Q}$ by an element α which is a root of an irreducible, monic polynomial $F \in \mathbb{Z}[t]$, the minimal

polynomial. The extended field is denoted as usual by $\mathbb{Q}(\alpha)$. The polynomial f to be factorised is a primitive, squarefree element of $\mathbb{Q}(\alpha)[y,x]$ with the property that

$$\mathrm{lc}_y(\mathrm{lc}_x(f)) = 1.$$

We shall want to operate modulo some prime p and we turn our attention first to the way in which p must be chosen. Let d be the defect (see §2.8) of α so that $f \in (1/d)\mathbb{Z}[\alpha][y,x]$ and let $\mathrm{discr}(f)$ be the discriminant of f. Then it may be shown that the factors of f are in $(1/D)\mathbb{Z}[\beta][y,x]$, where $D = d\Delta$ and $\Delta = |\mathrm{discr}(f)|$. The prime p must be chosen coprime with D.

It turns out in the present situation that F must have an approximate factor with properties corresponding to those of the approximate factor of f. Comparison with the LLL algorithm in section 3.3 shows that the following lemma holds.

Lemma 5.3.1 *Let $F \in \mathbb{Z}[t]$ be as above. Then we can determine a prime p coprime with D and $H \in \mathbb{Z}[t]$ such that*

1. *H is monic,*
2. *$(H \bmod p^k)$ divides $(F \bmod p^k)$ in $\mathbb{Z}_{p^k}[t]$,*
3. *$(H \bmod p)$ is irreducible in $\mathbb{Z}_p[t]$,*
4. *$(H \bmod p)^2$ does not divide $(F \bmod p)$ in $\mathbb{Z}_p[t]$.*

We denote the ideal generated by p^k and $(H \bmod p^k)$ by (p^k, H_k), for any $k \in \mathbb{Z}^+$; and we let q denote p^r, where $r = \deg(H)$. The rôle of J is played by (p, H_1) and of J' by (p^k, H_k).

The outline of the algorithm is as follows. First an integer s is chosen and $f(y,x)$ is evaluated at $y = s$. The integer s is not arbitrary but must satisfy a condition similar to (5.3) in section 5.2 and which ensures that $f(s,x)$ is squarefree. Corresponding to Lemma 3.3.1 we have

Lemma 5.3.2 *Let $f \in \mathbb{Q}(\alpha)[y,x]$ and p be as described. Then we can determine $h^* \in \mathbb{Z}[\alpha][x]$ such that*

1. h° *is monic*

2. $(h^{\circ} \bmod (p^k, H_k))$ *divides* $(f(s,x) \bmod (p^k, H_k))$ *in* $\mathbb{Z}_{p^k}[x]/(H \bmod p^k)$

3. $(h^{\circ} \bmod (p, H_1))$ *is irreducible in* $\mathbb{Z}_q[x]$

4. $(h^{\circ} \bmod (p, H_1))^2$ *does not divide* $(f(x,s) \bmod (p, H_1))$ *in* $\mathbb{Z}_q[x]$

The conditions satisfied by h° ensure that f has a factor which is divisible by h° modulo (p^k, H_k).

A lattice of possible factors is formed and a condition described which, if it is satisfied, ensures that the first element of a reduced basis divides f. The condition is based on an upper bound for the height of the factors of f and imposes requirements on the sizes of p and s.

Complexity

Lenstra shows that the computational effort required by the algorithm is

$$O((\deg_x(f)\deg_y(f)\deg(h^{\circ})\deg(F))^4 k \log(p))$$

operations on integers of binary length

$$O(\deg(F)\deg_x(f) k \log(p)).$$

5.4 Factorisation in $\mathbb{Z}_q[y,x]$

The finite field of the title has q elements, where q is a power of a prime. This field is denoted by $\mathbb{Z}_q$ and we look at an algorithm for factoring $f \in \mathbb{Z}_q[y,x]$. We first give a brief description of the lattice which is used and some of its properties.

Lattices over $\mathbb{Z}_q[y]$

Let $b_1, b_2, \ldots, b_n \in \mathbb{Z}_q[y]^n$ be linearly independent over $\mathbb{Z}_q[y]$. The lattice L of rank n spanned by $b_1, \ldots, b_n$ is

$$L = \left\{ \sum_{i=1}^{n} r_i b_i \mid r_i \in \mathbb{Z}_q[y] \text{ for } 1 \le i \le n \right\}. \tag{5.12}$$

The degree $\deg(a)$ of a vector $a \in \mathbb{Z}_q[y]^n$ is the maximum degree of any of its components. The determinant $d(L)$ of the lattice is defined, as it was in section 3.2, to be the determinant of the matrix B which has $b_1, \ldots, b_n$ as its columns. The reader should note that in his paper [38] Lenstra calls the degree of a vector its norm and denotes it by $|\cdot|$. This notation is avoided in the thesis because of a conflict with its earlier use.

The *orthogonality defect* $\delta(b_1, \ldots, b_n)$ of a basis is defined by

$$\delta(b_1, \ldots, b_n) = \left(\sum_{i=1}^{n} \deg(b_i)\right) - \deg(d(L)). \tag{5.13}$$

The algorithm below requires the concept of a *j-th successive minimum* $\deg(m_j)$ of a lattice which is defined recursively as the degree of an element of least degree which is linearly independent of $m_1, \ldots, m_{j-1}$. It can be shown that the j-th successive minimum is independent of the particular choice of $m_1, \ldots, m_{j-1}$ [42], [45].

The following definition of a reduced basis is valid only in this subsection. In it b_{ij} denotes the i-th component of b_j so that the i-th row of B contains the i-th coordinates of the b_j.

Definition 5.4.1 *A basis $b_1, \ldots, b_n$ is said to be* reduced *if the rows of the matrix B can be permuted in such a way that the columns $b_1^*, \ldots, b_n^*$ of the resulting matrix satisfy*

$$\deg(b_i^*) \leq \deg(b_j^*) \text{ for } 1 \leq i \leq n, \tag{5.14}$$

$$\deg(b_{ii}^*) \geq \deg(b_{ij}^*) \text{ for } 1 \leq j < i \leq n, \tag{5.15}$$

and

$$\deg(b_{ii}^*) > \deg(b_{ij}^*) \text{ for } 1 < j < i \leq n. \tag{5.16}$$

One can show that the orthogonal defect of a reduced basis is zero.

Lenstra describes an algorithm which, from a given basis, determines one which is reduced in the sense just described. The structure of the algorithm is similar to the earlier example in Chapter 3 and it requires $O(n^3 D(\delta(b_1, \ldots, b_n) + 1))$

arithmetic operations in $\mathbb{Z}_q$. The constant D, which must be at least two, is an upper bound for the degree of every term in the matrix B.

The reduced basis has its application through the following result which corresponds in the present context to Theorem 3.2.1 in the LLL algorithm.

Theorem 5.4.1 *Let $b_1, \ldots, b_n$ be a basis for a lattice L satisfying $\delta(b_1, \ldots, b_n) = 0$ and ordered in such a way that $\deg(b_i) \leq \deg(b_j)$ for $1 \leq i < j \leq n$. Then $\deg(b_j)$ is the j-th successive minimum of L for $1 \leq j \leq n$ and in particular $\deg(b_1) \leq \deg(x)$ for every non-zero x in L.*

The conditions of the theorem are met if the basis of L is reduced so that the j-th element of a reduced basis is also the j-th successive minimum.

The Factorisation Algorithm

The first step in the algorithm is to determine a polynomial $F \in \mathbb{Z}_q[y]$ of degree u which will generate a suitable ideal for us. Proceed as follows. If $q > \deg R(f, f')$ then choose $s \in \mathbb{Z}_q$ such that $(y-s)$ does not divide $R(f, f')$. In this case $F = y - s$ and $u = \deg(F) = 1$. If on the other hand $q \leq \deg R(f, f')$ take $v \in \mathbb{Z}^+$ to be the least number such that $q^v > \deg R(f, f')$ and determine (by a search) a monic irreducible polynomial G of $\mathbb{Z}_q[y]$ of degree v. An element β in $\mathbb{Z}_q[y]/(G)$ is determined such that $(y - \beta)$ does not divide $R(f, f')$. Then choose $F \in \mathbb{Z}_q[y]$ to be the minimal polynomial of β in which case $u = \deg(F) \leq v$. The fact that this process can be carried out in polynomial time, and the suitability of F, are justified in [38].

For a positive integer k we denote by (F^k) the ideal generated by F^k. If $\alpha = (y \bmod F^k)$ is a zero of F^k we can represent elements of the ring $\mathbb{Z}_q[y]/(F^k)$ as polynomials in α over $\mathbb{Z}_q$ of degree less than uk. We note that $\mathbb{Z}_q[y]/(F)$ is isomorphic to $\mathbb{Z}_{q^u}$. Finally, if $g = \sum_i b_i x^i \in \mathbb{Z}_q[y][x]$ we denote the polynomial $\sum_i (b_i \bmod F^k) x^i \in (\mathbb{Z}_q[y]/(F^k))[x]$ by $(g \bmod F^k)$.

Berlekamp's algorithm computes an element $\hat{h}$ of $\mathbb{Z}_q[y, x]$ such that $(\hat{h} \bmod F)$ is a factor of $(f \bmod F)$ in $(\mathbb{Z}_q[y]/(F))[x]$ and is irreducible and monic with respect to x. By invoking Hensel lifting we obtain a result corresponding to Lemma 3.3.1.

Lemma 5.4.1 *Let $f \in \mathbb{Z}_q[y,x]$ be primitive and squarefree and let k be a positive integer. Then we can determine an irreducible monic polynomial $F \in \mathbb{Z}_q[y]$ and a polynomial $h^* \in \mathbb{Z}_q[x,y]$ such that*

1. *h^* is monic with respect to x;*
2. *$(h^* \bmod F^k)$ divides $(f \bmod F^k)$ in $(\mathbb{Z}_q[y]/(F^k))[x]$;*
3. *$(h^* \bmod F)$ is irreducible in $\mathbb{Z}_{q^u}[x]$;*
4. *$(h^* \bmod F)^2$ does not divide $(f \bmod F)$ in $\mathbb{Z}_{q^u}[x]$.*

If these conditions hold then the following theorem corresponding to Theorem 3.3.1 holds.

