The University of Warwick

THEORY OF COMPUTATION

REPORT NO. 47

ALGORITHMS IN THE THEORY OF INTEGRAL BINARY QUADRATIC FORMS

BY

CONSTANTINOS S. ILIOPOULOS

Department of Computer Science
University of Warwick
COVENTRY CV4 7AL
ENGLAND.

September 1981

CONSTANTINOS S. ILIOPOULOS

B.Sc. (Athens)

ALGORITHMS IN THE THEORY OF INTEGRAL BINARY QUADRATIC FORMS

M.Sc. THESIS

University of Warwick

Department of Computer Science

September, 1981.

ΑΦΙΕΡΩΜΕΝΟ ΣΤΟΥΣ

Σπύρο και Δέσποινα

DEDICATED TO

Spiros and Despina

CONTENTS

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Meurig Beynon for the large amount of time and effort which he has spent in helping and advising me in my research work. In particular, I would like to thank him for the basis procedure (Algorithm 4.1.19). I would also like to thank Prof. Mike Paterson for many helpful conversations and in particular for his idea of the iterative version of Lenstra's algorithm (p. 99-100). Moreover I have to thank my girl-friend Jackie Daykin for her encouragement and help in correcting my faulty English.

Finally, I have to thank my parents for their continuous support, love, interest and encouragement during my whole student career - this work belongs to them.

INTRODUCTION

The classical ("old fashioned") theory of integral binary quadratic forms was developed by Gauss in his famous "*Disquisitiones Arithmeticae*". This thesis is concerned with the formal description and analysis of a selection of algorithms which make use of such forms.

For simplicity, we have adopted the same convention as Gauss, who considered only binary quadratic forms having an even middle coefficient. There is a more general theory permitting odd middle coefficient due to Dirichlet, and all the algorithms described and analysed in this thesis can be easily generalized to Dirichlet's forms.

The fact that the equivalence classes of binary quadratic forms of fixed non-square determinant D form a finite abelian group $Cl(D)$ was a fundamental discovery due to Gauss (? - see [13], p. 15 and [25], p. 171!). Most of the algorithms we describe are connected directly or indirectly with the "structure" and "infrastructure" (that is the structure of a class of $Cl(D)$) of form class groups. Apart from their intrinsic interest, these algorithms have applications in connection with orders in quadratic fields and their associated ideal class groups, and with the problem of factorization. Applications of algebraic number theory are briefly sketched where appropriate, but detailed reference is outside the scope of this thesis, since our main interest is in the theory of computation and the use of binary quadratic forms as a computational tool.

The content of the six chapters of the thesis is briefly summarised below.

Chapter 0 comprises basic definitions, notations and algorithms together with reduction procedures for binary quadratic forms due to Gauss and Lagarias.

Chapter 1 deals with composition of forms; methods of composition due to Gauss and Dirichlet are described and algorithms based on these methods are constructed. The group $Cl(D)$ is defined and the relationship between $Cl(D)$ and an appropriate ideal class group is also described.

Chapter 2 deals with the genus characters $Cl(D)$. Algorithms for computing a basis for genus characters and deciding whether or not a form belongs to the principal genus (or equivalently is a square in $Cl(D)$) are presented.

Chapter 3 is concerned with algorithms for deciding equivalence of forms. This decision problem is particularly interesting for forms of positive non-quadratic determinant, and recent work of Lenstra and Shanks on relatively efficient algorithms is described. Number-theoretic applications include the computation of regulator and fundamental unit of real quadratic fields and the related problems of solving Pellian and non-Pellian equation.

Chapter 4 deals with methods for computing the class number (the order of $Cl(D)$). Gauss' counting methods are described and analysed and the use of Dirichlet's formulae is considered. A refined and improved version of an $O(|D|^{1/5+\varepsilon})$ algorithm due to Shanks is described and analysed.

The computation of class-numbers is considered in conjunction with the computation of the class group structure (see algorithms 4.1.13, 4.1.19).

In Chapter 5 Shanks' factorization algorithm CLASSNO, which depends upon the relationship between factorization of D and the group structure of $C\ell(D)$, is described.

For the most part the results described are closely based on the work of Gauss, Shanks, Lagarias and Lenstra. The main problem has been the collation and interpretation of results from many sources, and the most original aspects of the work are refinements of algorithms and their analyses (see e.g. theorems 1.1.6 - 1.1.8, algorithm 2.1.9, theorem 4.1.7 (and its applications) etc.)

A note on the presentation of algorithms

In general the format of presentation of an algorithm is (i) informal description, (ii) formal description (iii) correctness, (iv) analysis. In our formal description of the algorithms we use a version of the language PIDGIN ALGOL given by Aho Hopcroft and Ullman in [2]. In our analysis of the algorithms, we measure always the worst-case time complexity (see [2]).

C.S.I.

Coventry

September 1981.

0.1. BASIC DEFINITIONS-NOTATIONS

DEFINITIONS 0.1.1.

A *binary quadratic form* (abbreviated binary form or simply form) Q is

$$Q(x,y) = ax^2 + 2bxy + cy^2 \text{ with } a,b,c, \text{ integers}$$

and will be denoted as $Q = (a,b,c)$.

The *coefficient matrix* of a form $Q = (a,b,c)$ is the symmetric matrix

$$M_Q = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

Determinant D of a form $Q = (a,b,c)$ is the integer

$$D = b^2-ac = - \det (M_Q) \qquad (0.1)$$

Two forms $Q = (a,b,c)$, $F = (A,B,C)$ are called *opposite*, if

$$a = A, \quad b = -B, \quad c = C.$$

The opposite form of a form Q is denoted by $\bar{Q}$.

The forms with determinant $D < 0$ are called *definite*, the forms with determinant $D > 0$ and D non-square are called *indefinite* and the forms with determinant $D \geq 0$ perfect square are called *degenerate*.

A form $Q = (a,b,c)$ is called *properly primitive*, if $\gcd(a,2b,c) = 1$.

A definite form $Q = (a,b,c)$ is called *positive*, if $a > 0$.

An integer N is *represented* by a form Q, if there exist integers x_1, x_2, such that

$$Q(x_1,x_2) = N \text{ and } \gcd(x_1,x_2) = 1.$$

A form $Q = (a,b,c)$ is called *ambiguous*, if $a \mid 2b$.

DEFINITION 0.1.2.

A form $Q_1 = (a_1,b_1,c_1)$ is *equivalent* to a form $Q_2 = (a_2,b_2,c_2)$, if there exists a 2×2 unimodular matrix S with integer entries such that

$$M_{Q_2} = S^T M_{Q_1} S \tag{0.2}$$

where S^T is the transpose matrix of S.
If (0.2) holds, we shall say that Q_1 is transformed to Q_2 and denote $Q_1 \to Q_2$ via S.

PROPOSITION 0.1.3.

Suppose S is a 2×2 matrix.

(i) If $Q_1 \to Q_2$ via S, then Q_1 and Q_2 have the same determinant

(ii) If $Q_1 \to Q_2$ via S, then there exists a 2×2 unimodular matrix S'.

Proof

(i) From (0.2) we have

$$\det(M_{Q_2}) = \det(S^T) \cdot \det(M_{Q_1}) \cdot \det(S) = \det(M_{Q_1})$$

and from (0.1) the result follows.

(ii) It can be shown for $S' = S^{-1}$ that $Q_2 \to Q_1$ via S'. □

Proposition 0.1.3 shows that the relation of being equivalent is an equivalence relation. We write $Q_1 \approx Q_2$ to denote that Q_1, Q_2 are equivalent.

PROPOSITION 0.1.4.

If $Q_1 \approx Q_2$ and Q_1 is properly primitive, then Q_2 is properly primitive.

Proof

Let $Q_1 = (a_1,b_1,c_1) \approx Q_2 = (a_2,b_2,c_2)$. Then there exists a matrix

$$S = \begin{pmatrix} \kappa & \lambda \\ \mu & \nu \end{pmatrix} \quad \text{with } \kappa\nu - \lambda\mu = 1$$

and

$$M_{Q_1} = S^T M_{Q_2} S$$

Then
$$\begin{aligned} a_1 &= a_2\kappa^2 + 2b_2\kappa\mu + c_2\mu^2 \\ 2b_1 &= 2a_2\kappa\lambda + 2b_2(\kappa\nu + \lambda\mu) + 2c_2\mu\nu \\ c_1 &= a_2\lambda^2 + 2b_2\lambda\nu + c_2\nu^2 \end{aligned} \tag{0.3}$$

If there exist p such that $p|a_2$, $p|2b_2$, $p|a_2$, then by (0.3) $p|a_1$, $p|2b_1$, $p|c_1$. Since Q_1 is properly primitive, it follows that $p = 1$. □

PROPOSITION 0.1.5.

If $Q_1 \approx Q_2$, then Q_1 and Q_2 represent the same integers.

Proof

Suppose that $Q_2 \to Q_1$ via $S = \begin{pmatrix} \kappa & \lambda \\ \mu & \nu \end{pmatrix}$.

Then if M is represented by Q_1 and $Q_1(m,n) = M$, then one can see that $Q_2(\kappa m + \lambda n, \mu m + \nu n) = M$. □

NOTATION 0.1.6.

For a $n \times n$ matrix $M = (m_{ij})$, $||M||$ denotes $\max_{i,j} \{|m_{ij}|\}$

If p is prime, we write $p^k||a$ to denote that $p^k|a$ and $p^{k+1} \nmid a$.

If Q is a form, then $||Q||$ denotes $||M_Q||$.

We use *log* m to denote the function defined by

$$\log m = \begin{cases} \log_2 m & \text{if } m > 4 \\ 2 & \text{if } m < 4 \end{cases}$$

The *natural logarithm* of a number m is denoted by $\ell n(m)$

When a form Q with determinant D is denoted by $Q = (a,b,*)$, then * indicates the integer $(b^2-D)/a$.

PROPOSITION 0.1.7.

If M,N are two n x n matrices, then

$$||MN|| < n^2 \, ||M|| \, ||N||. \qquad \square$$

REMARK

There is a more general theory of forms (introduced by Dirichlet) which permits odd middle coefficient. A form

$$Q(x,y) = ax^2 + bxy + cy^2$$

has *discriminant*

$$\Delta = b^2 - 4ac.$$

One can see that the Gaussian forms are a subset of the set of the forms above. Moreover if a form Q

$$Q(x,y) = ax^2 + 2bxy + cy^2$$

has determinant D, then in Dirichlet's terminology it has discriminant

$$\Delta = (2b)^2 - 4ac = 4D.$$

0.2. SOME COMPLEXITY BOUNDS

The time complexity of an algorithm is measured in terms of *elementary operations*. An elementary operation is a Boolean operation on a bit or pair of bits.

A function $f(x)$ is said to be $O(g(x))$ if there is a positive constant c such that $f(x) \leq cg(x)$ for every x.

Some preliminary complexity bounds for elementary algorithms are given below.

THEOREM 0.2.1. (Schönhage and Strassen)

Two integers of length n (in binary bits) can be multiplied in $M(n)$ elementary operations, where

$$M(n) = cn \log n \log \log n$$

and c is a sufficiently large constant. □

THEOREM 0.2.2 (Cook)

To divide an integer u of length n by an integer v of length at most n to find

$$u = qv + r \quad 0 \leq r < u$$

requires $O(M(n))$ elementary operations. □

The following theorem yields a bound for an algorithm given by Knuth [19], analyzed by Schönhage [32].

THEOREM 0.2.3. (Extended Euclidean algorithm)

There exists an algorithm which finds the gcd of two integers k,m of length at most n bits and yields integers λ, μ such that

$$\lambda k + \mu m = r$$

with $|\lambda| \leq \frac{m}{2}$, $|\mu| \leq \frac{k}{2}$ and $r = \gcd(k,m)$

in $O(M(n)\log n)$ elementary operations. □

The computation of gcd (k,m) (Euclidean algorithm) requires the same time as the algorithm above.

THEOREM 0.2.4

Suppose that m,k are integers, and b is an integer with $0 \leq b < m$. Then an integer x such that

$$x \equiv b^k \pmod{m} \text{ and } 0 \leq x < m$$

can be found in $O(M(\log m) \log k)$ elementary operations.

Proof

See Lagarias [20], p. 150. □

The following theorem yields a bound for an algorithm given by Shanks [35] based on a method of Tonelli (see Dickson [14] I, p. 215) and analyzed by Adleman, Manders and Miller [1].

THEOREM 0.2.5

Suppose that a complete factorization of an integer m and a quadratic non-residue n_i for each prime p_i dividing m are given and b is an integer with $0 \leq b < m$. Then it is possible to decide whether the equation

$$x^2 = b \pmod{m} \quad 0 \leq x < m$$

has a solution and find a solution if appropriate in $O(M(\log m)(\log m)^2)$ elementary operations. □

The following theorem assumes the truth of the Generalized Riemann Hypothesis (G.R.H.) (see Chapter 2) and it is based on the result of Aukeny [3] for the least quadratic non-residue. Also see Lagarias [20] p. 152.

THEOREM 0.2.6.

If the Generalized Riemann Hypothesis is true then for a prime p a quadratic n (mod p) can be found in $O(\log^3 pM(\log p))$ elementary operations.

Proof

Ankeny proved that the least quadratic non-residue n_p (mod p) assuming G.R.H. is

$$n_p = O(\log^2 p).$$

Now from Euler's criterion, k is a quadratic non-residue mod p iff

$$k^{(p-1)/2} \equiv -1 \bmod p. \qquad (0.4)$$

Hence by testing for $1 < k < n_p$ the condition (0.4), a quadratic non-residue n(mod p) will be found in $O(\log^3 pM(\log p))$ elementary operations, since the cost of one testing is $O(\log p\, M(\log p))$ elementary operations from Proposition 0.2.4. □

Now an analogue of Theorem 0.2.6 is given without assumption of unproved Hypotheses, making use of the result of Burgess [10].

THEOREM 0.2.7.

Given a prime p, a quadratic non-residue n (mod p) can be found in $O(p^{1/4+\varepsilon})$ elementary operations.

Proof

Burgess proved that the least quadratic non-residue n_p (mod p) is

$$n_p = O(p^{1/4+\varepsilon}).$$

Hence as in theorem 0.2.6, it can be shown that a quadratic non-residue n (mod p) is found in $O(p^{1/4+\varepsilon})$ elementary operations. □

0.3. REDUCTION OF BINARY QUADRATIC FORMS

A. DEFINITE FORMS

DEFINITION 0.3.1.

A binary quadratic form $Q = (a,b,c)$ with determinant $D < 0$ is *reduced* if

$$|2b| \leq |a| \leq |c| \tag{0.5}$$

which implies

$$|2b| \leq |a| \leq \sqrt{\tfrac{4}{3}\,|D|} \tag{0.6}$$

The definition of a reduced definite form Q, implies

$$\|Q\| < |D| \tag{0.7}$$

The following algorithm was given by Gauss; given a definite form Q, it finds a reduced form Q' equivalent to Q. It applies a series of transformations $Q = Q_o \to Q_1 \to Q_2 \to \ldots \to Q_k$, where $Q_{i-1} \to Q_i$ via S_i until a reduced form Q_k is found. If $Q_{i-1} = (a_{i-1}, b_{i-1}, c_{i-1})$, then the matrix S_i is given by

$$\begin{pmatrix} 0 & 1 \\ -1 & \lambda_i \end{pmatrix}$$

where the unique λ_i is selected by the *Gauss rule*

$$|-b_{i-1} - \lambda_i\, c_{i-1}| \leq |c_{i-1}|/2 \tag{0.8}$$

ALGORITHM 0.3.2

INPUT : A definite form $Q = Q_o = (a_o, b_o, c_o)$ with determinant D.