Theorem 5.4.2 *Let the polynomial $f \in \mathbb{Z}_q[y,x]$ be squarefree and monic in x, and let h^* satisfy the conditions above. Then f has an irreducible factor $h \in \mathbb{Z}_q[y,x]$ such that $(h^* \bmod F)$ divides $(h \bmod F)$. Further if g divides f in $\mathbb{Z}_q[y,x]$ then the following three assertions are equivalent:*

1. *$(h^* \bmod F)$ divides $(g \bmod F)$ in $\mathbb{Z}_{q^u}[x]$;*
2. *$(h^* \bmod F^k)$ divides $(g \bmod F^k)$ in $(\mathbb{Z}_q[y]/(F^k))[x]$;*
3. *h divides g in $\mathbb{Z}_q[y,x]$.*

In particular $(h^* \bmod F^k)$ divides $(h \bmod F^k)$ in $(\mathbb{Z}_q[y]/(F^k))[x]$.

The factor which we are seeking is divisible by h^* and so its degree in x must be at least as great as $\deg_x(h^*)$. We let m be a parameter satisfying $\deg_x(h^*) \leq m < \deg_x(f)$ and choose the lattice L_{xy} to be the collection of polynomials $g \in \mathbb{Z}_q[y,x]$ such that $\deg_x(g) \leq m$ and such that $(h^* \bmod F^k)$ divides $(g \bmod F^k)$ in $(\mathbb{Z}_q[y]/(F^k))[x]$. The condition for an element $b \in L_{xy}$ to be divisible by h is given in the next theorem which corresponds to Theorem 3.3.2 in the LLL algorithm.

Theorem 5.4.3 *Let $b \in L_{xy}$ satisfy*

$$(\deg_y(f))(\deg_x(b)) + (\deg_y(b))(\deg_x(f)) < uk \deg(h^*). \tag{5.17}$$

Then b is divisible by h in $\mathbb{Z}_q[y,x]$. In particular $\gcd(f,b) \neq 1$.

The final result corresponds to Theorem 3.3.5 in the LLL algorithm and shows how a reduced basis may be used to determine whether or not a lattice contains a factor of f.

Theorem 5.4.4 *Let h and h^* be as above, let $b_1, \ldots, b_n$ be a reduced basis for L_{xy} and let*

$$m \deg_y(f) + (\deg_y(f))(\deg_x(f)) < uk \deg_x(h^*) \tag{5.18}$$

be satisfied. Then the following three assertions are equivalent:

1. $\deg_x(h) \leq m$;

2. $\deg_y(b_1) \leq \deg_y(f)$;

3. $b_1 = dh$ *for some* $d \in \mathbb{Z}_q[x]$.

If h is to be a proper factor of f then its degree in x must not exceed $\deg_x(f)-1$. The integer k is therefore chosen to be the smallest positive integer for which equation (5.18) holds when m is replaced by $\deg_x(f) - 1$. The search for a factor is now carried out according to the schema below.

```
begin
  m := deg_x(h*);
  repeat
    form the lattice L_xy with the current value of m;
    determine a reduced basis for L_xy;
      {Use Theorem 5.4.4 to look for a factor h of f.}
    if condition (5.18) is satisfied then
      begin output the factor h; exit end
    m := m + 1;
  until (a factor h is found) or (m = deg_x(f));
    {If we reach here f is irreducible.}
end
```

Complexity

The complexity of the algorithm is given by the following theorem.

Theorem 5.4.5 *Let f be an element of $\mathbb{Z}_q[y,x]$ which is squarefree and monic in x. Then the factorisation of f into irreducible factors in $\mathbb{Z}_q[y,x]$ can be determined in $O((\deg_x(f))^6(\deg_y(f))^2 + pr(\deg_x(f))^3 + pr(\deg_y(f))^3)$ arithmetic operations in $\mathbb{Z}_q$, where $q = p^r$.*

Chapter 6

Other Developments

6.1 Introduction

This chapter contains a brief account of some of the developments in factorisation which have taken place in the last ten years or so and which are not described elsewhere. Some of these are theoretical results, but section 6.2 is concerned with a practical algorithm, so that there is some overlap between this chapter and the next.

The algorithms described in section 6.2 and 6.4 are randomised: if a number of independent trials with random choices is made there is a high probability that the correct answer is obtained.

6.2 Factorisation over a large finite field

As we saw in section 2.4 Berlekamp's algorithm applied to a polynomial of degree n in $\mathbb{Z}_p[x]$ requires $O(n^4p)$ field operations for a complete factorisation. This may be slow if n or p is large. Since factorisation in a finite field is a first step in all the algorithms we have described for determining the factors of a polynomial over $\mathbb{Z}$ or $\mathbb{Q}(\alpha)$, it is desirable to find a quick factorisation algorithm in the finite field case. Advances have been made by Berlekamp [6], Rabin [51] and by Cantor and Zassenhaus [11]. All of these papers offer a randomised algorithm for the finite field case. We describe the method in [11] which is a modification and improvement of the one in [51].

In the general case one wants to find the factors of $f \in \mathbb{Z}_q[x]$, where q is a power of a prime. We are mainly interested in the case when a factor of $f \in \mathbb{Z}[x]$ is approximated by a factor of $(f \bmod p)$ in $\mathbb{Z}_p[x]$, and this approximation is raised by Hensel lifting to one in $\mathbb{Z}_q[x]$, where $q = p^k$. We shall therefore confine attention to $\mathbb{Z}_p[x]$.

We need first two results which are stated as lemmas. For proofs see [41] or [17].

Lemma 6.2.1 *Let $u(x)$ be an irreducible element of $\mathbb{Z}_p[x]$ of degree d. Then $u(x)$ divides $x^{p^d} - x$ and is not a divisor of $x^{p^c} - x$ for any $c < d$.*

Lemma 6.2.2 *Let p be an odd prime, $g_d(x) \in \mathbb{Z}_p[x]$ be a product of distinct irreducible factors of degree d and $t(x)$ be any element of $\mathbb{Z}_p[x]$. Then*

$$g_d = \gcd(g_d, t)\,\gcd(g_d, t^{(p^d-1)/2} - 1)\,\gcd(g_d, t^{(p^d-1)/2} + 1).$$

If we are given a squarefree element f of $\mathbb{Z}_p[x]$, Lemma 6.2.1 indicates how to find the product of all the factors of f of each degree d. This is the basis of the distinct degree factorisation algorithm. The procedure *distinct_degree* overleaf outputs polynomials $g_d(x)$ for $1 \leq d \leq \lfloor n/2 \rfloor$, where g_d is the product of all the factors of degree d. On termination the variable *afactor* contains either the integer 1 or the unique factor of degree greater than $\lfloor n/2 \rfloor$.

Now instead of needing to factorise f we require instead to factorise those g_d which are not irreducible, that is, whose degree exceeds d. For this Cantor and Zassenhaus suggest using the identity in Lemma 6.2.2. They show that if $t(x)$ is chosen randomly from the polynomials of $\mathbb{Z}_p[x]$ whose degree does not exceed $(2d-1)$ then the probability that $\gcd(g_d, t^{(p^d-1)/2} - 1)$ is a non-trivial factor of f is about one half.

```
procedure distinct_degree(f);
    {f is an element of Z_p[x]and all operations are in Z_p[x]}
  begin
    v(x) := f(x); w(x) := x; d := 0;
    while d < ⌊n/2⌋ do
      begin
        d := d + 1;
        w(x) := w(x)^p   (mod f(x));
        g_d(x) := gcd(w(x) − x, f(x));
        f(x) := f(x)/g_d(x);
        w(x) := w(x)   (mod f(x));
      end;
    afactor:=f(x)
  end
```

Very recently Rothstein and Zassenaus [52] have proposed a new deterministic method for factorising univariate polynomials in a finite field but it is not clear yet how this works in practice.

6.3 Factorisation in $\mathbb{Z}[\alpha][x]$

In [33] Landau gives a method for factorising a polynomial over a number field $\mathbb{Z}[\alpha]$ which uses the LLL algorithm as a subroutine, but does not use a lattice over $\mathbb{Q}(\alpha)$.The idea of the paper is that given a polynomial $f \in \mathbb{Q}(\alpha)[x]$ there is, corresponding to it, a polynomial $\phi \in \mathbb{Q}[x]$ with the properties that

- ϕ can be determined from f
- the factors of f can be deduced from the factors of ϕ

The polynomial ϕ is factored by the LLL algorithm.

$\mathbb{Q}(\alpha)$ is an algebraic number field isomorphic to $\mathbb{Q}(t)/(g(t))$, where $g \in \mathbb{Q}[t]$ is the minimal polynomial of α and $\deg(g) = m$. The distinct conjugates of α are $\alpha_1 = \alpha, \alpha_2, \ldots, \alpha_m$ and we assume that these are algebraic integers.

An essential tool in Landau's paper is the *norm* which is a mapping from $\mathbb{Q}(\alpha)$ to $\mathbb{Q}$. The norm is a standard field-theoretic tool [43] which has the property that it is multiplicative, namely, $N(\beta\gamma) = N(\beta)N(\gamma)$. Suppose that $\beta = a_0 + a_1\alpha + \cdots + a_{m-1}\alpha^{m-1} \in \mathbb{Q}(\alpha)$. It may be shown that

$$N(\beta) = \prod_{i=1}^{m}(a_0 + a_1\alpha_i + \cdots + a_{m-1}\alpha_i^{m-1}),$$

and that $N(\beta) \in \mathbb{Q}$. The norm may be extended to $f \in \mathbb{Q}(\alpha)[x]$ as follows. Since the coefficients of f are polynomials in $\mathbb{Q}[\alpha]$ we may, with an abuse of notation, write f as $f(x, \alpha)$ and then the norm of $f(x)$ is

$$N(f(x)) = \prod_{i=1}^{m} f(x, \alpha_i).$$

It can be shown that $N(f(x))$ is in $\mathbb{Q}[x]$.

For a squarefree element $f \in \mathbb{Q}(\alpha)[x]$ the outline of Landau's algorithm is as follows.