OUTPUT : A definite reduced form F with determinant D, equivalent to Q and a unimodular matrix S such that

$$Q \rightarrow F \text{ via } S$$

Begin

$i \leftarrow 0$; $S \leftarrow$ <2 × 2 identity matrix>;

while <Q_i is not reduced> do

Begin

$i \leftarrow i+1$:

<choose a λ_i: $|-b_{i-1}-\lambda_i c_{i-1}| < |c_{i-1}|/2$> ;

$a_i \leftarrow c_{i-1}$;

$b_i \leftarrow -b_{i-1}-\lambda_i c_{i-1}$;

$c_i \leftarrow (b_i^2-D)/2$;

$Q_i \leftarrow (a_i, b_i, c_i)$;

$$S \leftarrow S \cdot \begin{pmatrix} 0 & 1 \\ -1 & \lambda_i \end{pmatrix};$$

end

Return $F = Q_i$;

end

THEOREM 0.3.3.

Algorithm 0.3.2 terminates and correctly yields a reduced form F, which is equivalent to Q, and a unimodular matrix S such that

$$Q \to F \text{ via } S.$$

Proof

Let $a_0, a_1, a_2, \ldots, a_i$ be the sequence of a_i's. Then there exists an integer k such that

$$|a_{i+1}| < |a_i| \quad \text{for } 0 < i < k$$

and

$$|a_k| < |a_{k+1}|. \tag{0.9}$$

Since if $|a_{i+1}| < |a_i|$ for every i, then we would have an infinite sequence of decreasing positive integers.

Now it will be proved that (a_k, b_k, c_k) is reduced. From Gauss rule we have

$$|b_k| = |-b_{k-1} - \lambda_i c_{k-1}| < |c_{k-1}|/2 = |a_k|/2$$

which implies

$$|2b_k| < |a_k| \tag{0.10}$$

From (0.9), we have

$$|a_k| < |a_{k+1}| = |c_k|. \tag{0.11}$$

Hence from (0.10), (0.11) $Q_k = (a_k, b_k, c_k)$ is reduced.

Now since the transformations $Q_{i-1} \to Q_i$ are via the unimodular matrix S_i, easily

$$S = \prod_{i=1}^{k} S_i \quad \text{is unimodular and}$$

$$Q \to Q_k = F \quad \text{via } S. \quad \square$$

The following theorem gives the complexity analysis of the above algorithm, following Lagarias ([20], p. 158).

THEOREM 0.3.4.

Algorithm 0.3.2 terminates in $O(M(\log \|Q\|) \log \|Q\|)$ elementary operations and $\log \|S\| = O(\log \|Q\|)$. $\square$

B. INDEFINITE FORMS.

DEFINITION 0.3.5.

A form $Q = (a,b,c)$ with determinant $D > 0$, non-square, is *reduced* if

$$|\sqrt{D} - |a|| < b < \sqrt{D} \tag{0.12}$$

The definition of a reduced indefinite form Q, implies

$$\|Q\| < 2\sqrt{|D|} \tag{0.13}$$

A reduced form $Q = (a,b,c)$ is *strictly reduced* if

$$|a| < \sqrt{|D|} \tag{0.14}$$

The following algorithm was given by Gauss ([16], A.181); given an indefinite form Q, it finds a reduced form Q' equivalent to Q. It applies a series of transformations $Q = Q_o \to Q_1 \to Q_2 \to \ldots \to Q_k$, where $Q_{i-1} \to Q_i$ until to find a reduced form Q_k. If $Q_{i-1} = (a_{i-1}, b_{i-1}, c_{i-1})$, then the matrix S_i is given by

$$\begin{pmatrix} 0 & 1 \\ -1 & \lambda_i \end{pmatrix}$$

where the unique λ_i is an integer selected by *Gauss rule*

$$\sqrt{D} - |c_{i-1}| < -b_{i-1} - \lambda_i c_{i-1} < \sqrt{D} \qquad (0.15)$$

The above algorithm is not efficient. Lagarias ([20], p. 154) found examples of infinite sequence of indefinite forms Q, for which the above Gauss' reduction procedure requires $\|Q\|^{1/4}$ transformations to find a reduced form equivalent to Q.

Moreover Lagarias ([20], p. 154) gave a reduction algorithm, which does a series of transformations

$$Q = Q_o \to Q_1 \to Q_2 \to \ldots \to Q_k = (a_k, b_k, c_k)$$

where

$$Q_{i-1} \to Q_i \text{ via } S_i \quad i = 1,2,\ldots .$$

If $Q_{i-1} = (a_{i-1}, b_{i-1}, c_{i-1})$, then the matrix S_i is given by

$$\begin{pmatrix} 0 & 1 \\ -1 & \lambda_i \end{pmatrix}$$

where λ_i is an integer selected by the *modified rule*

$$-|c_{i-1}|/2 < -b_{i-1} - \lambda_i c_{i-1} \leq |c_{i-1}|/2 \qquad (0.16)$$

The algorithm transforms a Q_i to Q_{i+1} until for some $k > 0$, a form Q_k such $|c_k| < 2\sqrt{D}$ is found. Lagarias proved that the form Q_{k+1} or Q_{k+2} is strictly reduced, where Q_{k+1}, Q_{k+2} are computed in the following way:

$$Q_k \to Q_{k+1} \text{ via } S_{k+1},\ Q_{k+1} \to Q_{k+2} \text{ via } S_{k+2}$$

where $S_{k+j} = \begin{pmatrix} 0 & 1 \\ -1 & k+j \end{pmatrix}$ for $j = 1,2$

and λ_{k+j} for $j = 1,2$ is selected by Gauss rule (0.15).

ALGORITHM 0.3.6.

INPUT : An indefinite form $Q = Q_o = (a_o, b_o, c_o)$ with determinant D.

OUTPUT : An indefinite form F, equivalent to Q, and a unimodular matrix S such that

$Q \to F$ via S.

Begin

$i \leftarrow 0$: $S \leftarrow$ <the 2×2 identity matrix>;

while <Q_i is not strictly reduced> do

Begin

If $c_i > 2\sqrt{D}$ then

<choose an integer λ_{i+1}: $-|c_i| / 2 < -b_i - \lambda_{i+1} c_i < |c_i|/2$>;

If $c_i < 2\sqrt{D}$ then

<choose an integer λ_{i+1}: $\sqrt{D} - |c_i| < -b_i - \lambda_{i+1} c_i < \sqrt{D}$>;

$i \leftarrow i+1$;

$a_i \leftarrow c_{i-1}$;

$b_i \leftarrow -b_{i-1} - \lambda_i c_{i-1}$;

$c_i \leftarrow (b_i^2 - D)/a_i$;

$Q_i \leftarrow (a_i, b_i, c_i)$;

$S \leftarrow S \cdot \begin{pmatrix} 0 & 1 \\ -1 & \lambda_i \end{pmatrix}$;

and

Return $Q_i = F$, S;

end

THEOREM 0.3.7.

Algorithm 0.3.6 correctly yields a strictly reduced form F, equivalent to Q and a unimodular matrix S such that $Q \to F$ via S; it terminates in $O(\log \|Q\| \, M(\log \|Q\|))$ elementary operations and

$$\log \|S\| = O(\log \|Q\|)$$

Proof

See Lagarias [20] p. 154. □

C. DEGENERATE FORMS

DEFINITION 0.3.8.

A form (a,b,c) with determinant $D = h^2$, $h > 0$ is reduced if

$$\begin{aligned} &\text{(i)} \quad 0 < a < 2h-1 \\ &\text{(ii)} \quad b = h \\ &\text{(iii)} \quad c = 0 \end{aligned} \tag{0.17}$$

Gauss ([16], A.206) gave a reduction procedure for degenerate forms. It reduces a degenerate form $Q=(a,b,c)$ using two transformations

$$Q \to Q_1 \text{ via } S_1 \text{ and } Q_1 \to Q_2 \text{ via } S_2$$

where $S_1 = \begin{pmatrix} \kappa & \lambda \\ \mu & \nu \end{pmatrix}$ unimodular, with κ,μ coprime, satisfying $\lambda a = \nu(h-b)$ and $S_2 = \begin{pmatrix} 1 & 0 \\ m & 1 \end{pmatrix}$ where m is an integer which satisfies

$$0 < 2mh + a_1 < 2h-1 \quad \text{where } Q_1 = (a_1,b_1,c_1).$$

ALGORITHM 0.3.9.

INPUT : A degenerate form $Q = (a,b,c)$ with determinant $D=h^2$

OUTPUT : A reduced form Q_2, equivalent to Q, and a unimodular matrix S, such that $Q \to Q_2$ via S.

Begin

$\nu \leftarrow a/\gcd(b-h,a)$; (0.18)

$\lambda \leftarrow (h-b)/\gcd(b-h,a)$; (0.19)

<Find κ,μ: $\kappa\nu-\lambda\mu = 1$, $|k| < |\lambda|$, $|\mu| < |\nu|$>;

Comment Use Extended Euclidean Algorithm (Theorem 0.2.3)

$$S_1 \leftarrow \begin{pmatrix} \kappa & \lambda \\ \mu & \nu \end{pmatrix}$$

$M_{Q_1} \leftarrow S_1^T M_Q S_1$;

<choose an m : $0 \leq 2mh + a_1 \leq 2h-1$>; (0.20)

$$S_2 \leftarrow \begin{pmatrix} 1 & 0 \\ m & 1 \end{pmatrix};$$

$M_{Q_2} \leftarrow S_2^T M_{Q_1} S_2$;

$S \leftarrow S_1 S_2$;

Return Q_2, S;

end

THEOEM 0.3.10

Algorithm 0.3.9 correctly computes a reduced form Q_2 equivalent to Q and a unimodular matrix S such that $Q \rightarrow Q_2$ via S.

Proof

From (0.18), (0.19) we have gcd $(\lambda,\nu) = 1$ and

$$a\lambda = \nu(h-b) \quad (0.21)$$

Hence there exist k,μ such that

$$\kappa\nu - \mu\lambda = 1$$

From (0.21) we have

$$c = -\lambda(b+h)/\nu \qquad (0.22)$$

since $h^2 = b^2 - ac$.

From the transformation $Q \to Q_1 = (a_1,b_1,c_1)$ via S_2 we have

$$c_1 = a\lambda^2 + 2b\lambda\nu + c\nu^2 =$$

$$= (h-b)\nu\lambda + 2b\lambda\nu - \lambda(b+h)\nu = 0$$

using (0.21) and (0.22).

Moreover

$$b_1 = ak\lambda + b(k\nu + \lambda\mu) + c\mu\nu =$$

$$= (h-b)\kappa + b(\kappa\nu + \lambda\mu) - \lambda(b+h)\mu = h(\kappa\nu-\lambda\mu) = h.$$

From the transformation $Q_1 \to Q_2$ via S_2 we have

$$a_2 = 2mh + a_1$$

$$b_2 = h$$

$$c_2 = 0$$

From (0.20) we have

$$0 < a_2 < 2h-1.$$

Hence Q_2 is reduced.

Directly we have that S is unimodular and

$$Q \to Q_2 \text{ via } S. \quad \square$$

THEOREM 0.3.11

Algorithm 0.3.9 terminates in $O(M(\log \|Q\|) \log\log \|Q\|)$ elementary operations in worst-case and $\log \|S\| = O(\log \|Q\|)$

Proof

The bottleneck of the algorithm is the application of the extended Euclidean algorithm (see Theorem 0.2.3) at the steps 1,2, which requires $O(M(\log \|Q\|) \log\log \|Q\|)$ elementary operations, since the remaining steps are of $O(M(\log \|Q\|)$ elementary operations for multiplications. Moreover easily we have $\lambda = O(\|Q\|)$, $\nu = O(\|Q\|)$ and using Theorem 0.2.3, one can find $|\kappa| < |\lambda|$, $|\mu| < |\nu|$ with $\kappa\nu - \lambda\mu = 1$.

Since $a_1 = a\kappa^2 + 2b\kappa\mu + c\mu^2 = O(\|Q\|^3)$, from (0.20) we have $m = O(\|Q\|^3)$.

Now using Proposition 0.1.7 we have

$$\|S\| < 4 \|S_1\| \|S_2\|$$

and since $\|S_1\| = O(\|Q\|)$ and $\|S_2\| = O(\|Q\|^3)$ we have

$$\log \|S\| = O(\log \|Q\|). \quad \square$$

1.1. GAUSS APPROACH

DEFINITION 1.1.1.

Gauss defined that the properly primitive form $Q_3 = (a_3,b_3,c_3)$ of determinant D (non-square, if $D > 0$) is composed of the properly primitive forms $Q_1 = (a_1,b_1,c_1)$ and $Q_2 = (a_2,b_2,c_2)$ with determinant D via a bilinear matrix B, if the following holds:

$$Q_1(X_1,X_2).Q_2(Y_1,Y_2) = Q_3(Z_1,Z_2) \qquad (1.1)$$

where $(Z_1,Z_2) = B \cdot (X_1,Y_1,X_1Y_2,X_2Y_1,X_2Y_2)^T$

for some bilinear matrix B with entries from the integers given by

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \end{pmatrix}$$

satisfying the conditions:

(i) Unimodularity : The greater common divisor of Δ_{ij}'s for $1 \leq i < j \leq 4$ is one, where Δ_{ij} is given by

$$\Delta_{ij} = \begin{vmatrix} b_{1i} & b_{ij} \\ b_{2i} & b_{2j} \end{vmatrix} = b_{1i}\, b_{2j} - b_{2i}b_{1j} \text{ for } 1 \leq i < j \leq 4$$

(ii) Orientability : $a_1\, \Delta_{12} > 0$

$a_2\, \Delta_{13} > 0$

Orientability of the bilinear matrix B is necessary to distinguish between composition with a form Q and composition with its opposite $\bar{Q}$.

The operation of composition of forms is denoted by

$$Q_1 \circ Q_2 = Q_3 \text{ via } B.$$

Now a method of composing two properly primitive forms Q_1 and Q_2 of determinant D (non-square if $D > 0$) to a properly primitive form Q_3 of determinant D via a bilinear matrix B is described. The method is due to Gauss (see [16].A.242-3)

The following lemma is in Dickson [15], p. 134.

LEMMA 1.1.2.

Suppose that $\gcd(m_1,m_2,\ldots,m_n) = 1$. If S divides $m_iq_j-m_jq_i$ for $1 \leq i, j \leq n$, there exist exactly one solution B mods of the system of equations

$$m_iB = q_i \pmod{S} \quad 1 \leq i \leq n \tag{1.2}$$

Proof

Since $\gcd(m_1,m_2,\ldots,m_n) = 1$, there exist integers a_i such that

$$\sum_{i=1}^{n} a_im_i = 1.$$

Also $S \mid m_iq_j-m_jq_i$ for $1 \leq i, j \leq n$, which implies $m_iq_j \equiv m_jq_i \pmod{S}$. Then $B \equiv \Sigma_{i=1}^{n} a_iq_i \bmod S$ is a solution

of the system (1.2), since

$$m_k B \equiv m_k \sum_{i=1}^{n} a_i q_i \equiv \sum_{i=1}^{n} a_i m_i q_k = q_k \sum_{i=1}^{n} a_i m_i \equiv q_k \pmod{S} \text{ for } 1 \leq k \leq n$$

Moreover if there were an integer B' such that $m_i B' \equiv q_i \pmod{S}$ $1 \leq i \leq n$, then $a_i m_i B' \equiv a_i q_i \pmod{S}$ for $1 \leq i \leq n$. Hence adding all the equations we have

$$(\sum_{i=1}^{n} a_i m_i) B' \equiv \sum_{i=1}^{n} a_i q_i \pmod{S}$$

and thus $B' \equiv B \pmod{S}$. □

Proofs of the following theorem were given by Gauss ([16], A.243), Mathews ([26], p. 152) and Pall ([29], p. 404).

THEOREM 1.1.3.

Suppose that $Q_1 = (a_1, b_1, c_1)$ and $Q_2 = (a_2, b_2, c_2)$ are properly primitive forms with determinant D (non-square , if $D > 0$). Let $\mu = \gcd(a_1, a_2, b_1 + b_2)$, $m_1 = a_1/\mu$, $m_2 = a_2/\mu$, $m_3 = (b_1 + b_2)/\mu$ and $a_3 = m_1 m_2$. Then

(i) The following system has integer coefficients, and a unique solution x (mod a_3)

$$\begin{aligned} m_1 x &\equiv b_2 m_1 && \text{(A)} \\ m_2 x &\equiv b_1 m_2 \pmod{a_3} && \text{(B)} \\ m_3 x &\equiv (b_1 b_2 + D)/\mu && \text{(C)} \end{aligned} \qquad (1.3)$$

(ii) Suppose b_3 is the solution of the system (1.3). Then $Q_3 = (a_3, b_3, *)$ is a form with determinant D and there exists a bilinear matrix B such that

$$Q_1 \circ Q_2 = Q_3 \quad \text{via B.}$$

Proof

(i) First we have

$$b_1b_2 + D = b_1b_2 + b_2^2 - a_2c_2 = b_2(b_1+b_2) - a_2c_2 \equiv 0 \pmod{\mu}$$

Hence the system (1.3) has integer coefficients.

Now one can see that gcd $(m_1, m_2, m_3) = 1$ and that a_3 divides $m_1m_2b_2 - m_2m_1b_1$, $m_2(b_1b_2+D)/\mu - m_3b_1m_2 = -m_1m_2c_1$ and $m_1(b_1b_2 + D)/\mu - m_3m_1b_2 = -m_1m_2c_2$. Hence applying Lemma 1.1.2 one can find the unique solution x (mod a_3) of the system (1.3) by choosing integers S_1, S_2, S_3 such that

$$S_1m_1 + S_2m_2 + S_3m_3 = 1$$

and defining

$$x = S_1b_2m_1 + S_2b_1m_3 + S_3(b_1b_2 + D)/\mu.$$

(ii) Let $Q_3 = (a_3, b_3, c_3)$ with b_3 a solution of the system (1.3). It will be shown that c_3 is an integer. Using (1.3)(c) we have

$$b_3^2 - D \equiv b_3^2 - (b_1+b_2)b_3 + b_1b_2 = (b_3-b_1)(b_3-b_2) \pmod{\mu a_3}.$$

Since from (1.3)(A), (1.3)(B),

$$b_3 - b_1 \equiv 0 \pmod{m_1} \text{ and } b_3 - b_2 \equiv 0 \pmod{m_2}$$

we have

$$b_3^2 - D \equiv 0 \pmod{a_3 = m_1 m_2}$$

Hence $c_3 = (b_3^2 - D)/a_3$ is an integer.

Now let B be a bilinear matrix given by

$$B = \begin{pmatrix} \mu & \frac{b_2-b_3}{m_2} & \frac{b_1-b_3}{m_1} & (b_1b_2+D-b_3m_3\mu)/\mu m_1 m_2) \\ 0 & m_1 & m_2 & m_3 \end{pmatrix}$$

Now after direct calculation we have

$$Q_1(X_1,X_2).Q_2(Y_1,Y_2) = Q_3(B\cdot Z)$$

where $Z^T = (X_1Y_1, X_1Y_2, Y_1X_2, Y_2X_2)$.

This implies that Q_3 is properly primitive, since Q_1 and Q_2 are properly primitive.

Moreover after direct calculation of Δ_{ij}'s of B can be shown that B is unimodular.

Since $a_1\Delta_{12} = a_1^2 > 0$, $a_2\Delta_{13} = a_2^2 > 0$, B satisfies the condition of orientability.

Hence $Q_1 \circ Q_2 = Q_3$ via B. □

ALGORITHM 1.1.4.

INPUT : Two properly primitive forms $Q_1 = (a_1,b_1,c_1)$ and $Q_2 = (a_2,b_2,c_2)$ of the same determinant D (non-square, if $D > 0$)

OUTPUT : A properly primitive form $Q_3 = (a_3,b_3,c_3)$ and a bilinear matrix B, which satisfy

$$Q_1 \circ Q_2 = Q_3 \text{ via } B.$$

Begin

1. $\mu \leftarrow \gcd(a_1,a_2,b_1+b_2)$;
2. $m_1 \leftarrow a_1/\mu$;
3. $m_2 \leftarrow a_2/\mu$;
4. $m_3 \leftarrow (b_1 + b_2)/\mu$;
5. <Find s_1,s_2,s_3 such that: $s_1m_1 + s_2m_2 + s_3m_3 = 1$>:

   Comment. This can be done with two applications of the Extended Euclidean algorithm.

6. $a_3 \leftarrow m_1m_2$;
7. $b_3 \leftarrow (s_1b_2m_1 + s_2b_1m_2 + s_3(b_1b_2 + D/\mu) \pmod{a_3}$;
8. $c_3 \leftarrow (b_3^2 - D)/a_3$;
9. $$B \begin{pmatrix} \mu & (b_2-b_3)/m_2 & (b_1-b_3)/m_1 & (b_1b_2+D-b_3m_3\mu)/\mu m_1m_2 \\ 0 & m_1 & m_2 & m_3 \end{pmatrix}$$

Return $Q_3 = (a_3,b_3,c_3)$, B;

end. □

EXAMPLE 1.1.5.

Let $Q_1 = (5,6,-8)$, $Q_2 = (3,-2,-24)$ be forms with determinant $D = 76$. Then

$$(5,6,-8) \circ (3,-2,-24) = (15,-14,8) \text{ via } B = \begin{pmatrix} 1 & 4 & 4 & 8 \\ 0 & 5 & 3 & 4 \end{pmatrix}. \quad \square$$

THEOREM 1.1.6.

Algorithm 1.1.4 correctly composes the forms Q_1 and Q_2 to a properly primitive form Q_3 via a bilinear matrix B.

Proof

It follows directly from 1.1.3. $\square$

THEOREM 1.1.7.

Algorithm 1.1.4 requires $O(M(\log ||Q||) \log\log ||Q||)$ elementary operations to compute a properly primitive form Q_3 and a bilinear matrix B such that $Q_1 \circ Q_2 = Q_3$ via B. Moreover $\log ||B|| = O(\log ||Q||)$, where $||Q|| = \max \{||Q_1||, ||Q_2||\}$.

Proof

Step 1 requires $O(M(\log ||Q||) \log\log ||Q||)$ using the Euclidean algorithm (see Theorem 0.2.3). The steps 2,3,4 require only $O(M(\log ||Q||))$ elementary operations for divisions. Step 5 requires $O(M(\log ||Q||) \log\log ||Q||)$ elementary operations for two applications of the Extended Euclidean algorithm (see Theorem 0.2.3).

Using Theorem 0.2.3, one can show that

$$s_1 = O(\|Q\|^2),\ s_2 = O(\|Q\|^2),\ S_3 = O(\|Q\|) \quad (1.4)$$

Step 6 requires only $O(M(\log \|Q\|))$ elementary operations for multiplication. Step 7 requires $O(M(\log \|Q\|)$ elementary operations for multiplications, since $\log |S_i| = O(\log \|Q\|)$ for $i = 1,2,3$. And Step 9 requires $O(M(\log \|Q\|))$ elementary operations for multiplications and divisions.

Hence the algorithm terminates in $O(M(\log \|Q\|) \log\log \|Q\|)$ elementary operations. It follows directly that $\log \|B\| = O(\log \|Q\|)$. □

COROLLARY 1.1.8.

If the forms Q_1, Q_2 of the input of algorithm 1.1.4 are reduced, then algorithm 1.1.4 requires $O(M(\log |D|) \log\log |D|)$ elementary operations to compute a properly primitive form Q_3 (not necessarily reduced) and a bilinear matrix B such that $Q_1 \circ Q_2 = Q_3$ via B. Moreover $\log \|B\| = O(\log |D|)$.

Proof

If Q_1, Q_2 are definite reduced forms, then from (0.7) we have

$$\|Q\| < |D|$$

which implies $\log \|Q\| = O(\log |D|)$.

If Q_1, Q_2 are indefinite reduced froms, then from (0.13)

$$||Q|| < 2\sqrt{D}$$

which implies $\log ||Q|| = O(\log |D|)$.

Hence the corollary follows from Theorem 1.1.7. □

DEFINITION 1.1.9.

Suppose that $M = (m_{ij})$ is a $p \times n$ matrix, and N a $k \times q$ matrix. Then

$$M \otimes N = \begin{pmatrix} m_{11}N & \cdots & m_{1n}N \\ m_{p1}N & \cdots & m_{pn}N \end{pmatrix}$$

is a $kp \times nq$ matrix and $M \otimes N$ is said to be the *Kronecker product* of M and N.

PROPOSITION 1.1.10

Suppose $Q_1 \circ Q_2 = Q_3$ via B, $Q_1' \to Q_1$ via S_1, $Q_2 \to Q_2'$ via S_2 and $Q_3 \to Q_3'$ via S_3. Then

$$Q_1' \circ Q_2' = Q_3' \text{ via } S_3^{-1}B(S_1 \otimes S_2)$$

where $\otimes$ is the Kronecker product.

Proof

Suppose $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = S_1 \cdot \begin{pmatrix} x_1' \\ x_2' \end{pmatrix}$ (1), $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = S_2 \cdot \begin{pmatrix} y_1' \\ y_2' \end{pmatrix}$ (2). Then

$$z = (S_1 \otimes S_2)z' \quad (3)$$

where $z^T = (x_1y_1, x_1y_2, x_2y_1, x_2y_2)$ and $(z')^T = (x_1'y_1', x_1'y_2', x_2'y_2', x_2'y_2')$.

From $Q_1 \circ Q_2 = Q_3$ we have,

$$Q_1(x_1,x_2)\cdot Q_2(y_1,y_2) = Q_3(B\cdot Z) \rightarrow$$

$$\rightarrow (x_1,x_2)\, M_{Q_1} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \cdot (y_1,y_2)\, M_{Q_2} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = (BZ)^T M_{Q_3} BZ$$

Now using (1), (2), (3) we have

$$(x_1',x_2')S_1^T M_{Q_1} S_1 \begin{pmatrix} x_1' \\ x_2' \end{pmatrix} \cdot (y_1',y_2')S_2^T M_{Q_2} S_2 \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} =$$

$$= (B(S_1 \otimes S_2)Z')^T M_{Q_2} B(S_1 \otimes S_2)Z'.$$

Now since

$$M_{Q_i'} = S_i^T M_{Q_i} S_i \text{ for } i = 1,2,3$$

we have $(x_1',x_2')\, M_{Q_1'} \begin{pmatrix} x_1' \\ x_2' \end{pmatrix} (y_1',y_2')\, M_{Q_2'} \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = (B'Z')^T M_{Q_3'} B'Z'$

where $B' = S_3^{-1}\cdot B(S_1 \otimes S_2)$.

Hence $Q_1'(X_1',X_2')\cdot Q_2'(Y_1',Y_2') = Q_3'(B'Z')$, which implies $Q_1' \circ Q_2' = Q_3'$ via B', since from Proposition 0.1.4 Q_1', Q_2', Q_3' are properly primitive forms and after computation it can be shown that B' satisfies the conditions of unimodularity and orientability. □

The following Corollary is an application of Corollary 1.1.8 and Proposition 1.1.10.

COROLLARY 1.1.11.

There exists an algorithm which composes two properly primitive reduced forms Q_1, Q_2 with determinant D (non-square, if D > 0) to a properly primitive reduced form Q_3 via a bilinear matrix B in $O(M(\log |D|) \log |D|)$ elementary operations and $\log ||B|| = O(\log |D|)$.

Proof

If algorithm 1.1.4 is applied to Q_1, Q_2, it will give a form Q_3' and bilinear matrix B' such that

$$Q_1 \circ Q_2 = Q_3' \text{ via } B'$$

in $O(M(\log |D|) \log \log |D|)$ elementary operations, since Q_1, Q_2 are reduced.

One can see that with $||Q|| = \max \{||Q_1||, ||Q_2||\}$

$$||Q_3'|| = O(||Q||^6) \tag{1.5}$$

from the steps 6-8 of the algorithm 1.1.4 and (1.4).

From (1.5) and since Q_1, Q_2 are reduced, we have

$$||Q_3'|| = O(|D|^6) \tag{1.6}$$

using (0.6) or (0.13) for the appropriate type of D.

Now we use a reduction procedure (0.3.2 if D < 0, 0.3.6 if D > 0) to obtain a reduced form Q_3 and a unimodular matrix S such that: $Q_3' \to Q_3$ via S. It requires $O(\log ||Q_3'|| \; M(\log ||Q_3'||)) = O(\log |D| \; M(\log D))$ elementary operations and

$$\log \|S\| = O(\log \|Q_3'\|) = O(\log |D|) \qquad (1.7)$$

Now using Proposition 1.1.8 we have

$$Q_1 \circ Q_2 = Q_3 \text{ via } B = S^{-1}B'$$

where Q_3 is reduced. Hence, from the above analysis, the algorithm terminates in $O(\log |D| \; M(\log |D|)$ elementary operations.

Moreover using 0.1.7 we have

$$\|B\| < 16 \|S^{-1}\| \quad \|B'\| \qquad (1.8)$$

Since S is unimodular, we have $\|S\| = \|S^{-1}\|$.

Hence from (1.7), (1.8) we have

$$\log \|B\| = O(\log |D|). \quad \square$$

1.2. THE FORM CLASS GROUP

It will be shown that the equivalent classes of properly primitive forms with determinant D (non-square, if D > 0) form an abelian group under composition. First it is necessary to show the following proposition.

PROPOSITION 1.2.1.

Suppose $Q_1 \circ Q_2 = Q_3$ via B, $Q_1 \circ Q_2 = Q_3'$ via B'. Then $Q_3 \approx Q_3'$.

Proof

Let $B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \end{pmatrix}$, $B' = \begin{pmatrix} b'_{11} & b'_{12} & b'_{13} & b'_{14} \\ b'_{21} & b'_{22} & b'_{23} & b'_{24} \end{pmatrix}$

and $\Delta_{ij} = b_{1i}b_{2j} - b_{2i}b_{1j}$, $\Delta'_{ij} = b'_{1i}b'_{2j} - b'_{2i}b'_{1j}$ for $1 \leq i < j \leq 4$.

It can be shown that $\Delta_{ij} = \Delta'_{ij}$ for $1 \leq i < j \leq 4$ with direct calculation from the relation $Q_1(X_1,X_2)\cdot Q_2(Y_1,Y_2) = Q_3(BZ) = Q_3'(B'Z)$, where Z as in Proposition 1.1.3.

Since the greatest common divisor of Δ_{ij}'s for $1 \leq i < j \leq 4$ is one, there exist integers a_{ij}, $1 \leq i < j \leq 4$ such that:

$$\sum_{i,j} a_{ij}\,\Delta_{ij} = 1.$$

Let now

$$S_1 = \sum_{i,j} a_{ij}(b'_{1i}b_{2j} - b'_{2j}b_{2i})$$

$$S_1 = \sum_{i,j} a_{ij}(b_{1i}b_{1j} - b_{1j}b'_{1i})$$

$$S_3 = \sum_{i,j} a_{ij}(b'_{2i}b_{2j} - b'_{2j}b_{2i})$$

$$S_4 = \sum_{i,j} a_{ij})b_{1i}b'_{2j} - b_{1j}b'_{2i})$$

Then with direct calculation, using the fact $\Delta_{ij} = \Delta'_{ij}$ for $1 \leq i < j \leq 4$, can be shown that the matrix

$$S = \begin{pmatrix} S_1 & S_2 \\ S_3 & S_4 \end{pmatrix}$$

is unimodular and $Q_3 \to Q'_3$ via S. For details of the calculations one can see Mathews ([26], p. 146). □

Proposition 1.1.10 and Proposition 1.2.1 show that composition is well defined and independent of the choice of the bilinear matrix B over the equivalence classes of properly primitive forms with fixed determinant D (non-square, if D > 0). Hence we may write:

$$Q_1 \circ Q_2 = Q_3$$

where Q_i is used to denote the class to which it belongs for $1 \leq i \leq 3$.