1. Determine $N(f(x))$.

2. Factor $N(f(x))$ using the LLL algorithm.

3. Recover the factors of f from the factors of $N(f(x))$.

The first step requires an efficient way to evaluate the norm. Finding it directly from the definition could require an exponential amount of calculation, so instead the norm is determined from the resultant with respect to t of $f(x,t)$ and the minimal polynomial $g(t)$. It can be shown that if $g(t)$ is monic then

$$N(f(x)) = R_t(g(t), f(x,t)).$$

Now we turn attention to the third step of the algorithm. Landau proves first the following result.

Lemma 6.3.1 *Let $f(x) \in \mathbb{Q}(\alpha)[x]$ be irreducible. Then $N(f(x))$ is a power of an irreducible polynomial.*

Therefore if $N(f(x))$ is irreducible so is $f(x)$. To obtain a guarantee that an irreducible $f(x)$ will result in an irreducible $N(f(x))$ we need the next result. Recollect that the minimal polynomial of α has degree m.

Lemma 6.3.2 *Let $f \in \mathbb{Q}(\alpha)[x]$ be squarefree of degree n. Then there are at most $\frac{1}{2}(mn)^2$ integers s such that $N(f(x - s\alpha))$ has a repeated factor.*

Reference to Lemma 6.3.1 shows that is possible to choose an s so that if $f(x)$ is irreducible then $N(f(x - s\alpha))$ is also irreducible. The last result shows that a factorisation of $f(x)$ may be obtained from a factorisation of $N(f(x - s\alpha))$.

Theorem 6.3.1 *Let $f(x) \in \mathbb{Q}(\alpha)[x]$ and $s \in \mathbb{Z}$ be such that $N(f(x - s\alpha))$ is squarefree. If*

$$N(f(x - s\alpha)) = \prod_{j=1}^{r} F_j(x)$$

is a factorisation into irreducible polynomials in $\mathbb{Q}[x]$ then

$$f(x) = \prod_{j=1}^{r} \gcd(F_j(x + s\alpha), f(x))$$

is a factorisation into irreducibles in $\mathbb{Q}(\alpha)[x]$.

The outline of the algorithm is now

1. Determine $s \in \mathbb{Z}$ such that $N(f(x - s\alpha))$ is squarefree.

2. Use the LLL algorithm to factor $N(f(x - s\alpha))$ into a product of irreducibles $F_j(x), j = 1, \ldots, r$.

3. Determine the factors $f_j(x)$ of $f(x)$ from

$$f_j(x) = \gcd(F_j(x + s\alpha), f(x)) \text{ for } j = 1, \ldots, r.$$

To describe the complexity we need some definitions.

Definition 6.3.1 *If* $\beta \in \mathbb{Q}(\alpha)$ *is given by*

$$\beta = b_0 + b_1\alpha + \cdots + b_{m-1}\alpha^{m-1} \in \mathbb{Q}(\alpha)$$

its absolute value $|\beta|$ *is given by*

$$|\beta| = \left(\sum_{r=0}^{m-1} b_r^2\right)^{1/2}.$$

Definition 6.3.2 *The* size $\|\beta\|$ *of* $\beta \in \mathbb{Q}(\alpha)$ *is the maximum of the absolute values of the conjugates of* β.

Definition 6.3.3 *The* height $\mathrm{ht}(f)$ *of*

$$f = \beta_0 + \beta_1 x + \cdots + \beta_n x^n \in \mathbb{Q}(\alpha)[x]$$

is the maximum of the absolute values of the β_j, $j = 0, \ldots, n$.

Landau shows that the complexity of the algorithm is

$$O(m^{9+\epsilon} n^{7+\epsilon}(\log^{2+\epsilon}(\mathrm{ht}(f)) + n\log^{2+\epsilon}(m^2 n|g|)))$$

$$+ O(m^7 n^8 \log^2(|g|)(\log^2(\mathrm{ht}(f)) + m^2\log(m|g|)))$$

for any $\epsilon > 0$. The first term comes from the LLL algorithm in step 2 of the algorithm and the second from the determination of the $f_j(x)$. The complexity is measured in terms of binary operations.

6.4 Bivariate Polynomials over Finite Fields

Kaltofen and von zur Gathen have described in [23] an algorithm (called Quick Factor) which, for any finite field $\mathbb{Z}_q$, uses Hensel lifting to determine a factorisation of $f(y,x) \in \mathbb{Z}_q[y,x]$ from a factorisation of $f(0,x) \in \mathbb{Z}_q[x]$. The amount of computation required by the algorithm depends on the effort required to factorise a univariate polynomial in $\mathbb{Z}_q[x]$. If we call this $c(e)$ for a univariate polynomial of degree e, then the number of operations in F carried out by the bivariate factorisation algorithm when factoring $f(y,x)$ of degree d_x in x is

$$O(n^4 d_x^4) + nc(d_x).$$

This is bounded above by

$$O(n^8) + nc(n), \tag{6.1}$$

where n is the total degree of $f(y,x)$. The $O(n^8)$ term arises from the Hensel lifting.

If Berlekamp's deterministic algorithm for factorising in $\mathbb{Z}_q[x]$ is used, then the result (6.1) gives the number of arithmetic operations required in $\mathbb{Z}_q$ to find a complete factorisation of $f(y,x)$. If the factorisation in $\mathbb{Z}_q[x]$ is done by the Cantor-Zassenhaus probabilistic algorithm the probability that the correct factorisation has not been found can be made $O(2^{-n})$ in $O(n)$ passes. The second term in (6.1) should be replaced by the appropriate term. The first term still dominates however and Quick Factor remains $O(n^8)$.

We do not give a description of the algorithm, which is very similar to the one presented in Chapter 4. In common with almost all the algorithms in this thesis Quick Factor requires a monic, squarefree input. A polynomial over a finite field can be made monic by a multiplication. The authors present a gcd algorithm which enables the squarefree part of a polynomial to be determined.

6.5 Sparse Representations

For any ring R the sparse representation of a polynomial in $R[x]$ consists of a list of pairs $(a, j) \in R \times \mathbb{N}$, each pair containing a non-zero coefficient a and a corresponding power j of x. In the case of a multivariate polynomial in indeterminates $x_1, \ldots, x_v$ the single power j is replaced by an indexed list of powers defining the appropriate monomial. By contrast, the dense representation, which we have used so far, consists of an ordered list of all the coefficients, including the zero coefficients. For example, the sparse representation of $x^n - 1$ is the two element list $((1, n), (-1, 0)$ while its dense representation is the $(n+1)$ element list $(1, 0, \ldots, 0, -1)$

The irreducible factorisation of $x^n - 1 \in \mathbb{Z}[x]$ is

$$(x-1)(x^{n-1} + x^{n-2} + \cdots + x + 1). \tag{6.2}$$

We see that both the dense and sparse representations of (6.2) have a total of $(n+2)$ elements. Since $(n+2) = 2^{\log_2(n+2)}$, the size of (6.2) is subexponential in the size of the sparse representation of $x^n - 1$, but not in the size of its dense representation. An algorithm whose running time is polynomial in the dense size of a polynomial need not have (and usually will not have) a running time which is polynomial in the sparse size. Indeed Plaisted [47] has shown that the problem of finding the gcd of two sparsely represented polynomials in $\mathbb{Z}[x]$ is NP-hard.

Another way of representing polynomials is through straight line programs. A description of these may be found in [31], but some idea can be gained by considering the following code, in which on the first line f refers to the polynomial $1 + x^d$ and the multiplications are symbolic.

$$\begin{aligned} f_1 &:= f; \\ f_2 &:= f_1 * f_1; \\ &\cdots \\ f_j &:= f_{j-1} * f_{j-1}; \end{aligned}$$

$$\ldots$$

$$f_n := f_{n-1} * f_{n-1}$$

Here code of length $O(n)$ has generated a polynomial whose dense representation has $\Omega(2^n)$ terms.

In order to obtain useful results it is necessary to pose questions appropriately. For example Kaltofen [29] has shown that with high probability the gcd of two polynomials given by straight line programs can be found it terms of a straight line program with a polynomial number of steps. In [30] he has shown that a similar result holds for factorisation. Details of this and other work may be found in the references cited as well as in [18].

Chapter 7

Practical Algorithms

7.1 Greatest Common Divisors

The factorisation algorithms which we have described have all required that the input polynomial is squarefree: given an arbitrary polynomial $\hat{f}$ in $R[x]$ we can obtain from it a squarefree polynomial $f = \hat{f}/\gcd(\hat{f}, \hat{f}')$. Here $\hat{f}'$ is the derivative of $\hat{f}$ with respect to x. We begin this section with a description of the modular gcd algorithm which, as we shall see, is often more efficient than older algorithms.

When the Euclidean algorithm is used to determine the gcd of two polynomials in $\mathbb{Q}[x]$, intermediate expressions may have exponential growth in size. Consider the following (well known) example [32], [7], [15].

Example 7.1.1 In determining, by the Euclidean algorithm, the gcd of

$$f(x) = x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5$$

and

$$3x^6 + 5x^4 - 4x^2 - 9x + 21$$

the following polynomials are generated:

$$-\frac{5}{9}x^4 + \frac{1}{9}x^2 - \frac{1}{3},$$

$$-\frac{117}{25}x^2 - 9x + \frac{441}{25},$$

$$\frac{233150}{6591}x - \frac{102500}{2197}$$

and

$$-\frac{1288744821}{543589225}.$$

Thus the gcd of the given polynomials is 1.

If we want to work in $\mathbb{Z}[x]$ division must be replaced by *pseudo-division.* What this means is that $f \in \mathbb{Z}[x]$ is multiplied by a suitable integer before division by $g \in \mathbb{Z}[x]$, so that the quotient and remainder are in $\mathbb{Z}[x]$. To be precise, if the degrees of f and g are n and m respectively ($n \geq m$) and $\mathrm{lc}(g) = b_m$ then f is multiplied by b_m^{n-m+1}.

Example 7.1.2 Using a pseudo-division Euclidean algorithm on the polynomials of the previous example generates the polynomials

$$-15x^4 - 3x^2 - 9,$$

$$15795x^2 + 30375x - 59535,$$

$$1254542875143750x - 1654608338437500$$

and

$$1259333879550074310093114199218750 0.$$

It is clear that the arithmetic would be greatly reduced if we could work modulo a small prime.