Also let $Cl(D)$ denote the set of equivalence classes of properly primitive forms with determinant D.

THEOREM 1.2.2.

$(Cl(D), o)$ is a finite abelian group, where o is the operation of composition and D is non-square, when D is positive.

Proof

Let $K,L,N \in Cl(D)$ and $(a_1,b_1,c_1) \in K$, $(a_2,b_2,c_2) \in L$.

(i) The operation of composition of classes is well defined by Propositions 1.1.10 and 1.2.1.

(ii) As an application of Theorem 1.1.3 we have

$$(K \circ L) \circ N = K \circ (L \circ N)$$ (see Mathews [26], p. 153)

(iii) Let I be the class of $Cl(D)$ which contains the form $(1,0,-D)$. Then

$$K \circ I = K \quad \text{for every } K \in Cl(D)$$

since using Theorem 1.1.3

$$(a_1,b_1,c_1) \circ (1,0,-D) = (a_1,b_1,c_1)$$

(iv) Let $(a_1,-b_1,c_1) \in M$. Then using Theorem 1.1.3 we have

$$(a_1,b_1,c_1) \circ (a_1,-b_1,c_1) = (1,0,-D)$$

Hence M is the inverse of the class K.

(v) Using Theorem 1.1.3 can be shown that

$$(a_1,b_1,c_1) \circ (a_2,b_2,c_2) = (a_2,b_2,c_2) \circ (a_1,b_1,c_1)$$

Hence $K \circ L = L \circ K$.

From (i) - (v) we have that $(\mathcal{C}\ell(D),\circ)$ is an abelian group.

Since every class can be represented by a reduced form belonging to it, we have that the number of classes of $C\ell(D)$ is less than the number of the reduced forms. Now if $D > 0$, then from inequality (0.13) it follows that the number of reduced forms is finite. Similarly if $D < 0$, then from inequality (0.7) we have that the number of the reduced forms is finite. Hence $C\ell(D)$ is a finite group. □

The group $(C\ell(D),\circ)$ is called the *form class group*. The identity element of the form class group is called the *principal class*.

In current research in number theory the language of divisors (see [8]) is used instead of the language of forms, since it is simpler. The forms seem to provide a convenience computational model of the algebraic number theory. Since later some results of algebraic number theory are used (theorems 4.1.5 - 4.1.6), it is useful to review briefly some relations between quadratic forms and quadratic fields. Extended reference to algebraic number theory is outside of the scope of this thesis; see references in [13] for further details.

DEFINITION 1.2.3.

Suppose that $\mathbb{Q}(\sqrt{d})$ is the quadratic extension of the field of rationals by $\sqrt{d}$, where d is square-free. If $f \geq 1$ is a rational integer (see [13], p. 43), then the ring consisting of quadratic integers of $\mathbb{Q}(\sqrt{d})$ which are expressible as

$$a + bfw_o$$

where a and b are rational integers, and

$$w_o = \begin{cases} \frac{1+\sqrt{d}}{2} & d \equiv 1 \pmod 4 \\ \sqrt{d} & d \not\equiv 1 \pmod 4 \end{cases}$$

is called the *order of index* f, denoted by $\mathcal{O}_f(d)$ (see [13],p.216).

The *discriminant* Δ_f of $\mathcal{O}_f(d)$ is

$$\Delta_f = f^2 d \qquad \text{(see [13], p.216)}$$

It is known that the strict equivalence classes of ideals of an order $\mathcal{O}_f(d)$ form an abelian group under the composition of classes of ideals. (See [13], p. 114; 197; 212; and [5], p. 278). Moreover it can be shown that the equivalence classes of properly primitive forms (in Dirichlet's terminology (see below 0.1.7), with fixed discriminant Δ form a group under composition, (proof similar to that of Theorem 1.2.2). This group is isomorphic to the class group of primitive ideals of the order $\mathcal{O}_f(d)$ with discriminant $\Delta = f^2 d$ with d square free (see [13], 200-216). Since the Gaussian forms are a special case of Dirichlet's forms and they have discriminant $\Delta = 4D$ (where D is the determinant of the form), we have that $C\ell(D)$ is isomorphic to class group of primitive ideals of the order $\mathcal{O}_{2f}(d)$, where $D = f^2 d$ with d square-free.

THEOREM 1.2.4.

If $J = [\alpha,\beta]$ is a primitive ideal of an order $\mathcal{O}_{2f}(d)$, then

$$Q(x,y) = N(\alpha x + \beta y)/N(J)$$

is a properly primitive integral binary quadratic form with determinant $D = f^2 d$. The map defined in this way induces a 1-1 correspondence between $C\ell(D)$ and the class group of primitive ideals which is a group isomorphism.

Proof

See Cohn ([13], p. 200-216). □

1.3. DIRICHLET'S APPROACH

In Section 1.1 we discussed Gauss' method for composition of forms, which makes use of bilinear substitutions. It was also shown that the operation of composition is well defined over $C\ell(D)$. Dirichlet's method of composition of classes of forms depends upon finding a pair of representatives which can be simply composed.

DEFINITION 1.3.1.

Two forms $Q = (A,B,C)$, $F = (a,b,c)$ with the same determinant D are called *concordant* if:

(i) $\gcd(A,a) = 1$

(ii) $B = b$.

The following proposition is a combination of results by Cassels ([12], Lemma 2.2, p. 334) and Gauss ([16], A.168).

PROPOSITION 1.3.2.

Suppose K and L are classes of $C\ell(D)$. Then there exist forms $Q_1 \in K$, $Q_2 \in L$ which are concordant.

Proof

Let $Q = (a,b,c) \in K$ and $F = (A,B,C) \in L$. First will be shown that Q can represent an integer N coprime to A. We require coprime integers x and y which satisfy for every prime divisor p of A:

(i) If $p \nmid a$, then $p \nmid x$ and $p \mid y$.

(ii) If $p \nmid c$, then $p \mid x$ and $p \nmid y$.

(iii) If $p \mid a$ and $p \mid c$ (hence $p \nmid 2b$), then $p \nmid x$ and $p \nmid y$.

One can see that such integers x,y always exist and that $Q(x,y) = N$ is coprime to A. Since x,y are coprime, there exist integers k, m such that

$$kx + my = 1.$$

Now let

$$Q \to G \text{ via } \begin{pmatrix} x & -m \\ y & k \end{pmatrix} .$$

Then $G = (N,M,*)$ with $M = k(bx + cy) - m(ax + by)$.

Since N, A are coprime, then there exist integers t, z such that

$$tN + zA = 1.$$

Now we do the transformations

$$G \to Q_1 \text{ via } \begin{pmatrix} 1 & \mu \\ 0 & 1 \end{pmatrix}$$

$$F \to Q_2 \text{ via } \begin{pmatrix} 1 & \nu \\ 0 & 1 \end{pmatrix}$$

with $\mu = t(B-M)$ and $\nu = z(M-B)$.

It is obvious that $Q_1 \in K$ and $Q_2 \in L$. Moreover direct calculation shows that Q_1 and Q_2 are concordant. □

PROPOSITION 1.3.3

Suppose that $Q_1 = (A,b,*)$ and $Q_2 = (a,b,*)$ are concordant properly primitive forms with determinant D. Then

$$Q_1 \circ Q_2 = (Aa,b,*) \text{ via } B = \begin{pmatrix} 1 & 0 & 0 & (b^2-D)/Aa \\ 0 & A & a & 2b \end{pmatrix}$$

Proof

It is a direct application of 1.1.3. □

Dirichlet's method of composing two properly primitive forms Q_1, Q_2 with the same determinant D, finds as in Proposition 1.3.2 a pair of concordant forms $Q_1' = (A,b,*)$ and $Q_2' = (a,b,*)$ equivalent to Q_1 and Q_2 respectively. Then by Proposition 1.3.1 there exists a bilinear matrix B such that:

$$Q_1' \circ Q_2' = (Aa,b,*) \text{ via } B \qquad (1.9)$$

If $Q_1' \to Q_1$ via S_1 and $Q_2' \to Q_2$ via S_2, where S_1, S_2 are unimodular, then by Proposition 1.1.10 and (1.9) we have

$$Q_1 \circ Q_2 = (Aa,b,*) \text{ via } B(S_1 \otimes S_2) \qquad (1.10)$$

Lagarias ([20], p. 169) gave an algorithm for composition of forms; it follows Dirichlet's method. To compose two properly primitive forms $Q_1 = (a_1,b_1,c_1)$ and $Q_2 = (a_2,b_2,c_2)$ with the same determinant D (non-square, if $D > 0$), it finds first two concordant forms $Q_1' = (a',b,*) \approx Q_1$ and $Q_1' = (A',b,*) \approx Q_2$, and thus a bilinear matrix B (using 1.3.3 and (1.10)) such that

$$Q_1 \circ Q_2 = (A'a',b,*) \text{ via } B.$$

To find two concordant forms Q_1', Q_2', it transforms

$$Q_1 \to F = (A,B,C) \text{ via } S = \begin{pmatrix} 1 & 0 \\ \lambda & 1 \end{pmatrix} \tag{1.11}$$

by choosing a λ such that

$$\gcd(A,a_2) = \gcd(a_1 + \lambda(2b_1 + \lambda c_1), a_2) = 1$$

and after transforms

$$F \to Q_1' \text{ via } S_1 = \begin{pmatrix} 1 & \mu \\ 0 & 1 \end{pmatrix}$$

$$Q_2 \to Q_2' \text{ via } S_2 = \begin{pmatrix} 1 & \nu \\ 0 & 1 \end{pmatrix}$$

where $\mu = m(B - b_2)$, $\nu = n(b_2 - B)$ for some integers m,n: $ma' + na_2 = 1$.

One can see with direct calculation that Q_1', Q_2' are concordant.

The hard part of the algorithm is to compute the λ and thus the form F. Hence first we give an algorithm to compute F as above.

ALGORITHM 1.3.4.

INPUT : A properly primitive form $Q = (a,b,c)$ and an integer N. We assume w.l.o.g. that a is odd. (see remark below)

OUTPUT : A form $F = (A,B,C) \approx Q$ with $\gcd(A,N) = 1$ and a unimodular matrix S such that

$$Q \to F \text{ via } S$$

Procedure PRIMEFACTOR (A,N)

Comment This procedure factors determines integer m_1 and m_2 such that

(i) $N = m_1 m_2$,

(ii) gcd $(m_1, m_2) = 1$,

(iii) gcd $(A, m_2) = 1$ and

(iv) If p prime divides m_1, then $p|A$

Begin

1. $z \leftarrow$ gcd (A,N);
   $v \leftarrow N$;
2. While $z \neq 1$ do
   Begin
3. <choose the maximum λ for which v/z^{λ} is integer>;
4. $v \leftarrow v/z^{\lambda}$;
5. $z \leftarrow$ gcd (z,v);
6. end
7. $m_1 \leftarrow N/v$; $m_2 \leftarrow v$;
   Return (m_1, m_2);
   end.

The algorithm for the computation of the form F is:

Begin

10. $(m, m_1) \leftarrow$ PRIMEFACTOR (a,N);

11. $(m_2, m_3) \leftarrow$ PRIMEFACTOR $(2b+c,N)$;

Comment Now $N = m_1 m_2 m_3$ and m_1, m_2, m_3 satisfy:

$\gcd(a,m_1) = \gcd(m_1,m_2,m_3) = \gcd(2b+c,N) = \gcd)m_2,m_3) = 1$

for p prime, if $p|m$, then $p|a$ and if $p|m_2$, then $p|2b_1+c_1$

12. <Find integers k_2, k_3 such that: $k_2 m_2 + k_3 m_3 = 1$>

13. <Find integers k_1, k_4 such that: $k_1 m_1 + k_4 m_2 m_3 = 1$>;

14. $\lambda \leftarrow k_1 m_1 (k_2 m_2 + 2\, k_3 m_3) \pmod{a_2}$;

Comment Now λ is as in (1.11) with the required property

$$S \leftarrow \begin{pmatrix} 1 & 0 \\ \lambda & 1 \end{pmatrix}$$

15. $M_F \leftarrow S^T M_Q S$;

Return F ,S:

end. □

Remark

If one has $Q = (a,b,c)$ with a even, then c is odd, since Q is properly primitive. Moreover

$$Q \rightarrow (c,-b,a) \text{ via } S' = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Hence one can apply algorithm 1.3.4 to the form $(c,-b,a)$, to compute a form $F = (A,B,C)$ with $\gcd(A,N) = 1$ and a unimodular matrix S such that,

$(c,-b,a) \rightarrow F$ via S.

Then easily $Q \approx F$ and

$Q \rightarrow F$ via S'S.

First the procedure PRIMEFACTOR is analysed and its correctness is proved.

THEOREM 1.3.5

The procedure PRIMEFACTOR (A,N) correctly computes (m_1,m_2) with the required properties in $O(M(\log K)\log K)$ elementary operations, where $K = \max\{|A|, |N|\}$.

Proof

Suppose that $A = \prod_\alpha p_\alpha^{\tau_\alpha} \prod_\beta q_\beta^{t_\beta}$ and $N = \prod_\alpha P_\alpha^{h_\alpha} \prod_\gamma S_\gamma^{n_\gamma}$. It will be shown that

$$m_2 = \prod_\gamma S_\gamma^{n_\gamma}$$

Now let v_i,λ_i,z_i denote v,λ,z respectively at the beginning of the i-th loop. One can see that $z_{i+1} = \gcd(z_i,v_{i+1}) < z_i$, hence for some integer T, we shall have $z_{T+1} = 1$ and $z_T \neq 1$.

It is obvious that $v_i | N$ for every i. Also $\prod_\gamma S_\gamma^{n_\gamma} | v_i$ for every i.

Now since $z_{T+1} = 1$, for each α there is j_α such that

$p_\alpha | z_{J_\alpha}$ and $p_\alpha \nmid z_{J_\alpha+1}$. Hence since $z_{i+1} = \gcd(z_i, v_{i+1})$, it follows that $p_\alpha \nmid v_{J_\alpha+1}$. But we know that $v_{i+1} | v_i$ for every i, so we have $p_\alpha \nmid v_{T+1}$ for every α.

Hence we conclude that

$$m_2 = v_{T+1} = \prod_\gamma S_\gamma^{n_\gamma} .$$

and thus the procedure PRIMEFACTOR correctly computes (m_1, m_2).

One can see that $z_{i+1} = \gcd(z_i, v_{i+1}) < z_i$ for $i < T$ and thus

$$z_i / z_{i+1} > 2 \text{ for every } i < T$$

which implies

$$z_{T-1} > 2^i \quad \text{for every } i < T$$

Hence we have

$$\prod_{i=1}^{T} z_i > \prod_{i=1}^{T} 2^i > 2^{T^2/2}$$

Now since $\prod_{i=1}^{T} z_i < N$, we have $2^{T^2/2} < N$ which implies

$$T = O(\sqrt{\log N}) < O(\sqrt{\log K}) .$$

Also one can see that

$$\sum_{i=1}^{T} \lambda_T < \sum_\alpha h_\alpha = O(\log N) < O(\log K)$$

Hence the procedure requires $T = O((\log K)^{1/2})$ applications of the Euclidean algorithm and $O(\log K)$ divisions. Hence the

procedure terminates in $O(M(\log K)\log K)$ elementary operations. □

THEOREM 1.3.6.

Algorithm 1.3.4 correctly computes a form $F \approx Q$ with $\gcd(A,N) = 1$ and a unimodular matrix S such that $Q \to F$ via S.

Proof.