Example 7.1.3 Repeating the previous example, but working modulo 5, we find

$$f(x) \equiv x^8 + x^6 + 2x^4 + 2x^3 + x^2 + 2x \pmod 5,$$

$$g(x) \equiv 3x^6 + x^2 + x + 1 \pmod 5,$$

and the polynomials generated are

$$2x^2 + 3,$$

$$x$$

and

$$3.$$

The modular gcd algorithm is based on the idea that we can use calculations like those in the Example 7.1.3 to find the gcd of two polynomials in $\mathbb{Z}[x]$. It will be useful for this section to use the notation f_p for (f mod p), $\gcd(f,g)_p$ for ($\gcd(f,g)$ mod p), and to write $\gcd(f_p,g_p)$ to indicate the gcd of f_p and g_p found from Euclid's algorithm in $\mathbb{Z}_p$.

Note first that if $h = \gcd(f,g)$ then h_p divides $\gcd(f_p,g_p)$, but in general $\gcd(f,g)_p \neq \gcd(f_p,g_p)$. For example $f = x+2$ and $g = x-3$ are coprime in $\mathbb{Z}[x]$ so that $\gcd(f,g) = \gcd(f,g)_5 = 1$. On the other hand $f_5 = g_5 = \gcd(f_5,g_5) = x+2$. Thus if $\gcd(f_p,g_p) \neq 1$ it does not follow that $\gcd(f,g) \neq 1$. We shall see that it is possible to choose p in such a way that, with the obvious abuse of notation,

$$\gcd(f,g) = \gcd(f,g)_p = \gcd(f_p,g_p).$$

We give an outline of the theory, details of which can be found in [15], §4.1.1.1.

Definition 7.1.1 *If f and g are elements of $\mathbb{Z}[x]$ and p is a prime such that $\gcd(f_p,g_p) = \gcd(f,g)_p$ then p is said to be of* good reduction *for f and g; otherwise p is said to be of* bad reduction.

We want to choose primes p which are of good reduction for the problem in hand. The following lemma shows that for any $f,g \in \mathbb{Z}[x]$ there are at most finitely many primes of bad reduction.

Lemma 7.1.1 *Let f and g be elements of $\mathbb{Z}[x]$ and $h = \gcd(f,g)$. Then p is a prime of good reduction if*

1. *p does not divide both $\mathrm{lc}(f)$ and $\mathrm{lc}(g)$; and*

2. *p does not divide the resultant $R(f/h, g/h)$.*

An immediate consequence of Lemma 7.1.1 is that if $\gcd(f,g) = 1$ we can find p such that $\gcd(f_p,g_p) = 1$.

A prime p will satisfy the conditions of Lemma 7.1.1 provided it is sufficiently large. We can find a lower bound for the required size of p from a corollary, which we state as a lemma, of the Landau-Mignotte inequality of Theorem 3.3.3.

Lemma 7.1.2 *Let* $f = \sum_{i=0}^{n} a_i x^i$ *and* $g = \sum_{i=0}^{m} b_i x^i$ *be elements of* $\mathbb{Z}[x]$. *Then every coefficient of* $\gcd(f,g)$ *is bounded by*

$$M = 2^{\min(n,m)} \gcd(a_0, b_0) \min\left(\frac{1}{|a_0|}\sqrt{\sum_{i=0}^{n} a_i^2}, \frac{1}{|b_0|}\sqrt{\sum_{i=0}^{m} b_i^2} \right).$$

We shall refer to M as the Landau-Mignotte gcd-bound of f and g. (This is not the same as the Landau-Mignotte factor-bound A of (3.11).)

Since any coefficient c of $\gcd(f,g)$ satisfies $-M \leq c \leq M$, it lies in an interval of length $2M$. The prime p must be chosen so that the $2M+1$ integers

$$-M, -M+1, \ldots, 0, \ldots M-1, M$$

are distinct modulo p. We therefore choose $p > 2M$ and since p exceeds M it does not divide either $\mathrm{lc}(f)$ or $\mathrm{lc}(g)$. The modular gcd algorithm is as follows.

```
begin
  M:= Landau-Mignotte gcd-bound of f and g;
  repeat
    determine a new prime p exceeding 2M;
    h_p := gcd(f_p, g_p)
  until (h divides f and h divides g);
  gcd(f,g) := h
end
```

The coefficients of h are uniquely determined from h_p because $p > 2M$. In each execution of the repeat loop, a prime p different from all the previous ones is used.

A disadvantage of the algorithm as presented is that the lower bound M of the primes may be rather large. One way of avoiding large primes is to use several smaller ones and reconstruct $\gcd(f,g)$ using the Chinese Remainder Theorem, so that from $\gcd(f_p, g_p)$ and $\gcd(f_q, g_q)$ we determine $\gcd(f_{pq}, g_{pq})$. In this situation the CRT is applied to the integer coefficients and not to the polynomial greatest common divisors. Details can be found in [15 §4.1.1.2].

Example 7.1.4 Let $f = x^5 + 8x^4 + 14x^3 + 17x^2 + 32x + 12$ and $g = x^4 + 15x^3 + 65x^2 + 60x - 36$. Then $2M = 1024.61$ and the smallest prime exceeding this is 1031. However we easily find that

$$\begin{aligned} f &\equiv x^5 + 2x^4 + 2x^3 + 2x^2 + 2x \pmod 3, \\ g &\equiv x^4 + 2x^2 \pmod 3, \end{aligned}$$

and the gcd of these polynomials in $\mathbb{Z}_3[x]$ is $x^2 + 2x$.

Similarly we have

$$\begin{aligned} f &\equiv x^5 + 3x^4 + 4x^3 + 2x^2 + 2x + 2 \pmod 5, \\ g &\equiv x^4 + 4 \pmod 5, \end{aligned}$$

and in this case the gcd in $\mathbb{Z}_5[x]$ is $x^2 + 3x + 2$. Hence

$$\begin{aligned} \gcd(f_3, g_3) &= x^2 + 2x \quad \in \mathbb{Z}_3[x], \\ \gcd(f_5, g_5) &= x^2 + 3x + 2 \quad \in \mathbb{Z}_5[x], \end{aligned}$$

from which the CRT gives

$$\gcd(f_{15}, g_{15}) = x^2 + 8x + 12 \quad \in \mathbb{Z}_{15}[x].$$

Continuing in this way we find that $\gcd(f_7, g_7) = x^2 + x + 5$ and that $\gcd(f_{11}, g_{11}) = x^2 + 8x + 1$. Further application of the CRT gives $\gcd(f_{1155}, g_{1155}) = x^2 + 8x + 12$. Since $1155 > 2M$ we deduce that $\gcd(f, g) = x^2 + 8x + 12$.

A similar modular algorithm can be constructed for multivariate polynomials. In the case when f and g are in $\mathbb{Z}[y, x]$ the idea is to look at f and g modulo $(y - a)$, for some suitable $a \in \mathbb{Z}$, which is equivalent to evaluating f and g at $y = a$. Since $\gcd(f(a, x), g(a, x))$ divides $(\gcd(f, g))(a, x)$, we can hope to reconstruct $\gcd(f, g)$ from $\gcd(f(a, x), g(a, x))$. Just as in the case of univariate polynomials we encounter cases of bad reduction, so that the algorithm uses a set of modular reductions with integers $a_1, a_2, \ldots, a_r$ and a reconstruction of the gcd using the Chinese Remainder Theorem.

When f and g are polynomials in more than two indeterminates it is possible to proceed recursively reducing the number of indeterminates at each recursion. If, for example, f and g are regarded as elements of $\mathbb{Z}[x_1, \ldots, x_{r-1}][x_r]$ then $(f \bmod (x_r - a))$ can be regarded as an element of $\mathbb{Z}[x_1, \ldots, x_{r-1}]$. Useful references are [15] and [9].

7.2 Factorising polynomials with coefficients in $\mathbb{Z}$ or $\mathbb{Z}[\alpha]$

7.2.1 Introduction

Two algorithms for factorising polynomials with coefficients in $\mathbb{Z}_p$—or GF(p) to which it is isomorphic—have been described, namely Berlekamp's method in section 2.5 and the Cantor-Zassenhaus method in section 6.2. Since these are also practical methods if p is not too large, nothing more remains to be said except that both may be generalised to polynomials with coefficients in GF(p^r) by doing arithmetic in GF(p^r). The polynomial time methods based on lattices for factorising polynomials with coefficients in $\mathbb{Z}$ or $\mathbb{Z}[\alpha]$ are not recommended by their authors for practical use, however. In this section we shall cover the recommended methods which all have an exponential worst-case behaviour, but are fast in practice.

Methods for factorising over $\mathbb{Z}$ have the following general schema.

- Given $f \in \mathbb{Z}[x]$, let (p) be an ideal of $\mathbb{Z}$ and $f_p := (f \bmod p)$.
- Determine by a finite field algorithm the irreducible factors $\{h_1, \ldots, h_r\}$ of f_p in $\mathbb{Z}_p[x]$ so that $f \equiv h_1 \cdots h_r \pmod{p}$.
- Use $\{h_1, \ldots, h_r\}$ to determine the irreducible factors $\{g_1, \ldots, g_t\}$ of f so that $f = g_1 \cdots g_t$.

All three steps are capable of generalisation or elaboration. For example if $f \in \mathbb{Z}[\alpha][x]$, where α satisfies an irreducible polynomial of degree m, we can regard $(f \bmod p)$ as an element of GF(p^m)[x]. If f is a bivariate polynomial in $\mathbb{Z}[y, x]$ then $(f \bmod (y-a))$ is an element of $\mathbb{Z}[x]$. It is evident that further generalisation to multivariate polynomials with coefficients in either $\mathbb{Z}$ or $\mathbb{Z}[\alpha]$ is possible.

In the second step there are reasons why we may wish the prime p to be large, but this will slow up the finite field factorisation algorithm. The strategy adopted

is the one we have seen earlier, that is, to use a small value of p and then determine from Hensel lifting $\{h_1^{(k)}, \ldots, h_r^{(k)}\}$ such that $f \equiv h_1^{(k)} \cdots h_r^{(k)} \pmod{p^k}$.

The use of modular methods with several moduli and reconstruction by means of the CRT is unfortunately not suitable. An explanation of this is given at the end of the next section.

The final step is the hardest one and in the worst case may require search through a number of cases which is exponential in the degree of f. One algorithm due to A. K. Lenstra [36] uses a lattice to aid the search: this is described at the end of this chapter.

7.2.2 Univariate polynomials over $\mathbb{Z}$

Given a polynomial $f \in \mathbb{Z}[x]$ we first find $f_p = (f \bmod p)$ and factorise f_p using either Berlekamp's algorithm or the Cantor-Zassenhaus algorithm. If the factors of f_p are $h'_1, \ldots, h'_r$ we have

$$f \equiv h'_1 \cdots h'_r \pmod{p}. \tag{7.1}$$

Not all of the h'_i will correspond to true factors of f in $\mathbb{Z}[x]$. In fact it may be that none of them does. However it simplifies the task if any true factor of f has the same coefficients as its modular image. We don't know the factor coefficients in advance but the Landau-Mignotte Theorem 3.3.3 shows that this will be achieved if the prime p is chosen to be twice the Landau-Mignotte factor-bound Λ of inequality (3.11). (The factor 2 allows for a choice of sign.)

In practice it seems to be more efficient to use a small prime p and raise the factors in (7.1) using Hensel's lemma so that

$$f \equiv h_1 \cdots h_r \pmod{p^k}. \tag{7.2}$$

Now the index k is chosen so that $p^k > 2\Lambda$.

To find the true factors of f from (7.2) we first try each of the h_i to see if it divides f exactly. Suppose that $h_1, \ldots, h_s$ do so. Then writing $g_i = h_i$ for

$1 \leq i \leq s$ we have

$$f = g_1 \cdots g_s f_2, \tag{7.3}$$

where

$$f_2 \equiv h_{s+1} \cdots h_r \pmod{p^k}. \tag{7.4}$$

Now each product formed from two of the $h_{s+1}, \ldots, h_r$ is tried as a factor of f. If this yields factors $g_{s+1}, \ldots, g_t$ then we have

$$f = g_1 \cdots g_s g_{s+1} \cdots g_t f_3, \tag{7.5}$$

where

$$f_3 \equiv h_{2t-s} \cdots h_r \pmod{p^k}, \tag{7.6}$$

and we have assumed that the h_i have been indexed in the most convenient way to start with. Now we try products of three of the remaining h_i and so on until we know that the product of any remaining h_i is an irreducible factor of f. We finally obtain

$$f = g_1 \cdots g_u,$$

where $g_1, \ldots, g_u$ are irreducible factors of f in $\mathbb{Z}[x]$.

The maximum number of factors h_i is n and in the worst case (when f is irreducible) we require $\binom{n}{r}$ tests to see if any product of r of the h_i is a true factor of f. Hence in this case we need in total $\Omega(2^n)$ tests. However Collins [14] has shown that subject to certain assumptions the root testing part will require only polynomial number of trials on average, so that the entire algorithm requires only polynomial time on average.

A number of optimisations to the algorithm have been proposed in [46] and [2]. For example one can carry out the modular factorisation by the Berlekamp-Hensel algorithm with several different primes and then use the smallest set of factors which arises.

One can attempt to use information from different modular factorisations in a more sophisticated way. For example if $f \in \mathbb{Z}[x]$ has degree four, is equal to the product of two quadratics modulo p, and equals the product of a linear factor and

a cubic modulo q, then f must in fact be irreducible in $\mathbb{Z}[x]$. However it is difficult to write code which uses such information in a systematic and helpful way.

We remark finally that the Chinese Remainder Theorem cannot be used to speed up the algorithm. To see why this is suppose, we find that

$$f = abc \pmod{p_1}$$

and

$$f = uvw \pmod{p_2}.$$

The difficulty in obtaining a factorisation modulo p_1p_2 is that we have no way of knowing whether to associate u with a, b, c or some product of these.

Example 7.2.1 Let

$$f(x) = x^3 + 9x^2 + 19x + 161 = (x^2 + 19)(x + 9)$$

in $\mathbb{Z}[x]$. Then

$$f(x) \equiv (x + 2)(x + 3)(x + 4) \pmod{7}$$

and

$$f(x) \equiv (x + 5)(x + 6)(x + 9) \pmod{11}.$$

There seems to be no way other than trial and error to discover that

$$\begin{aligned} x + 9 &\equiv x + 9 \pmod{11} \\ &\equiv x + 2 \pmod{7} \end{aligned}$$

is a true factor of $f(x)$ and that

$$\begin{aligned} x^2 + 19 &\equiv (x + 5)(x + 6) \pmod{11} \\ &\equiv (x + 3)(x + 4) \pmod{7} \end{aligned}$$

is irreducible in $\mathbb{Z}[x]$.

7.2.3 Bivariate and Multivariate Polynomials over $\mathbb{Z}$

For bivariate and multivariate polynomials the general strategy is to reduce the problem to one of univariate factorisation and we shall describe this for the bivariate case.

Let $f(y,x)$ be a given squarefree element of $\mathbb{Z}[y,x]$. The polynomial f is replaced by (f mod J) where J is the ideal $(y-a)$ and $a \in \mathbb{Z}$. Since the resulting polynomial $f(a,x)$ is in $\mathbb{Z}[x]$ it can be factorised using the univariate algorithm. Then we have

$$f(y,x) \equiv h_1 h_2 \cdots h_t \pmod{(y-a)}. \tag{7.7}$$

This factorisation is lifted using Hensel's method so that

$$f(y,x) \equiv h_1^{(k)} h_2^{(k)} \cdots h_t^{(k)} \pmod{(y-a)^k}. \tag{7.8}$$

Finally the true factors of $f(y,x)$ are determined by trial divisions. The resemblance to the univariate algorithm is evident. One difference is that the parameter k does not depend on a Landau-Mignotte type of equality because now the degree in y takes the place of the coefficient size and k is chosen so that $k+1 \geq \deg_y(f)$.

Notice the contrasts to Kaltofen's algorithm:

- here we do not work in an extension field; and
- because of the trial divisions needed the algorithm will require exponential time in the worst case.

Experience with the algorithm suggests that the trial divisions do not usually dominate the running time [60].

There are some difficulties which do not occur in the univariate case.

1. The integer a must be chosen so that $f(a,x)$ is squarefree. This can be ensured in a small number of trials.
2. If $|a|$ is not small there will be a blow up of coefficients going from $f(y,x)$ to $f(a,x)$ which will slow up the algorithm. Thus a should be chosen to make $|a|$ as small as possible while ensuring that $f(a,x)$ is squarefree.

3. The leading coefficient of $f(y,x)$ with respect to x will in general be a polynomial in y. Any attempt to make this an integer by a change of variables is likely to destroy any sparseness and is sure to increase the degree in x (see §2.3). Wang [60] has shown how the leading coefficient problem may be overcome. A description is also given in [15].

The extension of this algorithm to the multivariate case seems obvious and is described by Wang [60].

7.2.4 Univariate polynomials with coefficients in $\mathbb{Z}[\alpha]$

Introduction

The general idea of the method is similar to the previous ones in this chapter, namely to replace the original polynomial f with its image under a mapping from $\mathbb{Z}[\alpha]$ to a finite field, factorise the polynomial over the finite field, and use this information to recover the factors of f in $\mathbb{Z}[\alpha]$. Factorisation over $\mathbb{Z}[\alpha]$ turns out to be considerably harder than factorisation over $\mathbb{Z}$ for reasons that will be outlined. We have not attempted to describe the entire algorithm in detail.

Let us suppose that $f \in \mathbb{Z}[\alpha][x]$ is squarefree and that α satisfies a monic irreducible polynomial $F \in \mathbb{Z}[t]$ of degree m. Then by analogy with the algorithm in section 7.2.2 we look for factors of

$$f_p = (f \bmod p) \in \mathbb{Z}_p[\alpha_p][x] \tag{7.9}$$

where α_p is a root of the (monic) polynomial

$$F_p = (F \bmod p) \in \mathbb{Z}_p[x]. \tag{7.10}$$

The factors of f_p are raised by Hensel's Lemma so that we have

$$f \equiv g_1 g_2 \cdots g_r \pmod{p^k}, \tag{7.11}$$

and from this the factors of f are found by trial divisions.

This algorithm has two difficulties which we have yet to address.

1. The size of the parameter k in the Hensel lifting must be determined.

2. The polynomial F_p need not be irreducible in $\mathbb{Z}_p[t]$.

The size of k

The Landau-Mignotte inequality (3.11) allows us to use M as a bound for the coefficients of the factors of f as elements of $\mathbb{C}$. However in the present context what we need is a bound on the coefficients of a factor of $f \in \mathbb{Z}[\alpha][x]$, that is, a bound on certain elements of $\mathbb{Z}[\alpha]$. To describe this we recollect the idea of the *defect* from §2.8. If α is an algebraic integer then an element of $\mathbb{Z}[\alpha]$ can be expressed in the form $q(\alpha)/\delta$ where $q \in \mathbb{Z}[\alpha]$ and δ is an integer, the fractions being expressed in their lowest terms. The largest δ which is necessary for any integer in $\mathbb{Z}[\alpha]$ is called the defect, which we shall denote by d.

In [61] it is shown that if a typical factor coefficient has the form

$$\sum_{i=0}^{m-1} a_i \alpha^i$$

then

$$|a_i| < \frac{dBm!\|\alpha\|^{m(m-1)/2}}{\sqrt{|\operatorname{discr}(F)|}}, \tag{7.12}$$

where B is a constant depending on f. Here $\|\alpha\|$ is the *size* of α and is defined to be the maximum absolute value of any conjugate of α. The exponent k is chosen so that p^k is at least twice the bound appearing on the right of (7.12).

It turns out that it is not easy to calculate the defect d and it may even be hard to obtain a good estimate. An overestimate of d (and hence k) will result in unnecessary effort being spent on the Hensel lifting. Wang [59], in an example, uses a smaller coefficient bound than (7.12), but does not give a justification for it. A discussion of this point appears in [3].