It is sufficient to prove that the chosen λ satisfy

$$\gcd(A,N) = 1 \text{ with } A = a + \lambda(b + \lambda c)$$

First we have that $2 \nmid a$.

The integer $N = m_1 m_2 m_3$ with m_i, $i = 1,2,3$ satisfying:

$$\gcd(a, m_1) = 1 \tag{1.12}$$

$$\gcd(m_1, m_2 m_3) = 1 \tag{1.13}$$

$$\gcd(2b+c, m_3) = 1 \tag{1.14}$$

$$\gcd(m_2, m_3) = 1 \tag{1.15}$$

and for p prime we have

$$p \mid m_2 m_3 \Rightarrow p \mid a \text{ and } p \neq 2 \text{ (since } 2 \nmid a) \tag{1.16}$$

$$p \mid m_2 \Rightarrow p \mid 2b + c \tag{1.17}$$

Now from step 14 of the algorithm we have

$$\lambda \equiv 0 \pmod{m_1} \tag{1.18}$$

From steps 12, 14 we have

$$\lambda \equiv 2k_1 m_1 k_3 m_3 = 2k_1 m_1 (1 - k_2 m_2) \equiv k_1 m_1 \equiv 2 \pmod{m_2} \tag{1.19}$$

and from steps 13, 14 we have

$$\lambda \equiv k_1 m_1 k_2 m_2 = (1 - k_4 m_2 m_3) k_2 m_2 \equiv k_2 m_2 \equiv 1 \pmod{m_3} \quad (1.20)$$

Now let p prime such that $p|N$. Then either $p|m_1$ or $p|m_2$ or $p|m_3$

Case $p|m_1$

From (1.12) $p \nmid a$ and from (1.18) $p|\lambda$. Hence $p \nmid a$

Case $p|m_2$

From (1.16) $p|a$ and $p \neq 2$. From (1.17) $p|2b+c$ and hence using (1.19) we have

$$2b + \lambda c \equiv 2b + 2c \equiv c \pmod{p} \quad (1.21)$$

If $c \equiv 0 \pmod{p}$, then since $p|2b + c$, we have $2b \equiv 0 \pmod{p}$ which contradicts with the fact that Q is properly primitive. Hence $c \not\equiv 0 \pmod{p}$ and from (1.21)

$$2b + 2c \not\equiv 0 \bmod p \quad (1.22)$$

Now using (1.19), (1.21), (1.22) and the facts $p|a$, $p \nmid 2$ we have

$$A = a + \lambda(2b + \lambda c) \equiv 2(2b + 2c) \not\equiv 0 \pmod{p}$$

Hence $p \nmid A$

Case $p \nmid m_3$

From (1.14) we have

$$2b + c \not\equiv 0 \pmod{p} \quad (1.23)$$

From (1.16), (1.20), (1.23) we have

$$A = a + \lambda(2b + \lambda c) \equiv \lambda(2b + c) = 2b + c \not\equiv 0 \bmod p$$

Hence $p \nmid A$.

From the above cases we conclude that

$$\gcd(A,N) = 1. \quad \square$$

THEOREM 1.3.7

The algorithm 1.3.4 terminates in $O(M(\log K)\log K)$ elementary operations in worst-case and $\log ||S|| = O(\log K)$, where $K = \max \{||Q||, N\}$.

Proof

Steps 10-11 by Theorem 1.3.5 require $O(M(\log K)\log K)$ elementary operations. Steps 12-13 by Theorem 0.2.3 require $O(M(\log K)\log\log K)$ elementary operations and $\log |k_i| = O(\log K)$ for $i = 1,2,3,4$. Hence step 14 by Theorem 0.2.2 require $O(M(\log K))$ elementary operations and $\log |\lambda| = O(\log K)$.

Step 15 requires $O(M(\log K))$ elementary operations.

Hence the algorithm terminates in $O(M(\log K)\log K)$ elementary operations and since $\log |\lambda| = O(\log K)$ we have

$$\log ||S|| = O(\log K). \quad \square$$

The following algorithm is Lagarias' algorithm for composition using algorithm 1.3.4.

ALGORITHM 1.3.8

INPUT : Two properly primitive forms $Q_1 = (a_1,b_1,c_1)$, $Q_2 = (a_2,b_2,c_2)$ with the same determinant D (non-square, if $D > 0$).

OUTPUT : A properly primitive form $Q_3 = (a_3,b_3,c_3)$ with determinant D and a bilinear matrix B such that

$$Q_1 \circ Q_2 = Q_3 \text{ via } B$$

Begin

1. <Using algorithm 1.3.4 find a form $F = (A,B,C) \approx Q_1$ with $\gcd(A,a_2) = 1$ and a unimodular matrix S such that $Q_1 \to F$ via S>
2. <Find integers m,n: $mA + na_2 = 1$>;
3. $\mu \leftarrow m(B - b_2)$;
4. $\nu \leftarrow n(b_2 - B)$;
5. $S_1 \leftarrow \begin{pmatrix} 1 & \mu \\ 0 & 1 \end{pmatrix}$;
6. $S_2 \leftarrow \begin{pmatrix} 1 & \nu \\ 0 & 1 \end{pmatrix}$;
7. $M_{Q_1'} \leftarrow S_1^T M_F S_1$;
8. $MQ_2' \leftarrow S_2^T MQ_2 S_2$;

   Comment Now Q_1', Q_2' are concordant;

   <Let $Q_1' = (a',b,*)$ and $Q_2' = (A',b,*)$>;

   $Q_3 \leftarrow (A'a',b,*)$;

9. $B \leftarrow \begin{pmatrix} 1 & 0 & 0 & (b^2-D)/A'a' \\ 0 & A' & a' & 2b \end{pmatrix} ((S\cdot S_1)^{-1} \otimes S_2^{-1})$;

Comment The matrix B is constructed as in (1.10)

Return Q_3, B;

end. □

THEOREM 1.3.9

Algorithm 1.3.8 correctly composes the forms Q_1 and Q_2 to a form Q_3 via a bilinear matrix B.

Proof

See Theorem 1.3.3 and description of the algorithm (below Theorem 1.3.3). □

THEOREM 1.3.10

Algorithm 1.3.8 terminates in $O(\log \|Q\| M(\log \|Q\|))$ elementary operations and $\log \|B\| = O(\log \|Q\|)$, where $\|Q\| = \max \{\|Q_1\|, \|Q_2\|\}$.

Proof

By Theorem 1.3.5 step 1 requires $O(\log \|Q\| M(\log \|Q\|))$ elementary operations and $\log \|S\| = O(\log \|Q\|)$ (1.24)

Step 2 by Theorem 0.2.3 requires $O(\log \log \|Q\| M(\log \|Q\|))$ elementary operations, since $\log |A| = O(\log \|Q\|)$ using (1.24). Moreover by Theorem 0.2.3 one can see that

$\log |m| = O(\log ||Q||)$ and $\log |n| = O(\log ||Q||)$. Hence steps 3-4 require $O(M(\log ||Q||))$ elementary operations for multiplications and $\log ||S_1|| = O(\log ||Q||)$, $\log ||S_2|| = O(\log ||Q||)$ (1.25)

Easily steps 7-8 require only $O(M(\log ||Q||))$ elementary operations.

From (1.24) and (1.25) we have that Step 9 requires only $O(M(\log ||Q||)$ elementary operations for multiplications.

Hence the algorithm terminates in $O(\log ||Q|| M(\log ||Q||))$ elementary operations.

Also by Theorem 0.1.7, (1.24) and (1.25) we have

$$\log ||B|| = O(\log ||Q||). \quad \square$$

COROLLARY 1.3.11

. If the forms Q_1, Q_2 of the input of the algorithm 1.3.8 are reduced, then the algorithm has running time $O(\log |D| M(\log |D|))$ elementary operations and $\log ||B|| = O(\log|D|)$.

Proof

Using the inequality (0.7) or (0.13) for the appropriate type of D, we have

$$\log ||Q|| = O(\log |D|) \qquad (1.26)$$

Hence the corollary follows from Theorem 1.3.10. $\square$

Remark

Comparing algorithm 1.1.4 (Gauss' method) with algorithm 1.3.8 (Dirichlet's method) one can see that 1.1.4. is simpler

Moreover our analysis of 1.1.4 shows that the Gauss' method composes two forms faster than Lagarias' algorithm (see Theorems 1.1.7 - 1.1.8 and 1.3.10 - 1.3.11)

COROLLARY 1.3.12

There exists an algorithm which composes two properly primitive reduced forms Q_1, Q_2 with determinant D (non-square, if $D > 0$) to properly primitive reduced form Q_3 via a bilinear matrix B in $O(M(\log |D|)\log |D|)$ elementary operations and $\log ||B|| = O(\log |D|)$.

Proof

If algorithm 1.3.8 is applied to the reduced forms Q_1, Q_2, it will yield a form Q_3' (not necessarily reduced) and a bilinear matrix B' such that

$$Q_1 \circ Q_2 = Q_3' \text{ via } B'$$

in $O(M(\log |D|) \log |D|)$ elementary operations using Corollary 1.3.11.

Now one using (1.24) - (1.26) can show

$$\log ||Q_3'|| = O(\log |D|) \tag{1.27}$$

Now we reduce Q_3' to Q_3 using algorithm 0.3.2 or 0.3.6 for the appropriate type of D and let

$$Q_3' \to Q_3 \text{ via } S.$$

By Theorem 0.3.4 or 0.3.7 and (1.27) the reduction can be done in $O(M(\log \|Q_3'\|) \log \|Q_3'\|) = O(\log |D| \; M(\log |D|))$ elementary operations and

$$\log \|S\| = O(\log \|Q_3'\|) = O(\log |D|) \qquad (1.28)$$

Now from Proposition 1.1.10 we have

$$Q_1 \circ Q_2 = Q_3 \text{ via } B = S^{-1}B'$$

Now using Theorem 0.1.7 we have

$$\|B\| < 16 \|S^{-1}\| \; \|B'\|$$

Since $\|S^{-1}\| = \|S\| = O(\log |D|)$ by (1.28) and $\log \|B'\| = O(\log |D|)$ by Corollary 1.3.11, we have

$$\log \|B\| = O(\log |D|). \quad \square$$

2.1. GENUS CHARACTERS

DEFINITION 2.1.1.

A homomorphism χ from a group $(G,\circ)$ to the multiplicative subgroup of the field of the complex numbers is a *character* of the group G.

DEFINITION 2.1.2.

The character χ_o of the group G which satisfies

$$\chi_o(a) = 1 \quad \forall\ a \in G$$

is called the *principal character*.

DEFINITION 2.1.3.

The characters of a group G themselves form a group with identity element χ_o with respect to the multiplication

$$(\chi_1 \cdot \chi_2)(a) = \chi_1(a) \cdot \chi_2(a) \quad \forall\ a \in G \quad \text{(see Apostol [4], p.135)}$$

A character χ of a group G of order 2 i.e. such that

$$\chi^2 = \chi_o$$

is called a *genus character*.

In this section the genus characters of the form class group will be studied. First it is necessary to show the following lemmas. The first is due to Mathews ([26], p. 132).

LEMMA 2.1.4.

If Q is a properly primitive binary quadratic form, there is a number N represented by Q with gcd(N,2D) = 1.

Proof

Suppose that Q = (a,b,c). Then let

$$\gcd(a,c,2D) = \prod_{\alpha} p_{\alpha}^{n_{\alpha}}$$

$$\gcd(a,2D) = \prod_{\alpha} p_{\alpha}^{m_{\alpha}} \prod_{\beta} q_{\beta}^{v_{\beta}} \tag{2.1}$$

$$\gcd(c,2D) = \prod_{\alpha} p_{\alpha}^{k_{\alpha}} \prod_{\gamma} \tau_{\gamma}^{t_{\gamma}} \tag{2.2}$$

$$2D = \prod_{\alpha} p_{\alpha}^{e_{\alpha}} \prod_{\beta} q_{\beta}^{u_{\beta}} \prod_{\gamma} \tau_{\gamma}^{z_{\gamma}} \prod_{i=1}^{\delta} s_i^{h_i} \tag{2.3}$$

where p_i, q_i, τ_i, are distinct primes and $n_{\alpha} < m_{\alpha} < e_{\alpha}$, $n_{\alpha} < k_{\alpha} < e_{\alpha}$, $v_{\beta} < u_{\beta}$ and $t_{\gamma} < z_{\gamma}$.

If $x = \prod_{\beta} q_{\beta} \prod_{i=1}^{\delta} s_i$ and $y = \prod_{\gamma} \tau_{\gamma}$, then let

$$N = Q(x,y).$$

It can be shown that gcd(N,2D) = 1. □

Remark

If we partition the s_i's in two disjoint sets, say $\{s_1,\ldots,s_{\mu}\}$, $\{s_{\mu+1},\ldots,s_{\delta}\}$, then Q(x',y') = N' with

$x' = \prod_{\beta} q_{\beta} \cdot s_1 \ldots s_{\mu}$, $\gamma' = \prod \tau_{\gamma} \cdot s_{\mu+1} \ldots s_{\delta}$ satisfies

gcd(N',2D) = 1.

LEMMA 2.1.5.

Suppose N_1, N_2 are represented by the forms Q_1, Q_2 (with the same determinant) respectively. Then $N_1 \cdot N_2$ is represented by $Q_1 \circ Q_2$.

Proof

Let $Q_1(x_1,x_2) = N_2$, $Q_2(x_1,y_1) = N_2$ and $Q_2 \circ Q_2 = a_3$ via B. Then by definition of the composition

$$Q_3(z_1,z_2) = Q_1(x_1,x_2) \cdot Q_2(y_1,y_2) = N_1N_2$$

where $(z_1,z_2) = B \cdot (x_1y_1, x_1y_2, x_2y_1, x_2y_2)^T$. □

DEFINITION 2.1.6.

For p odd prime and n an integer, the *Legendre symbol* is defined by

$$\left(\frac{n}{p}\right) = \begin{cases} +1 & \text{if } x^2 \equiv n \bmod p \text{ is solvable} \\ -1 & \text{if } x^2 \equiv n \bmod p \text{ is not solvable} \\ 0 & \text{if } p \mid n \end{cases}$$

A Legendre symbol has the following properties

(i) $\left(\frac{mn}{p}\right) = \left(\frac{m}{p}\right)\left(\frac{n}{p}\right)$

(ii) $\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)(-1)^{(p-1)(q-1)/4}$ for q odd prime

(iii) $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$

(iv) $\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8}$

Now for each prime divisor p_i of D we define the "function" χ_{p_i} (cf. below) such that

$$\chi_{p_i} : C\ell(D) \to \{-1,1\}$$

$$\text{via} \qquad \chi_{p_i}(Q) := \chi_{p_i}(N) := \left(\frac{N}{p_i}\right)$$

where Q is a form representing a class of $C\ell(D)$, N is an integer represented by Q with $\gcd(N,2D) = 1$ and $\left(\frac{N}{p_i}\right)$ is the Legendre symbol.

Moreover we define

$$\chi_{-4}(Q) := \chi_{-4}(N) = (-1)^{(N-1)/2} \text{ when } D \equiv 0,3,4,7 \pmod 8 \tag{2.4}$$

$$\chi_{8}(Q) := \chi_{8}(N) = (-1)^{(N^2-1)/8} \text{ when } D \equiv 2,0 \pmod 8 \tag{2.5}$$

$$\chi_{-8}(Q) := \chi_{-4}(Q) \cdot \chi_{8}(Q) \qquad \text{when } D \equiv 6 \pmod 8 \tag{2.6}$$

where Q is a form with determinant D and N is aninteger represented by Q with $\gcd(N,2D) = 1$.

The following theorem is due to Mathews ([26], p. 133).

THEOREM 2.1.7

The "functions" χ_p, for each prime divisor p of D, χ_{-4}, χ_8, χ_{-8} are genus characters of $C\ell(D)$.