The case when F_p factorises

We remarked earlier that a polynomial $F \in \mathbb{Z}[x]$ which is irreducible over $\mathbb{Z}$ may factorise modulo p for every prime p. We suppose that

$$F \equiv \Phi_1 \Phi_2 \cdots \Phi_\rho \pmod{p} \tag{7.13}$$

and we denote a root of Φ_i by $\alpha^{(i)}$. Then we can determine a factorisation of f modulo p in $\mathbb{Z}/(\Phi_i)$ and by Hensel lifting obtain

$$f \equiv f_1^{(i)} \cdots f_{r(i)}^{(i)} \pmod{p^k} \text{ in } \mathbb{Z}_{p^k}[\alpha^{(i)}][x] \text{ for } 1 \leq i \leq \rho. \tag{7.14}$$

The roots of f in $(1/d)\mathbb{Z}[\alpha][x]$ can be reconstructed using the Chinese Remainder Theorem. The difficulty remains that we do not know for the different values of i which factors in (7.14) correspond to the same factors of f. The number of trials required to find the true factors of f may therefore be very large.

A. K. Lenstra's Algorithm

An alternative method, based on lattices, has been proposed by A. K. Lenstra [36]. This method has the advantage that no special course of action need be taken in the case when F factorises modulo p. We need first the definition of the fundamental region of a lattice.

Definition 7.2.1 *Let $b_1, \ldots, b_m$ be a set of linearly independent vectors in $\mathbb{Z}^m$ and $L = \sum_{j=1}^m \mathbb{Z} b_j$ the lattice for which the b_j form a basis. The* fundamental region *of L is the set of vectors v in $\mathbb{R}^m$ of the form $v = \sum_{j=1}^m c_j b_j$ for which $-\frac{1}{2} \leq c_j < \frac{1}{2}$ for $1 \leq j \leq m$.*

The only vector which the fundamental region has in common with the lattice is the zero vector.

Let M be the $m \times m$ matrix whose j-th column is b_i. Then if $\tilde{u} \in \mathbb{R}^m$ has the form $\sum_{i=1}^m \tilde{\xi}_j b_j$ and $\tilde{\xi} = (\tilde{\xi}_1, \tilde{\xi}_2, \ldots, \tilde{\xi}_m)^T$ we can write $\tilde{u} = M\tilde{\xi}$. We recollect that for any $x \in \mathbb{R}$ the nearest integer function $\langle \cdot \rangle$ is given by

$$\langle x \rangle = \left\lfloor x + \frac{1}{2} \right\rfloor \in \mathbb{Z}.$$

It is clear that

$$-\frac{1}{2} \leq x - \langle x \rangle < \frac{1}{2}.$$

It follows that the vector

$$u = \sum_{j=1}^{m} \left(\tilde{\xi}_j - \langle \tilde{\xi}_j \rangle\right) b_j = \tilde{u} - M\langle \tilde{\xi} \rangle = \tilde{u} - M\langle M^{-1}\tilde{u} \rangle \tag{7.15}$$

lies in the fundamental region and is unique for a fixed $\tilde{u}$. The nearest integer function in (7.15) acts component-wise when applied to a vector or a matrix.

Lenstra's algorithm proceeds at first in a way similar to the others, except that the conditions imposed on the Hensel parameter k are different. Suppose that p has been chosen and that F factorises as shown in (7.13). We choose one of the factors, say Φ_1, which we can assume to be monic. Let Φ_1 have degree l and define the lattice L to be generated by the basis

$$\{p^k, p^k t, \ldots, p^k t^{l-1}, \Phi_1, \Phi_1 t, \ldots, \Phi_1 t^{m-l-1}\}.$$

L is the set of polynomials of degree less than m which divide by Φ_1 modulo p^k.

Lenstra shows that it is possible to choose k so that the fundamental region of L contains $f_i \pmod{p^k, \Phi_1}$ for all the factors f_i of f. Furthermore he also gives a formula for reconstructing trial factors in $\mathbb{Z}[\alpha][x]$ from their images in the fundamental region. The entire algorithm is as follows.

1. Find $d \in \mathbb{N}$ such that f and the factors of f lie in $(1/d)\mathbb{Z}[\alpha][x]$.

2. Choose a prime p so that it satisfies the following conditions:

   - p does not divide d;
   - F is squarefree modulo p;
   - f is squarefree modulo p and Φ_1.

3. Choose $B \in \mathbb{R}$ so that B/d bounds the size of the coefficients of the factors of f in $(1/d)\mathbb{Z}[\alpha]$.

4. Choose $C \in \mathbb{R}$ to exceed the orthogonal defect of L, and find the least k such that

$$\|F\|^{m-1}(2CB)^m < p^{kl}.$$

5. Find the complete factorisation of f modulo p^k and Φ_1:

$$f \equiv h_1 \cdots h_r \pmod{p^k, \Phi_1}.$$

6. Compute, for all subsets S of $\{1, \ldots, r\}$ and such that $\deg(\tilde{h}) \leq \lfloor n/2 \rfloor$,

$$\tilde{h} = (d \prod_{i \in S} h_i) \pmod{p^k, \Phi_1} = \sum_{i=0}^{\deg(\tilde{h})} \tilde{v}_i x^i,$$

and test whether

$$h = \frac{1}{d}\left(\sum_{i=0}^{\deg(h)} (\tilde{v}_i - M\langle M^{-1}\tilde{v}\rangle x^i\right) \in \frac{1}{d}\mathbb{Z}[\alpha][x] \tag{7.16}$$

is a factor of f in $(1/d)\mathbb{Z}[\alpha][x]$.

The fourth item is the one which ensures that the fundamental region of the lattice will have the images of the factors of f in it. The images of the trial factors are reconstructed using (7.16).

Chapter 8

Factorisation in quotient rings

8.1 Introduction

In this chapter we shall look for a way of deciding which elements of certain factor rings have factors, how these factors may be found, and the complexity of the processes involved.

Let p be an element of $\mathbb{Q}[x]$ which is primitive and consider the ring $\mathbb{Q}[x]/(p)$. Any element $\alpha \in \mathbb{Q}[x]/(p)$ may be uniquely represented as $a + (p)$ where a is either zero or a polynomial with $\deg(a) < \deg(p)$. When no confusion arises over which polynomial p is involved we shall write α as a. Let F be an element of $\mathbb{Q}[x]$. We shall write $(F \bmod p)$ as f and define the projection ϕ from $\mathbb{Q}[x]$ to $\mathbb{Q}[x]/(p)$ by

$$f = F\phi := (F \bmod p).$$

We can suppose that p factorises as

$$p = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}, \tag{8.1}$$

where each p_j is irreducible, each $e_j > 0$ and the p_j are distinct. Here and subsequently all polynomials have positive degree unless we indicate otherwise. Of course it is implicit that there is an effective algorithm for finding the factorisation of p, namely the LLL algorithm.

Denote by Q_r the ring $\mathbb{Q}[x]/(p_1^{e_1}) \times \cdots \times \mathbb{Q}[x]/(p_r^{e_r})$, where the $\times$ indicates direct product. To each element f of $\mathbb{Q}[x]/(p)$ there corresponds the unique element

$$f_j \equiv f \pmod{p_j^{e_j}}$$

in $\mathbb{Q}[x]/(p_j^{e_j})$. Conversely the Chinese Remainder Theorem asserts that to each r-tuple $(f_1, \ldots, f_r)$ in Q_r there is a unique element of $\mathbb{Q}[x]/(p)$. It is a consequence of the algebra of congruences that the mapping ϕ defined by $f\phi = (f_1, \ldots, f_r)$ is a ring homomorphism. Hence ϕ is an isomorphism, so $\mathbb{Q}[x]/(p)$ is isomorphic to Q_r, with the natural definitions of addition and multiplication in each ring. We shall be interested in factorisation in Q_r (i.e. in $\mathbb{Q}[x]/(p)$) and in particular its relation to factorisation in the component $\mathbb{Q}[x]/(p_j^{e_j})$.

8.2 The ring $\mathbb{Q}[x]/(p^e)$, where p is irreducible

This is a particular case of equation (8.1) in which $e_i = 0$ if $i > 1$ and for simplicity we shall write p for p_1 and e for e_1 in this section. If $e = 1$ then the ring is $\mathbb{Q}[x]/(p)$, for an irreducible p, and this is a field. Because all the non-zero elements of a field are units no element of $\mathbb{Q}[x]/(p)$ factorises, and we note that in this case we obtain no information about factorisation of $F \in \mathbb{Q}[x]$ by looking at $f = F\phi \in \mathbb{Q}[x]/(p)$.

Now assume that $e \geq 2$ and that f is an element of $\mathbb{Q}[x]/(p^e)$. The units of the ring $\mathbb{Q}[x]/(p^e)$ are the elements f for which there is another element a such that $af = 1$ or, regarding f and a as elements F and A in $\mathbb{Q}[x]$, such that $AF - 1$ lies in the ideal (p^e) of $\mathbb{Q}[x]$. This is equivalent to saying that for some $A, B \in \mathbb{Q}[x]$ we have $AF + Bp^e = 1$ or, writing C for Bp^{e-1}, that $AF + Cp = 1$. Thus f is a unit of $\mathbb{Q}[x]/(p^e)$ if and only if $\gcd(F, p) = 1$ in $\mathbb{Q}[x]$ and the elements f which are not units are those that are multiples of p (and hence zero divisors).

If f is not a unit then either $f = 0$ or

$$f = up^d, \text{ for } 1 \leq d < e, \tag{8.2}$$

where u is a unit. Since p is irreducible it follows that if $d = 1$ then f is irreducible and if $1 < d < e$ then any factorisation of f must be of the form

$$f = u_o f_1 \cdots f_d,$$

where $f_i = u_i p$ and the u_i are units.