Proof

(i) First will be shown that $\chi_p(Q)$ is well defined on $C\ell(D)$.

By Lemma 2.1.4, there always exists an number N represented by Q with $\gcd(N,2D) = 1$. Let N_1, N_2 be such that

$$N_1 = Q(k,v) \quad , \quad \gcd(N_1,2D) = 1$$
$$N_2 = Q(m,n) \quad , \quad \gcd(N_2,2D) = 1$$

then $N_1N_2 = Q(k,v)\cdot Q(m,n) = x^2 - Dy^2$ (2.7)

where $x = akm + b(kn + mv) + cvn$, $y = kn - vm$.

Now from (2.7) we have

$$\left(\frac{N_1N_2}{p}\right) = 1 \quad \forall\ p|D,\ p \text{ odd prime}$$

Hence $\left(\frac{N_1}{p}\right) = \left(\frac{N_2}{p}\right)$ which implies $\chi_p(N_1) = \chi_p(N_2)$.

Moreover if $Q_1 \approx Q_2$, then from Theorem 0.1.5 they represent the same numbers, hence $\chi_p(Q_1) = \chi_p(Q_2)$.

Hence has been shown that χ_p is well defined on $C\ell(D)$. Now will be shown that χ_p is an homomorphism.

Let N_1, N_2 are represented by Q_1, $Q_2 \in C\ell(D)$ respectively. Then from Lemma 2.1.5, N_1N_2 is represented by $Q_1 \circ Q_2$. Now we have

$$\chi_p(Q_1)\cdot\chi_p(Q_2) = \left(\frac{N_1}{p}\right)\cdot\left(\frac{N_2}{p}\right) = \left(\frac{N_1N_2}{p}\right) = \chi_p(Q_1 \circ Q_2).$$

Hence χ_p is a character of $C\ell(D)$.

(ii) It will be shown that χ_{-4} is character when $D \equiv 3,7 \pmod 8$. If $D \equiv 3,7 \pmod 8$, then $D \equiv -1 \pmod 4$. Hence from (2.7) we have

$$N_1N_2 = x^2 - Dy^2 \equiv x^2 + y^2 \pmod 4$$

Since N_1, N_2 are odd, one of x,y is odd and the other even. Hence

$$N_1N_2 \equiv 1 \pmod 4 \qquad \text{which implies}$$

$$N_1 \equiv N_2 \pmod 4 \qquad \text{or}$$

$$(-1)^{(N_1-1)/2} = (-1)^{(N_2-1)/2}.$$

Hence $\chi_{-4}(Q) = \chi_{-4}(N_1) = \chi_{-4}(N_2)$ is well defined.

Now let M_1, M_2 be represented by Q_1, Q_2 respectively with $\gcd(M_1,2D) = \gcd(M_2,2D) = 1$. Then

$$\chi_{-4}(Q_1 \circ Q_2) = \chi_{-4}(M_1M_2) = (-1)^{(M_1M_2-1)/2}$$

and $$\chi_{-4}(Q_1)\cdot\chi_{-4}(Q_2) = (-1)^{((M_1-1)+(M_2-1))/2}.$$

One using the fact $M_1 \equiv 1,3 \pmod 4$ and $M_2 \equiv 1,3 \pmod 4$ can show

$$(M_1M_2 - 1)/2 \equiv (M_1 + M_2 - 2)/2 \pmod 2$$

and thus $\chi_{-4}(Q_1 \circ Q_2) = \chi_{-4}(Q_1)\cdot\chi_{-4}(Q_2)$. The other cases can be shown similarly. □

Let $D = 2^t p_1^{2a_1+1} \cdots p^{2at+1} q_1^{2b_1+2} q_s^{2b_s+2}$ and M be the number of distinct prime divisors of D.

TABLE

BASIS FOR GENUS CHARACTERS

Determinant $D = df^2$		d square free		Number of generators
$d \equiv 1 \pmod 4$	$f \equiv 1 \pmod 4$	$\chi_{p_1},\ldots,\chi_{p_r}$	$\chi_{q_1},\ldots,\chi_{q_s}$	M - 1
	$f \equiv 2 \pmod 4$	$\chi_{p_1},\ldots,\chi_{p_r}$	$\chi_{q_1},\ldots,\chi_{q_s},\chi_{-4}$	M - 1
	$f \equiv 0 \pmod 4$	$\chi_{p_1},\ldots,\chi_{p_r}$	$\chi_{q_1},\ldots,\chi_{q_s},\chi_{-4},\chi_8$	M
$d \equiv 3 \pmod 4$	$f \equiv 1 \pmod 2$	$\chi_{-4},\chi_{p_1},\ldots,\chi_{p_r}$	$\chi_{q_1},\ldots,\chi_{q_s}$	M
	$f \equiv 2 \pmod 4$	$\chi_{-4},\chi_{p_1},\ldots,\chi_{p_r}$	$\chi_{q_1},\ldots,\chi_{q_s}$	M - 1
	$f \equiv 0 \pmod 4$	$\chi_{-4},\chi_{p_1},\ldots,\chi_{p_r}$	$\chi_{q_1},\ldots,\chi_{q_s},\chi_8$	M
$d \equiv 2 \pmod 8$	$f \equiv 1 \pmod 2$	$\chi_8,\chi_{p_1},\ldots,\chi_{p_r}$	$\chi_{q_1},\ldots,\chi_{q_s}$	M - 1
	$f \equiv 0 \pmod 2$	$\chi_8,\chi_{p_1},\ldots,\chi_{p_r}$	$\chi_{q_1},\ldots,\chi_{q_s},\chi_{-4}$	M
$d \equiv 6 \pmod 8$	$f \equiv 1 \pmod 2$	$\chi_{-8},\chi_{p_1},\ldots,\chi_{p_r}$	$\chi_{q_1},\ldots,\chi_{q_s}$	M - 1
	$f \equiv 0 \pmod 2$	$\chi_{-4},\chi_8,\chi_{p_1},\ldots,\chi_{p_r}$	$\chi_{q_1},\ldots,\chi_{q_s}$	M

THEOREM 2.1.8

If the first character of each row of the above table is deleted, then the remaining characters are a basis for the genus characters of $C\ell(D)$ for the appropriate type of D.

Proof

See Venkov ([40], p. 143-144). The proof is omitted, since it refers to the theory of teruary quadratic forms. □

Lagarias in [20], gave an algorithm for computation of the characters $\chi_p(Q)$. His algorithm first obtains an integer M represented by Q with gcd(M,2D) = 1; after it evalutes $\chi_p(Q)$ by computing the symbol $(\frac{M}{p})$ using Euler's criterion. To find an integer M represented by Q, coprime to 2D, it uses algorithm 1.3.5. By algorithm 1.3.5 finds a form F = (A,B,C) with gcd (A,2D) = 1. Then

$$F(1,0) = A = M \text{ and } \gcd(M,2D) = 1.$$

We shall give an algorithm which obtains a number N represented by Q with gcd(N,2D) by using the constructive proof of Mathews' lemma 2.1.4, and after computes the symbol $(\frac{N}{p})$. Our algorithm improves only the constant of worst-case complexity bound of Lagarias' algorithm.

ALGORITHM 2.1.9

INPUT : The set $P = \{p_1, p_2, \ldots, p_r\}$ of all odd distinct prime divisors of D and a reduced form Q = (a,b,c) with determinant D.

OUTPUT : $\chi_p(Q)$ $\forall$ $p \in P$ and $\chi_{-4}(Q)$, $\chi_8(Q)$, $\chi_{-8}(Q)$ when appropriate.

Begin

1. $m_1 \leftarrow \gcd(a,c,2D)$;
2. $m_2 \leftarrow \gcd(a,2D)$;
3. $m_3 \leftarrow \gcd(c,2D)$;

4. <Compute $R = \{q_1,\ldots,q_m\} \subseteq P$ where q_i as in (2.1)>;

5. <Compute $T = \{\tau_1,\ldots,\tau_e\} \subseteq P$ where τ_i as in (2.2)>;

6. <Compute $S = \{s_1,\ldots,s_n\} \subseteq P$ where s_i as in (2.3)>;

Comment To compute R,T,S, we do for each $p \in P$, which does not divide m_1 the following. If $p|m_2$, then $p \in R$. If $p|m_3$, then $p \in T$. Else $p \in S$.

$x \leftarrow 1$;

7. For <each element w of $R \cup S$> do

$x \leftarrow x \cdot w$;

$y \leftarrow 1$;

8. For <each element τ of T> do

$y \leftarrow y \cdot \tau$;

9. $N \leftarrow ax^2 + 2bxy + cy^2$;

10. For p = 1 until k do

Begin

Comment The symbol $(\frac{N}{p})$ will be computed using Euler's criterion; $(\frac{N}{p}) \equiv N^{(p-1)/2} \pmod p$

11. $N_1 \leftarrow N \pmod{p_i}$;

12. $\chi_{p_i}(Q) \leftarrow N_1^{(p_i-1)/2} \pmod{p_i}$;

13. end

if $D \equiv 0,3,4,7 \pmod 8$ then

14. $\chi_{-4} \leftarrow (-1)^{(N-1)/2}$;

if $D \equiv 0,2 \pmod 8$ then

15. $\chi_{-8}(Q) \leftarrow (-1)^{(N^2-1)/8}$;

if $D \equiv 6 \pmod 8$ then

16. $\chi_{-8}(Q) \leftarrow (-1)^{(N-1)/2+(N^2-1)/8}$;

end

THEOREM 2.1.10

Algorithm 2.1.9 correctly computes the characters χ_{p_i}, for each p odd prime divisor of D, and χ_{-4}, χ_{-8}, χ_{-8} when appropriate.

Proof

Lemma 2.1.4 shows that N has the required properties and Euler's criterion justifies the computation of the χ'_{p_i} s. □

THEOREM 2.1.11

Algorithm 2.1.9 terminates in $O(\log |D| \; M(\log |D|))$ elementary operations.

Proof

From Theorem 0.2.3 and using (0.6) or (0.13) (since Q is reduced) we have that steps 1-3 require $O(\log \log |D| \; M(\log |D|))$ elementary operations.

Steps 4-6 require at $O(r)$ divisions, where r is the number of distinct prime divisors of D. Since $r = O(\log |D| / \log \log |D|)$, steps 4-6 require $O(\log |D| \; M(\log |D|)/\log \log |D|)$ elementary operations.

Loop 7 and loop 8 require r multiplications at most. Also $|x| < |D|$ and $|y| < |D|$.

Since Q is reduced, step 9 requires $O(M(\log |D|))$ elementary operations and $N = O(|D|^3)$.

From Theorem 0.2.2, step 11 requires $O(M(\log |D|))$ elementary operations.

From Theorem 0.2.4, step 12 requires $O(\log p_i M(\log p_i))$ elementary operations.

Now the loop 10-13 require

$$O(\log |D| M(\log |D|)/ \log |D| + \sum_{p|D} \log p \; M(\log p)) =$$

$$O(\log |D| M(\log |D|))$$

elementary operations.

Finally steps 14-15 require only $O(M(\log |D|))$ elementary operations. Hence the algorithm 2.1.9 terminates in $O(M(\log |D|) \log |D|)$ elementary operations. □

THEOREM 2.1.12

Suppose that χ is a genus character expressed in terms of the basis for genus characters specified in Theorem 2.1.8. For Q a reduced form with dterminant D, one can compute $\chi(Q)$ in $O(\log |D| \; M(\log |D|))$ elementary operations. □

Proof

Suppose that $\{\chi_i\}$is the basis of genus characters and $\chi = \prod_p \chi_i^{a_i}$ with $a_i \in \{0,1\}$. To compute $\chi(Q)$, compute $\chi_i(Q)$

for all the i's using algorithm 2.1.9. In $O(\log |D| M(\log |D|)$ elementary operations and afterwards compute the product of $\chi_i(Q)$'s in $O(\log |D| / \log \log |D|)$ elementary operations, since $\chi_i^{a_i}(Q) = \pm 1$. □

DEFINITION 2.1.13

A form Q (or a class $K \ni Q$) belongs to the *principal genus* if

$$\chi(Q) = 1 \text{ for every genus character.}$$

THEOREM 2.1.14

Suppose that the complete factorization of D is given. It is possible to decide whether or not a form Q with determinant D (non-square if $D > 0$) belongs to the principal genus in $O(\log \|Q\| M(\log \|Q\|))$ elementary operations in general and in $O(\log |D| M(\log |D|))$ elementary operations if Q is a reduced form.

Proof

First by using the algorithm 0.3.2 or 0.3.6 we reduce Q to Q'. This from Theorem 0.3.4 or 0.3.7 can be done in $O(\log \|Q\| M(\log \|Q\|))$ elementary operations. Afterwards we compute the basis characters $\chi_i(Q')$ in $O(\log |D| M(\log |D|))$ elementary operations. If $x_i(Q') = 1$ for every i, then Q belongs to the principal genus.

Now easily for every form Q, we have

$$||Q|| > \sqrt{|D|/2}$$

Hence the algorithm terminates in $O(\log ||Q|| \, M(\log ||Q||))$ elementary operations.

If Q is reduced, then by (0.7) or (0.13) we have

$$\log ||Q|| = O(\log |D|)$$

and thus in this case the algorithm terminates in $O(\log |D| M(\log |D|))$ elementary operations. □

2.2. DIRICHLET'S L-SERIES

First an extension of the Legendre symbol is defined

DEFINITION 2.2.1

If $n > 0$ odd and $n = p_1^{a_1} \ldots p_\tau^{a_\tau}$, then for an integer m with $\gcd(m,n) = 1$ let $(\frac{m}{n}) := (\frac{m}{p_1})^{a_1} \ldots (\frac{m}{p})^{a_\tau}$, where $(\frac{m}{p_i})$ is the Legendre symbol. Then $(\frac{m}{n})$ is called the Jacobi symbol.

The following are some properties of the Jacobi symbol (see Ayoub [5], p. 289).

(i) $(\frac{a}{mn}) = (\frac{a}{m})\,(\frac{a}{n})$ for m,n odd and $\gcd(a,m) = \gcd(a,n) = 1$.

(ii) $(\frac{ab}{n}) = (\frac{a}{n})\,(\frac{b}{n})$ for n odd and $\gcd(a,n) = \gcd(b,n) = 1$.

(iii) $(\frac{a}{n})\,(\frac{n}{a}) = (-1)^{(n-1)(a-1)/4}$ for a,n odd.

(iv) $(\frac{-1}{n}) = c-1)^{(n-1)/2}$ for n odd.

(v) $(\frac{2}{n}) = (-1)^{(n^2-1)/8}$ for n odd.

The restriction that n to be odd of the Jacobi symbol $(\frac{m}{n})$ makes necessary an extension to permit even n.

DEFINITION 2.2.2

The Kronecker symbol for an integer d is defined: see Ayoub [5], p. 290).

(a) $\left(\frac{d}{n}\right) := \begin{cases} \text{Jacobi symbol} & \text{if } \gcd(d,n) = 1 \text{ for } n \text{ odd positive} \\ 0 & \text{if } \gcd(d,n) \neq 1 \text{ for } n \text{ odd positive} \end{cases}$

(b) $\left(\frac{d}{2}\right) := \begin{cases} 0 & \text{if } 2 \mid d \\ +1 & \text{if } d \equiv 1 \bmod 8 \\ -1 & \text{if } d \equiv 5 \bmod 8 \end{cases}$

(c) $\left(\frac{d}{mk}\right) := \left(\frac{d}{m}\right)\left(\frac{d}{k}\right)$ for m,k positive.

This extension of the Jacobi symbol respects properties (i)-(v) above.

Now Dirichlet's L-function will be defined as an expression of Kronecker symbols.

DEFINITION 2.2.3

Let $\chi_d(n) = \left(\frac{d}{n}\right)$ the Kronecker symbol. Then

$$L(s,\chi_d) = \sum_{n=1}^{\infty} \frac{\chi_d(n)}{n^s} \quad \text{for } s \in \mathbb{C} \text{ (complex numbers)}$$

is Dirichlet's L-function.