The relation of factorisation of $F \in \mathbb{Q}[x]$ to $f = F\phi \in \mathbb{Q}[x]/(p^e)$ is as follows. An irreducible non-unit element f of $\mathbb{Q}[x]/(p^e)$ corresponds to an element F of $\mathbb{Q}[x]$ which has p as a factor of multiplicity one and a factorising element of $\mathbb{Q}[x]/(p^e)$ corresponds to an element of $\mathbb{Q}[x]$ which has p as a repeated factor. The multiplicity of this repeated factor is less than e if $f \neq 0$ and is greater that or equal to e if $f = 0$. No other information about factorisation in $\mathbb{Q}[x]$ can be obtained from $\mathbb{Q}[x]/(p^e)$.

To decide if an element f of $\mathbb{Q}[x]/(p^e)$ factorises it is sufficient to regard it as an element of $\mathbb{Q}[x]$ and find the largest power of p which will divide it exactly. The size $|f|$ of an element f in $\mathbb{Q}[x]/(p^e)$ is $|f|$, where f is regarded as an element of $\mathbb{Q}[x]$. Then the division of F by p in $\mathbb{Q}[x]$ can be carried out in a time which is polynomial in the quantities $\max(\deg(F), \deg(p))$ and $\max(\log|F|, \log|p|)$. If the complexity is regarded as a function of F only we have the following theorem.

Theorem 8.2.1 *Let $F \in \mathbb{Q}[x]$ and $F\phi = f$. Then using divisions of F by p in $\mathbb{Q}[x]$ we can determine whether f is a unit, an irreducible or a factorising element in a time which is polynomial in* $\deg(q)$ *and* $\log(|f|)$. *Furthermore the procedure finds factors of f when these exist.*

8.3 The ring $\mathbb{Q}[x]/(p)$ where p has distinct factors

Here $p = p_1^{e_1} \cdots p_r^{e_r}$ where $r \geq 2$ and each e_i is positive. Throughout this section f is an element of $\mathbb{Q}[x]/(p)$ and $f\phi = (f_1, \ldots, f_r)$ where $f_i \in \mathbb{Q}[x]/(p_i^{e_i})$.

Lemma 8.3.1 *An element $f \in \mathbb{Q}[x]/(p)$ is a unit if and only if each f_i in $f\phi$ is a unit of $\mathbb{Q}[x]/(p_i^{e_i})$*

Proof If f is a unit then there is an element a such that $af = 1$. Then since ϕ is an isomorphism we have

$$(1_1, \ldots, 1_r) = 1\phi = (af)\phi = (a\phi)(f\phi) = (a_1, \ldots, a_r)(f_1, \ldots, f_r) = (a_1 f_1, \ldots, a_r f_r).$$

Conversely if for each f_i there is an a_i such that $a_i f_i = 1$ we have

$$1 = (1,\ldots,1)\phi^{-1} = (a_1 f_1,\ldots,a_r f_r)\phi^{-1} = (a_1,\ldots,a_r)\phi^{-1}(f_1,\ldots,f_r)\phi^{-1} = af.$$

□

According to the discussion in the previous section, if f_i is not a unit then it divides by p_i and is either zero or a zero divisor. It follows from the definition of f_i that if f_i divides by p_i the so does f. Suppose $f \neq 0$ has the form $f = ug$ where u is a unit and

$$g = p_1^{n_1} p_2^{n_2} \cdots p_r^{n_r},$$

with at least one $n_i > 0$ so that f is not a unit. If $h = p/g$ then

$$fh = up = 0 \in \mathbb{Q}[x]/(p),$$

and we have the following corollary.

Corollary 8.3.1 *An element $f \neq 0$ is a zero divisor in $\mathbb{Q}[x]/(p)$ if and only if at least one f_j is either zero or a zero divisor in $\mathbb{Q}[x]/(p_j^{e_j})$.*

Thus an element of $\mathbb{Q}[x]/(p)$ is either a unit, zero or a divisor of zero. We turn our attention to the non-unit elements which are the only ones that factorise.

Consider first the case $p = p_1 p_2$ where p_1, p_2 are irreducible in $\mathbb{Q}[x]$. Suppose that $f = up_1$ for some unit u such that $f\phi = (0,1)$. Then since

$$(0,1)(0,1) = (0,1)$$

it follows from the fact that ϕ is an isomorphism that

$$f = up_1 = (up_1)(up_1) = ff.$$

This is a factorisation according to Definition 2.2.2, because f is a divisor of zero and so is not a unit.

In the general case if f is an element of $\mathbb{Q}[x]/(p)$ such that in $f\phi$ we have, say, $f_1 = 0$ then for any unit u_1

$$\begin{aligned} f\phi &= (0, f_2,\ldots,f_r) \\ &= (u_1, f_2,\ldots,f_r)(0,1,\ldots,1) \\ &= (u_1, f_2,\ldots,f_r)(0,1,\ldots,1)(0,1,\ldots,1). \end{aligned}$$

If follows that if any f_j is zero in $f\phi$ then $f\phi$ does not have a unique factorisation in Q_r and nor does f in $\mathbb{Q}[x]/(p)$. In particular the zero of $\mathbb{Q}[x]/(p)$ has a non-unique factorisation (as is bound to happen if we have zero-divisors).

Now consider the case when f is a divisor of zero but no f_i is zero. Suppose, for example, that f_1 and f_2 are divisors of zero while the other f_i are units. Then there are units $u_1 \dots u_r$ such that

$$f\phi = (u_1 p_1^{d_1}, u_2 p_2^{d_2}, u_3, \dots, u_r) \text{ for } 1 \leq d_k < e_k \text{ and } k = 1, 2.$$

Hence we have

$$f\phi = (u_1, \dots, u_r)(p_1^{d_1}, 1, 1, \dots, 1)(1, p_2^{d_2}, 1, \dots, 1) = ugh\phi, \tag{8.3}$$

say. We saw in §8.2 that $u_1 p_1^{d_1}$ and $u_2 p_2^{d_2}$ are unique factorisations of f_1 and f_2 in $\mathbb{Q}[x]/(p_1^{e_1})$ and $\mathbb{Q}[x]/(p_2^{e_2})$ respectively. It follows that (8.4) gives a unique factorisation of $f\phi$ (up to multiplication by units). The corresponding ugh is a unique factorisation of f. In the case when

$$f\phi = (u_1, \dots, u_r)(1, \dots, p_j, \dots, 1)$$

and p_j is the only non-unit amongst the f_i we observe that $f\phi$ is irreducible and that elements such as f are the irreducible elements of $\mathbb{Q}[x]/(p)$. The theorem below sums up the algebraic properties of factorisation in $\mathbb{Q}[x]/(p)$.

Theorem 8.3.1 *Let f be zero or a divisor of zero of $\mathbb{Q}[x]/(p)$ and in addition let $f\phi = (f_1, \dots, f_r) \in Q_r$.*

Either some f_j in $f\phi$ is zero, in which case $f\phi$ has infinitely many factors of the form

$$(1, \dots, 1, 0, 1, \dots, 1)$$

with zero in the j-th place; or no f_i is zero, in which case $f\phi$ can be factorised as a product of a unit $(u_1, \dots, u_r)$ and terms of the form $(1, \dots, 1, p_j, 1, \dots, 1)$, and this factorisation is unique up to the order of the factors. With appropriate changes of notation, this account of factorisation in Q_r also describes factorisation in $\mathbb{Q}[x]/(p)$ which is isomorphic to Q_r.

Turning to the complexity issues, let F and p be any polynomials in $\mathbb{Q}[x]$. Then the factors of p in $\mathbb{Q}[x]$ may be found by the LLL algorithm. The $f \in \mathbb{Q}[x]/(p)$ corresponding to F can be determined by division of F by p in $\mathbb{Q}[x]$. Finally the f_i in $f\phi$ can be determined from divisions of f by the p_i, regarded as elements of $\mathbb{Q}[x]$, and the time required for these divisions is polynomial in the degree of f and $\log(|f|)$. The last step will also furnish the information needed in Theorem 8.3.1. These remarks are formalised in the next theorem.

Theorem 8.3.2 *Let $f \in \mathbb{Q}[x]/(p)$ and suppose the complete factorisation of $p \in \mathbb{Q}[x]$ into a product $p_1^{e_1} \cdots p_r^{e_r}$ of powers of irreducibles in $\mathbb{Q}[x]$ is known. Then in a time which is polynomial in $\deg(f)$ and $\log(|f|)$ we can determine whether f is a unit, an irreducible, or a factorising element; and in the last case we can describe the factorisation of f in terms of Theorem 8.3.1.*

If $F \in \mathbb{Q}[x]$ corresponds to $f \in \mathbb{Q}[x]/(p)$ it is straightforward to rephrase some of the results in this section in terms of $\gcd(F,p)$. Without giving the details we note that $\gcd(F,p) = 1$ if and only if f is a unit, and knowledge of f and its factorisation gives no information about factors of F which are coprime with p.

Bibliography

Guide to the references

Chapter 2

There are many books which cover the basics of the algebraic background; the one by Fraleigh is typical [17]. A number of topics which are needed are not included however.

The Chinese Remainder Theorem, which underlies many of the algorithms studied, can be found in Aho, Hopcroft and Ullman [1] or in Knuth's *The Art of Computer Programming*, Volume 2 [32]. Both Berlekamp's book *Algebraic Coding Theory* [5] and [32] contain an explanation of Berlekamp's algorithm. For a more extensive coverage of polynomials over finite fields the book by Lidl and Niederreiter [41] can be recommended. The technique of Hensel lifting is outlined in the text by Davenport, Siret and Tournier [15]. Resultants are covered in the classic text by van der Waerden [57]; and a good account of subresultants can be found in the paper by Brown and Traub [8]. Field extensions are expounded in [17], but to find a treatment of ideas such as algebraic integers it is better to look at a book on algebraic number theory, such as the ones by Lang [34] or Marcus [43].

Chapter 3

The main reference here is obviously LLL [40]. For an independent account of lattices the standard reference is the book by Cassels [12]. A more modern reference is [50] which also gives an account of a reduced basis. Since the publication of LLL there has been some interest in reduced basis algorithms and improvements to the algorithm in LLL have been published by Schönhage and Schnorr [53], [54], [55], but these are not central to the result of LLL. The

LLL factorisation algorithm is not one that one would use in practice: a practical algorithm is described in Chapter 7.