Some of our complexity analyses for later algorithms are based on the assumption of the truth of the *Generalized Riemann Hypothesis* (G.R.H). The G.R.H. asserts that the zeros of $L(s,\chi_d)$ in the critical strip $0 < \mathrm{Re}(s) < 1$ all lie on the line $\mathrm{Re}(s) = 1/2$ where $\mathrm{Re}(s)$ is the real part of s.

Some basic results about $L(1,\chi_d)$ are given below. There are important formulae, due to Dirichlet, relating L-functions and class numbers, and algorithms for computing $L(1,\chi_d)$ are given in Chapter 4.

THEOREM 2.2.3. EULER'S PRODUCT

$$L(1,\chi_d) = \prod_{p \text{ prime}}^{\infty} \left(\frac{p}{p-\chi_d(p)}\right)$$

Proof

See Apostol ([4], p. 231). □

THEOREM 2.2.4

$$0 < L(1,\chi_d) < 3 \ln |d|.$$

Proof

See Ayoub ([5], p. 338). □

3.1. INFRASTRUCTURE OF Cℓ(D) WITH D POSITIVE NON-SQUARE

The number λ of reduced forms in a class of the form class group Cℓ(D) is usually quite big ($\lambda \approx c\sqrt{D}$, where c is a positive constant, see 3.1.19 and remark below it). Hence there is no algorithm running in polynomial time to compute the set of reduced forms of a class of Cℓ(D). Moreover there is no known efficient algorithm for deciding whether two indefinite forms Q and F belong to the same class. The only known strategy deciding equivalence of indefinite forms is the following:

(i) Reduce F and Q to F' and Q' respectively.

(ii) Search if Q' belongs to the set S of reduced forms equivalent to F'.

(iii) If $Q' \in S$, then $Q \approx F$, if $Q' \notin S$, then $Q \not\approx F$.

Gauss' algorithm for computing the reduced forms in an equivalence class and deciding equivalence of forms is described in Section A below. This algorithm runs in time $O(D^{1/2+\varepsilon})$. An $O(D^{1/4+\varepsilon})$ algorithm, due to Lenstra and Shanks, for computing the regulator appears in Section D.

A. GAUSS ALGORITHM

DEFINITION 3.1.1.

Two forms F = (a,b,c), G = (A,B,C) with the same determinant D > 0 and

$$A = c$$

$$b + B \equiv 0 \bmod A$$

are said to be *neighbours*. Moreover G is a neighbour to F by *last part* and F is neighbour to G by *first part*.

It is not difficult to see that $F \approx G$ and

$$F \to G \text{ via } \begin{pmatrix} 0 & -1 \\ 1 & k \end{pmatrix} \text{ with k such that: } b+B = kA.$$

THEOREM 3.1.2

Every indefinite reduced form has exactly one reduced form neighbour by either part.

Proof

Let $F = (a,b,c)$ be a reduced indefinite form and $G = (c,B,*)$ be a form with the same determinant and B satisfies

$$B + b \equiv 0 \pmod{c} \qquad \text{and} \qquad (3.1)$$

$$\sqrt{D} - |c| < B < \sqrt{D}\,. \qquad (3.2)$$

Let $B + b = kc$. Then k is uniquely determined by (3.1), (3.2).

Since F is reduced, from (0.12) replacing a by $(b^2-D)/c$ we have

$$\sqrt{D} - |c| < b < \sqrt{D} \qquad (3.3)$$

From (3.2) and (3.3) we have

$$kc = B + b = (\sqrt{D} + B - |c|) + (b - \sqrt{D} - |c|) > 0$$

Hence

$$kc > |c| \tag{3.4}$$

Moreover using (3.2), (3.3), (3.4) we have

$$2B = (\sqrt{D} - b) + (B - \sqrt{D} + |c|) + (kc - |c|) > 0 \tag{3.5}$$

Hence from (3.2) and (3.5) we have

$$0 < B < \sqrt{D} \tag{3.6}$$

Moreover from (3.2), (3.4) we have

$$\sqrt{D} + B - |c| = (\sqrt{D} - b) + (kc - |c|) > 0 \tag{3.7}$$

Hence from (3.2), (3.7), we have

$$\sqrt{D} - B < |c| < \sqrt{D} + B$$

and from (3.6)

$$|\sqrt{D} - |c|| < B < \sqrt{D}$$

which implies that G is reduced.

Hence G is the unique neighbouring reduced form to F by last part, since if Q is a reduced form neighbour to F by last part, then Q will satisfy (3.1), (3.2) and thus Q = G.

Similarly it can be shown that F has exactly one neighbour by last part. □

EXAMPLE

The form (3,8,-5) with determinant 79 has neighbour by last part (-10, 7, 3) and neighbour by first part (-5, 7, 6).

THEOREM 3.1.3

Suppose $F = F_o, F_1, \ldots, F_i, F_{i+1}, \ldots$ are indefinite reduced forms with the same determinant and F_i is neighbour to F_{i+1} by first part. Then there exists an even positive integer λ such that

$$F = F_\lambda.$$

Proof

From Theorem 3.1.2, there always exists one reduced neighbour to a reduced form and since the number of the reduced forms of fixed determinant is finite (from (0.13)), there are integers $m > 0$, $\lambda > 0$ such that

$$F_m = F_{m+\lambda}.$$

If $F_m = F_{m+\lambda}$ then from Theorem 3.1.2 we have

$$F_{m-1} = F_{m+\lambda-1}.$$

Hence applying this recursively, we have $F = F_o = F_\lambda$.

If $Q = (a,b,c)$ is reduced form with determinant D, then $b^2 - D < 0$ since $b < \sqrt{D}$. Hence $ac = b^2 - D < 0$.

Let $F_i = (a_i, b_i, c_i)$, $F_{i+1} = (a_{i+1}, b_{i+1}, c_{i+1})$. Then

$$a_i a_{i+1} = a_i c_i < 0. \tag{3.8}$$

since F_i, F_{i+1} are neighbours and F_i reduced.

Now since the first terms of two reduced neighbouring forms have opposite sign, we conclude that the period λ is even. □

The sequence of forms $\{F = F_0, F_1, \ldots, F_\lambda = F\}$, where F_i's and λ as in Theorem 3.1.3, is called the *cycle of* F and λ *the period* of the cycle of F.

THEOREM 3.1.4.

Two, reduced forms are equivalent if and only if one belongs to the cycle of the other. □

Gauss ([16], A.193) gave a lengthy proof of the above theorem but Dirichlet (see Smith [38], A. 93) observing connections between the sequence of neighbouring forms of a reduced indefinite form (a,b,c) with determinant D and the continued fraction expansion of $\frac{\sqrt{D}-b}{a}$, gave a shorter proof. For an alternative proof see Beynon and Iliopoulos [7].

Let F = (a,b,c) be a reduced indefinite form with determinant D and

$$F = F_0 \to F_1 \to \ldots \to F_{2\lambda}=F \qquad (3.9)$$

$$\text{with } F_i \to F_{i+1} \quad \text{via } S_i = \begin{pmatrix} 0 & -1 \\ 1 & k_i \end{pmatrix}$$

be the sequence of neighbouring forms of F.

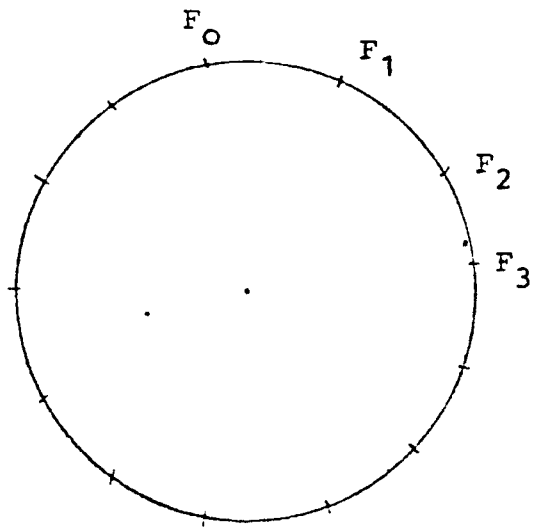
Dirichlet proved that $\{k_1, k_2, \ldots, k_{2\lambda}\}$ is the continued fraction expansion of $\frac{\sqrt{D}-b}{a}$. Hence one can compute the c.f.e of $(\sqrt{D} - b)/a$ by computing the sequence of neighbouring forms.

EXAMPLE

The cycle of the form (5, 8, -3) is:

(5, 8, -3), (-3,7,10), (10,3,-7), (-7,4,9), (9,5,-6),(-6,7,5). □

Usually the sequence of neighbouring forms of a reduced form F, is represented by placing them on a circle



By 3.1.4 the cycle of F contains all the reduced forms of the class to which F belongs. If F belongs to the principal class, then the cycle of F is called the *principal cycle*. In this section let $I = (1,b_o,c_o)$ denote a form of the principal cycle of $C\ell(D)$, with $b_o = \lfloor \sqrt{D} \rfloor$.

Gauss' method for computation of the set of reduced forms of a class, begins with a reduced form F of the class and does a series of transformations

$$F = F_o \rightarrow F_1 \rightarrow F_2 \rightarrow \ldots \rightarrow F_i \rightarrow F_{i+1}$$

where F_{i+1} is the unique reduced form neighbour to F_i by first part. For some even integer $\lambda > 0$, it finds a form $F_\lambda = F$. By Theorem 3.1.4 the set $\{F_1, F_2, \ldots, F_\lambda\}$ is the set of the reduced forms of the class.

ALGORITHM 3.1.5

INPUT : A reduced form $Q = Q_o = (a_o, b_o, c_o)$ with determinant $D > 0$, non-square.

OUTPUT : All the forms belonging to the cycle of Q and the period λ of the cycle.

Begin

$i \leftarrow 1$;

Repeat

Choose a k_i such that: $\sqrt{D} - |c_{i-1}| < -b_{i-1} + k_i c_{i-1} < \sqrt{D}$

$a_i \leftarrow c_{i-1}$;

$b_i \leftarrow -b_{i-1} + k_i c_{i-1}$;

$c_i \leftarrow (b_i^2 - D)/a_i$;

$Q_i \leftarrow (a_i, b_i, c_i)$;

$i \leftarrow i+1$;

Until $Q_{i-1} = Q$

Return $\{Q_i\}$, $\lambda = i-2$;

end. □

Assuming the truth of the fact that the period λ of a cycle of an indefinite form with determinant D is $O(\sqrt{D})$ (see Theorem 3.1.16), we have the following theorem.

THEOREM 3.1.6

Algorithm 3.1.5 yields correctly the period and all the reduced forms of the cycle of Q in $O(\sqrt{D} \log D\, M(\log D))$ elementary operations.

Proof

Since Q_i is reduced for every i, from (0.13) we have $\|Q_i\| < 2\sqrt{D}$. Hence each iteration of the loop requires only $O(M(\log D))$ elementary operations for multiplications. Since the period is of $O(\sqrt{D})$, it can be computed in $O(\sqrt{D}\, M(\log D))$ elementary operations and all the reduced forms of the cycle of Q can be written down in $O(\sqrt{D} \log D\, M(\log D))$ elementary operations (since $\log \|Q_i\| = O(\log D)$). □

Gauss method for deciding equivalence of the indefinite forms F and G, reduces them to F' and G' respectively and computes the cycle of F'. If G' belongs to the cycle of F', then by Theorem 3.1.4 $F \approx Q$. If not, then $F \not\approx G$. Hence if F,G are reduced, then one can decide equivalence of F and G with the above method in $O(\sqrt{D} \log D\, M(\log D))$ elementary operations by Theorem 3.1.6.

B. FUNDAMENTAL UNIT

DEFINITION 3.1.7

If (T_o, U_o) is the smallest non-trivial integral solution of the equation $T^2 - DU^2 = 1$, then $\eta = T_o + \sqrt{D}\, U_o$ is said to be the *fundamental unit* of the order $\mathcal{O}_{2f}(d)$ with $D = f^2 d$ and

$R = \ell n(\eta)$ is said to be the *regulator* of the order $\mathcal{D}_{2f}(d)$.

Remark

The terms fundamental unit and regulator are used here in the "strict" sense. In algebraic number theory, the fundamental unit $\bar{\eta}$ in the "ordinary" sense of an order with discriminant Δ is

$$\bar{\eta} = \tfrac{1}{2}(T_o + \sqrt{\Delta}\, U_o), \quad \text{(see [35], p. 53)}$$

where (T_o, U_o) is the smallest non-trivial integral solution of the equations

$$T^2 - \Delta U^2 = \pm 4$$

and the regulator $\bar{R}$ is

$$\bar{R} = \ell n\ \bar{\eta} \qquad \text{(see [35], p.56).}$$

The fundamental unit η in the "strict" sense of an order with discriminant Δ is $\eta = \tfrac{1}{2}(T_o + \sqrt{\Delta}\, U_o)$ where (T_o, U_o) is the smallest non-trivial solution of the equation $T^2 - \Delta U^2 = 4$ and the regulator $R = \ell n\ \eta$ (see [25], p. 10, 21)

Now we shall show that our definition of fundamental unit is essentially the same as the fundamental unit in the "strict" sense in algebraic number theory. If (t,u) is a solution of the equation $T^2 - DU^2 = 1$, then $(2t,u)$ is a solution of the equation $T^2 - 4DU^2 = 4$. Moreover if (T_o, U_o) is the smallest non-trivial integral solution of $T^2 - DU^2 = 1$, then one can see that $(2T_o, U_o)$ is the smallest non-trivial

integral solution of $T^2 - 4DU^2 = 4$. Hence the fundamental unit (in the "strict" sense) of an order with discriminant $\Delta = 4D$ is

$$\eta = \tfrac{1}{2}(2T_o + \sqrt{4D}\, U_o) = T_o + \sqrt{D}\, U_o .$$

Since we deal only with orders having discriminant $\Delta = 4D$, (see remark below, Theorem 1.2.2), definition 3.1.7 is the same as in algebraic number theory.

Similarly it can be shown that if (T_o,U_o) is the smallest non-trivial integral solution of the equation $T^2 - DU^2 = -1$, then $(2T_o,U_o)$ is the smallest non-trivial integral solution of the equation $T^2 - 4DU^2 = -4$.

Methods of solving the equations $T^2 - DU^2 = 1$ and $T^2 - DU^2 = -1$ are given below; thus one can compute the fundamental unit in both the "strict" and the "ordinary" sense.

One method of computation of the fundamental unit and the regulator is based on the computation of the cycle of a reduced indefinite form with determinant D. It relies on the fact that the least non-trivial integral solution (T_o,U_o) of the equation can be expressed in terms of the coefficient of the matrix $S = \prod_{i=1}^{n} S_i$ where S_i are unimodular matrices such that

$$F_i \rightarrow F_{i+1} \text{ via } S_i$$

where $\{F = f_o, F_1, F_2, \ldots, F_n = F_o\}$ is the cycle of a reduced indefinite form with determinant D.

The formula to obtain (T_o, U_o) from the coefficients of S is given in the following theorem.

THEOREM 3.1.8.

Suppose that F is reduced indefinite form of $C\ell(D)$ let $F_o, F_1, \ldots, F_{2\lambda} = F$ be the cycle of F and $F_{i-1} \to F_i$ via S_i. If

$$S = \prod_{i=1}^{2\lambda} S_i = \begin{pmatrix} x & y \\ z & t \end{pmatrix}$$

then $(T_o, U_o) = (|ax + bz)/a|, |z/a|)$ is the smallest positive non-trivial integral solution of the equation $T^2 - DU^2 = 1$.