Chapter 4

The ideas in this chapter were presented originally in two papers by Kaltofen, one showing that factorisation of a multivariate polynomial over the integers can be reduced to factorisation of a bivariate polynomial over the integers in polynomial time [25], and another giving the corresponding result for bivariate to univariate [26]. A unified account appears in [27]. This algorithm is theoretical, and not recommended for practical use.

Chapter 5

The three papers by A. K. Lenstra [37], [38] and [39] described in this chapter all assume familiarity with LLL. The proofs in each case are along the lines of LLL, but now rather more involved because the underlying fields are not so simple.

Chapter 6

The Cantor-Zassenhaus algorithm of [11] is described by Knuth in [32]. This algorithm is the only one described in the thesis which is randomised. There are earlier randomised algorithms by Berlekamp [6] and Rabin [51].

Chapter 7

The description of the modular gcd algorithm is based on the account in Davenport, Siret and Tournier [15], as is the algorithm for factorising in $\mathbb{Z}[x]$. The algorithm of Lenstra [36] for factorisation in $\mathbb{Z}[\alpha][x]$ has been subjected to trials by Abbott, Bradford and Davenport [2], [3] which are illuminating about the difficulties encountered and relative merits of this algorithm and the one by Wang [60].

References

In the references below volume n of the Springer-Verlag series of Lecture Notes in Computer Science is referred to as LNCS n.

1. A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms*, (1974) Addison-Wesley, Reading, Mass.

2. J. A. Abbott, R. J. Bradford, J. H. Davenport, *A remark on factorisation* SIGSAM Bulletin 19 (1985) 2, 31–33, 37.

3. J. A. Abbott, R. J. Bradford, J. H. Davenport, *Factorisation of Polynomials: Old Ideas and Recent Results* in *Trends in Computer Algebra*, (ed. R. Janssen) (1988) 81–91, LNCS 290.

4. E. R. Berlekamp, *Factoring Polynomials over Finite Fields*, Bell System Tech. J. 46 (1967) 1853–1859.

5. E. R. Berlekamp, *Algebraic Coding Theory*, (1968) McGraw-Hill, New York.

6. E. R. Berlekamp, *Factoring polynomials over large finite fields*, Math. Comp. 24 (1970) 713–735.

7. W. S. Brown, *On Euclid's algorithm and the computation of polynomial greatest common divisors*, JACM 18 (1971) 478–504.

8. W. S. Brown, J. S. Traub, *On Euclid's algorithm and the theory of subresultants*, JACM 18 (1970) 505–514.

9. B. Buchberger, G. E. Collins, G. Loos, (editors) *Computer Algebra: Symbolic Manipulation and Computation*, Springer Verlag 1983.

10. D. G. Cantor, *Irreducible Polynomials with integral coefficients have succinct certificates*, J. Algorithms 2 (1981) 385–392.

11. D. G. Cantor, H. Zassenhaus, *A new algorithm for factoring polynomials over finite fields*, Math. Comp. 36 (1981) 587–592.

12. J. W. S. Cassels, *Geometric Number Theory*, (1959) Springer Verlag, Berlin.

13. G. E. Collins, *Subresultants and reduced polynomial remainder sequences*, JACM 14 (1967) 128–142.

14. G. E. Collins, *Factoring univariate polynomials in polynomial average time*, Proceedings of EUROSAM 79, 317–329, LNCS 72.

15. J. H. Davenport, Y. Siret, E. Tournier, *Computer Algebra: Systems and Algorithms for Algebraic Computation*, (1988) Academic Press, London and New York.

16. U. Finke, M Pohst, *Inproved methods for calculating vectors of short length in a lattice, including complexity analysis*, Math. Comp. 44 (1985) 463–471.

17. J. A. Fraleigh, *An Introduction to Abstract Algebra*, 4th edition, (1982) Addison Wesley, Reading, Mass.

18. J. von zur Gathen, *Factoring Sparse Multivariate Polynomials*, Proceedings of the 24th IEEE Annual Symposium on the Foundations of Computer Science (1983) 463–471.

19. J. von zur Gathen, *Hensel and Newton Methods in Valuation Rings*, Math. Comp. 42 (1984) 637–661.

20. J. von zur Gathen, *Parallel Algorithms for Algebraic Problems*, SIAM J. Comp. 13 (1984) 802–824.

21. J. von zur Gathen, *Irreducibility of Multivariate Polynomials*, J. Comp. Sys. Sci. 31 (1985) 256–287.

22. J. von zur Gathen, E. Kaltofen, *Factoring sparse multivariate polynomials* , J. Comp. Sys. Sci. 31 (1985) 256–287.

23. J. von zur Gathen, E. Kaltofen, *Factorisation of multivariate polynomials over finite fields*, Math. Comp. 45 (1985) 251–261.

24. K. Hensel, *Theorie der Algebraischen Zahlen*, (1908) Teubner, Leipzig.

25. E. Kaltofen, *A polynomial-time reduction from multivariate to bivariate polynomial factorisation*, Proceedings of the Annual A. C. M. Symposium on the Theory of Computing (1982) 261–266.

26. E. Kaltofen, *A polynomial-time reduction from bivariate to univariate integral polynomial factorisation*, Proceedings of the 23rd Annual IEEE Symposium on the Foundations of Computer Science (1982) 57–64.

27. E. Kaltofen, *Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorisation*, SIAM J. Comp. 14 (1985) 469–489.

28. E. Kaltofen, *Effective Hilbert irreducibility*, Information and Control 66 (1985) 123–137.

29. E. Kaltofen, *Computing with polynomials given by straight-line programs I: greatest common divisors*, Proceedings of the 17th Annual ACM Symposium on the Theory of Computer Science (1985) 131–142.

30. E. Kaltofen, *Computing with polynomials given by straight-line programs II: sparse factorisation*, Proceedings of the 26th Annual IEEE Symposium on the Foundations of Computer Science (1985) 451–457.

31. E. Kaltofen, *Computer Algebra Algorithms*, in *Annual Review of Computer Science* (1987) 91–118.

32. D. E. Knuth, *The Art of Computer Programming, Vol2: Seminumerical Algorithms*, 2nd edtion, (1981) Addison Welsey, Reading, Ma.

33. S. Landau, *Factoring polynomials over algebraic number fields*, SIAM J. Comput. 14 (1985) 184–195.

34. S. Lang, *Algebraic Number Fields*, (1970), Addison-Wesley,Reading,Ma.

35. A. K. Lenstra, *Lattices and the factorisation of polynomials*, SIGSAM Bulletin 15, 3 (1981) 15–16.

36. A. K. Lenstra, *Factoring polynomials over algebraic number fields*, EuroCAM'82 (ed. J. Calmet) LNCS 144.

37. A. K. Lenstra, *Factoring multivariate integral polynomials*, Th. Comp. Sci. 34 (1984) 207–213.

38. A. K. Lenstra, *Factoring multivariate polynomials over finite fields*, J. Comp. Sys. Sci. 30 (1985) 235–248.

39. A. K. Lenstra, *Factoring multivariate polynomials over algebraic number fields*, SIAM J. Comput. 16 (1987) 591–598.

40. A. K. Lenstra, H. W. Lenstra jr., L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. 261 (1982) 515–534.

41. R. Lidl, H. Niederreiter, *Introduction to finite fields and their applications*, (1986) Cambridge University Press.

42. K. Mahler, *An analogue of Minkowski's geometry of numbers in a field of series*, Ann. of Math. 42 (1941) 488–522.

43. D. A. Marcus, *Number Fields*, (1977) Springer Verlag, New York-Heidelberg-Berlin.

44. M. Mignotte, *An inequality about factors of polynomials*, Math. Comp. 28 (1974) 1152–1157.

45. H. Minkowski, *Geometrie der Zahlen*, (1908, reprinted 1953) Chelsea, New York.

46. D. R. Musser, *Multivariate polynomial factorisation*, JACM 22 (1975) 291–308.

47. D. A. Plaisted, *Sparse complex polynomials and polynomial reducibility*, J. Comp. Sys. Sci. 14 (1977) 210–221.

48. M. Pohst, *On the computation of lattice vectors of minimal length, successive minima and reduced bases, with applications*, SIGSAM Bulletin 15, 1 (1981) 37–44.

49. M. Pohst, *A modification of the LLL reduction algorithm*, J. Symbolic Computation 4 (1987) 123–127.

50. M. Pohst, H. Zassenhaus, *Algorithmic Algebraic Number Theory*, (1989) Cambridge University Press.

51. M. Rabin, *Probabilistic algorithnms in finite fields*, SIAM J. Comput. 9 (1980) 273–280.

52. M. Rothstein, H. Zassenhaus, *Deterministic Factorisation method of polynomials in one variable over finite fields*, Seminar given at RISC-Linz Summer School in Computer Algebra, July 1990.

53. A. Schönhage, *Factorisation of univariate polynomials by diophantine approximation and an improved basis reduction algorithm*, Proceedings of the 11th International Conference on Automata, Languages and Programmming (9184), LNCS 182,

54. C. P. Schnorr, *A hierarchy of polynomial-time lattice basis reduction algorithms*, Th. Comp. Sci. 53 (1987) 201–204.

55. C. P. Schnorr, *A more efficient lattice reduction algorithm*, J. of Algorithms 9 (1988) 47–62.

56. B. M. Trager, *Algebraic Factoring and rational function integration*, Proceedings of the ACM Symposium on Algebraic Computation (1976) 219–226.

57. B. L. van der Waerden, *Modern Algebra*, (1929) Springer Verlag, Berlin.

58. P. S. Wang, L. Rothschild, *Factoring multivariate polynomials over the integers*, Math. Comp. 29 (1975) 935–950.

59. P. S. Wang, *Factoring multivariate polynomials over algebraic number fields*, Math. Comp. 30 (1976) 324–336.

60. P. S. Wang, *An improved multivariate polynomial factoring algorithm*, Math. Comp. 32 (1978) 1215–1231.

61. P. J. Weinberger, L. P. Rothschild, *Factoring polynomials over algebraic number fields*, ACM Trans. on Mathematical Software 2 (1976) 335–350.

62. H. Zassenhaus, *On Hensel Factorisation I*, J. Number Theory 1 (1969) 291–331.