Proof

See Gauss ([16], A.198) and Beynon and Iliopoulos [7]. □

One can compute the matrix S of the above theorem, using algorithm 3.1.5. and (3.9) to compute the matrices S_i for $1 < i < 2\lambda$, and thus evaluate the fundamental unit. Sometimes the size of the fundamental unit can be enormous ($\approx e^{\sqrt{D}}$, see remark below Proposition 3.1.19, also [35], p. 54). In view of this fact, there is in general no more efficient method (up to a constant factor) for computing the fundamental unit, but better methods for computing the regulator exist (see below).

C. DISTANCES AND REGULATOR

The description and analysis of Shanks' algorithm (see [36]) for computation of the regulator makes use of a notion of "distance" between forms which was used informally by Shanks

and described formally by Lenstra [25]. The significant features and properties of Lenstra's distance function are briefly summarized below; for detailed definitions and proofs see Lenstra [25] and Schoof [33].

DEFINITION 3.1.9.

A form F is $\mathbb{Z}$-*equivalent* to a form G if there exists a matrix

$$S = \begin{pmatrix} 1 & m \\ 0 & 1 \end{pmatrix}, \quad m \text{ integer}$$

such that

$$F \to G \text{ via } S.$$

It can be shown that being $\mathbb{Z}$-equivalent is an equivalence relation, stronger than the usual equivalence of forms.

Now let $\mathbb{F}(D)$ denote the set of the $\mathbb{Z}$-equivalence classes of properly primitive forms of fixed determinant D.

THEOREM 3.1.10.

Gaussian composition is well-defined over the $\mathbb{Z}$-equivalence classes and the set $\mathbb{F}(D)$ under Gaussian composition forms an abelian group.

Proof

See [25], p. 13). □

Remark

One can see that Dirchlet's method of composition is unsuitable in this case, because if (a,b,c) is a form of a class C_1 of $\mathbb{F}(D)$, then all the forms of the class are of the shape (a,b',c') and thus if (A,B,C) is a form of another class C_2 of $\mathbb{F}(D)$ with $\gcd(A,a) \neq 1$, then there does not exist a pair of concordant forms Q_1, Q_2 such that $Q_1 \in C_1$ and $Q_2 \in C_2$.

COROLLARY 3.1.11

Let $\mathbb{G}(D)$ denote the set of $\mathbb{Z}$-equivalence classes of forms belonging in the principal class of $C\ell(D)$. Then the set $\mathbb{G}(D)$ under Gaussian composition forms an abelian group.

Proof

It is easy to show that $\mathbb{G}(D)$ is a subgroup of $\mathbb{F}(D)$ and thus by 3.1.10, $\mathbb{G}(D)$ is an abelian group. □

Lenstra's distance function d is defined in terms of an injective group homomorphism

$$\delta:\mathbb{G}(D) \to (\mathbb{R}/R\,\mathbb{R} \oplus \mathbb{Z}/2\,\mathbb{Z},+)$$

where R is the regulator of the order $\mathcal{O}_{2f}(K)$ and $D = f^2K$ with K square free. For the precise definition of δ - which is omitted here - see [25], p. 15 and [33], p. 183 (The map δ has been introduced since a knowledge of how d is defined from δ is conceptually helpful and unifies the results cited from [25] below). The distance between two forms F_1, F_2 of $\mathbb{G}(D)$ is defined to be

$d(F_1,F_2)$ = the first component of $\delta(F_1 \circ F_2^{-1})$

Now let

$$J_a = \begin{cases} -|a|/2 < x \leq |a|/2 & \text{if } |a| > 2\sqrt{D} \\ \sqrt{D} - |a| < x < \sqrt{D} & \text{if } |a| < 2\sqrt{D} \end{cases}$$

Each class C of $\mathbb{F}(D)$ contains exactly one form (a,b,c) such that $b \in J_a$., and this form is reduced if and only if C contains a reduced form.

It will be convenient in the sequel to identify a class in $\mathbb{F}(D)$ (or $\mathbb{G}(D)$) with the unique form (a,b,c) with $b \in J_a$ which it contains, and to use $\circ$ to denote the operation of Gaussian composition on this set of representatives.

The map $\sigma:\mathbb{F}(D) \to \mathbb{F}(D)$ is defined by

$$\sigma((a,b,c)) = (c,b',*) \text{ with } b \in J_a \text{ and } b' \in J_c$$

It is easy to see that the form $\sigma(F)$ is a neighbour by last part to F and if F is reduced, then $\sigma(F)$ is the unique reduced form neighbour to F by last part.

Let $\mathcal{R}$ to be a set of properly primitive reduced forms of determinant $D > 0$ (non-square). The map

$\sigma^*:\mathbb{F}(D) \to \mathcal{R}$ is defined by

$\sigma^*(F) = \sigma^k(F)$, $F \in \mathbb{F}(D)$

where k is the smallest positive integer such that $\sigma^*(F)$ is reduced form. Observe that the computation of the function σ^* can be done by algorithm 0.3.6, since every transformation $Q_i \to Q_{i+1}$ (see 0.3.6) is an application of the function σ.

Now we shall give some results about distance. The

following results hold for forms in $\mathbb{G}(D)$ but generally are true for forms in $\mathbb{F}(D)$ (see Schoof [33]).

THEOREM 3.1.12

Suppose that $F = (a,b,c) \in \mathbb{G}(D)$. Then

$$d(F,\sigma(F)) = \tfrac{1}{2} \ln \left| \frac{b+\sqrt{D}}{b-\sqrt{D}} \right| \pmod{R}$$

Proof

See Lenstra [25], p. 18. □

We note that

$$d(\sigma(F),F) = -d(F,\sigma(F)) \tag{3.10}$$

which follows from the fact that δ is homomorphism and $\delta(G^{-1}) = -\delta(G)$.

THEOREM 3.1.13

Suppose that $F = (a_o,b_o,c_o) \in \mathbb{G}(D)$, $\sigma^n(F) = G$ and $\sigma^i(F) = (a_i,b_i,c_i)$ for $1 < i < n$. Then

$$d(F,G) = \sum_{i=0}^{n-1} \tfrac{1}{2} \ln \left| \frac{b_i + \sqrt{D}}{b_i - \sqrt{D}} \right| \pmod{R}$$

Proof

We have

$$d(F,G) = \delta(F\circ G^{-1}) = \delta(F\circ(\sigma(F))^{-1}\circ\sigma(F)\circ(\sigma^2(F))^{-1}\circ\sigma^2(F) \circ \ldots \circ \sigma^{n-1}(F)) \circ G^{-1}) = \sum_{i=0}^{n-1} d(\sigma^i(F),\sigma^{i+1}(F)). \quad \square$$

THEOREM 3.1.14

Suppose that $I = (1,b_o,c_o)$, $F,G \in \mathfrak{G}(D)$. Then

$$d(I,GoQ) = d(I,G) + d(I,Q) \pmod{R}$$

Proof

Using the fact that δ is homomorphism, we have

$$d(I,GoQ) = \delta(IoG^{-1} \circ Q^{-1}) = \delta(IoG^{-1} \circ IoQ^{-1})$$

$$= d(I,G) + d(I,Q).$$

since I is the identity element of $\mathfrak{G}(D)$. □

COROLLARY 3.1.15

Suppose that $I = (1,b_o,c_o)$ with $b_o \in J_1$ and $F,G \in \mathfrak{G}(D)$. Then

$$d(I,\sigma^*(FoG)) = d(I,F) + d(I,G) + d(FoG,\sigma^*(FoG)).$$ □

The following proposition yields an upper and a lower bound for the function of distance.

PROPOSITION 3.1.16

Suppose that $F \in \mathfrak{G}(D)$ and F is reduced. Then

(i) $d(F,\sigma(F)) < \frac{1}{2} \ell n\ 4D$

(ii) $d(F,\sigma(F)) + d(\sigma(F), \sigma^2(F)) > \ell n\ 2.$

Proof

(i) Let $F = (a,b,c)$. Then since F is reduced we have

$$|\sqrt{D} - |a|\,| < b < \sqrt{D}$$

Hence

$$d(F,\sigma(F)) = \tfrac{1}{2} \ell n \left| \frac{b + \sqrt{D}}{b - \sqrt{D}} \right| = \tfrac{1}{2} \ell n \frac{(b + \sqrt{D})^2}{|ac|} <$$

$$< \tfrac{1}{2} \ell n \, (b + \sqrt{D})^2 < \tfrac{1}{2} \ell n \, 4D.$$

(ii) See Lenstra [25], p. 18-19. □

PROPOSITION 3.1.17

Suppose that $F \in \mathfrak{C}(D)$. Then

$$d(F,\sigma^*(F)) < \tfrac{1}{2} \ell n \, (1 + (1 + \sqrt{5}) \sqrt{D})$$

Proof

See Lenstra [25], p. 20. □

THEOREM 3.1.18

Suppose that $I = (1,b_o,c_o)$ with $b_o \in J_1$. Then the regulator of the orders $\mathcal{O}_{2f}(k)$ with $D = f^2k$ and k square free is given by

$$R = \sum_{i=0}^{\lambda-1} d(\sigma^i(I), \sigma^{i+1}(I)) \qquad (3.11)$$

where λ is the period of the principal cycle.

Proof

See Lenstra [25], p. 19. □

If the forms $I = F_o, F_1, \ldots, F_\lambda = I$ of the principal cycle are placed on a cycle in such a way that F_i and F_{i+1} have distance $d(F_i, F_{i+1})$, then from (3.11) the circumference of the cycle is given by

$$R = \sum_{i=1}^{\lambda} d(F_{i-1}, F_i) \tag{3.12}$$

The following proposition yields an upper bound for the regulator the period of a cycle and the fundamental unit.

PROPOSITION 3.1.19.

Suppose that η and R are the fundamental unit and the regulator respectively of an order $\mathcal{O}_{2f}(d)$ with $D = f^2 d$ and λ is the period of the principal cycle of $C\ell(D)$. Then

(i) $R < 6\sqrt{D}\ \ell n\ 4D$

(ii) $\lambda = O(\sqrt{D})$

(iii) $\log\ \eta = O(\sqrt{D}\ \log D)$.

Proof

(i) From (4.19) and Theorem 2.2.4 we have

$$R < 6\sqrt{D}\ \ell n\ 4D \tag{3.13}$$

(ii) From (3.12) and Proposition 3.1.16(i) we have

$$R = \sum_{i=1}^{\lambda} d(F_{i-1},F_i) < \lambda(\tfrac{1}{2} \ell n\ (4D))$$

and thus from (3.13) we have

$$\lambda = O(\sqrt{D}) \qquad (3.14)$$

(iii) Follows from (i) and the fact $R = \ell n\ \eta$. □

Lagarias in [21] (p. 486) had an example (known to Dirichlet) of an infinite sequence of values of D for which the continued fraction expansion of $\sqrt{D}$ is greater than $c\sqrt{D}\ (\log D)^{-1}$, where c positive constant. Hence since there is one to one correspondence between the forms of the principal cycle of $C\ell(D)$ and the terms in the period of continued fraction expansion of $\sqrt{D}$, one can see that the bound (3.14) cannot be improved except perhaps by a factor $(\log D)^{-1}$. Now one can see using Proposition 3.1.16(ii) that

$$R = \sum_{i=1}^{\lambda} d(F_{i-1},F_i) > \frac{\lambda}{2} \ell n\ 2.$$

Similarly the bounds (i), (iii) of Proposition 3.1.19 are seen to be optimal.

Now a theorem is given which shows that in the principal cycle there exists an ambiguous form (different to I) and its distance from I is R/2, where R is the regulator.

THEOREM 3.1.20.

Suppose that $I = F_o, F_1, \ldots, F_{2\lambda} = I$ are the forms of the principal cycle of $C\ell(D)$. Then (i) the form F_λ is ambiguous

and

$$d(I,F_\lambda) = R/2$$

where R is the regulator.

(ii) The forms I and F_λ are the only ambiguous forms of the principal cycle.

Proof

(i) Let $F_o = (1,b_o,c_o) = F_{2\lambda} = I$. Then we have $F_{2\lambda-1} = (c_o,b_o,1)$.

Now let $F_1 = (c_o,b_1,c_1)$ and $F_{2\lambda-2} = (a,b,c_o)$. Using Proposition 3.1.2 and the fact that

$$b + b_1 \equiv b + b_o \equiv 0 \pmod{c_o}$$

we have $F_{2\lambda-2} = (c_1,b_1,c_o)$

And generally by induction on k can be shown that

$$F_k = (a_k,b_k,c_k) \iff F_{2\lambda-k-1} = (c_k,b_k,a_k) \qquad (3.15)$$

Now for $k = \lambda-1$ we have

$$F_{\lambda-1} = (a_{\lambda-1},b_{\lambda-1},c_{\lambda-1}) \iff F_\lambda = (c_{\lambda-1},b_{\lambda-1},a_{\lambda-1}).$$

Now since $F_{\lambda-1}$ is neighbour to F_λ by first part, we have

$$b_{\lambda-1} + b_{\lambda-1} = 2b_{\lambda-1} \equiv 0 \pmod{c_{\lambda-1}}$$

Hence F_λ is ambiguous.

Now from (3.11) we have

$$d(I,F_\lambda) = \sum_{k=0}^{\lambda-1} \tfrac{1}{2} \ln \left| \frac{b_k + \sqrt{D}}{b_k - \sqrt{D}} \right|$$

and

$$d(F_\lambda,F_{2\lambda}) = \sum_{k=\lambda}^{2\lambda} \tfrac{1}{2} \ln \left| \frac{b_k + \sqrt{D}}{b_k - \sqrt{D}} \right|$$

From (3.15) we have that $b_k = b_{2\lambda-k-1}$ and thus

$$d(I,F_\lambda) = d(F_\lambda,F_{2\lambda})$$

Hence we have

$$d(I,F_\lambda) = R/2$$

since

$$d(I,F_\lambda) + d(F_\lambda,F_{2\lambda}) = R$$

(ii) See Gauss ([16], A.187). □

Now applying algorithm 3.1.5 to the form $I = (1,b_o,c_o) = F_o$, one can compute the forms $F_o,F_1,\ldots,F_\lambda$ until to find an ambiguous form F_λ for some integer λ. From the middle terms of the F_i's one can compute $d(I,F_\lambda)$ and thus the regulator $R = 2(I,F_\lambda)$ using Theorem 3.1.20. This method has running time of $O(\sqrt{D}\, M(\log D) \log D)$ (see 3.1.6 and 3.1.19).

D. LENSTA-SHANKS ALGORITHM

Shanks [35] gave an algorithm for computation of the regulator which was formulated later by Lenstra [25]. The Lenstra-Shanks algorithm is the following:

1. Let $F_o = I = (1,b_o,c_o)$ (I is a reduced form of $C\ell(D)$). Compute the sequence of forms $F_1,F_2,\ldots,F_k$, where $F_i = (a_i,b_i,c_i) = \sigma(F_{i-1})$ in conjunction with the distances $d(I,F_i)$, $(d(I,F_i) = d(I,F_{i-1}) + d(F_{i-1},F_i))$ until either an ambiguous form F_j is encountered or for some k

$$d(I,F_k) > d_o = (6\sqrt{D}\ \ell n\ 4D)^{1/2}$$

(This is taking "baby-steps" around the principal cycle in Shanks terminology).

2. If an ambiguous F_j form is found, then $R = 2\,d(I,F_j)$ (using Theorem 3.1.20), otherwise:

3. Compute the forms F_o, $F_{-1},F_{-2},\ldots,F_{-k}$, where $F_i = \sigma(F_{i-1})$ on the distances $d(I,F_{-i})$ (using the relations

$$F_{i-1} = (c_i,b_i,a_i) \text{ and } d(I,F_{1-i}) = -d(I,F_i)$$

which follow from (3.15) and (3.10)).

4. Compute $G = \sigma^*(F_k^2)$ and $d(I,G)$ (using the algorithm of Corollary 1.1.11 and Corollary 3.1.15).

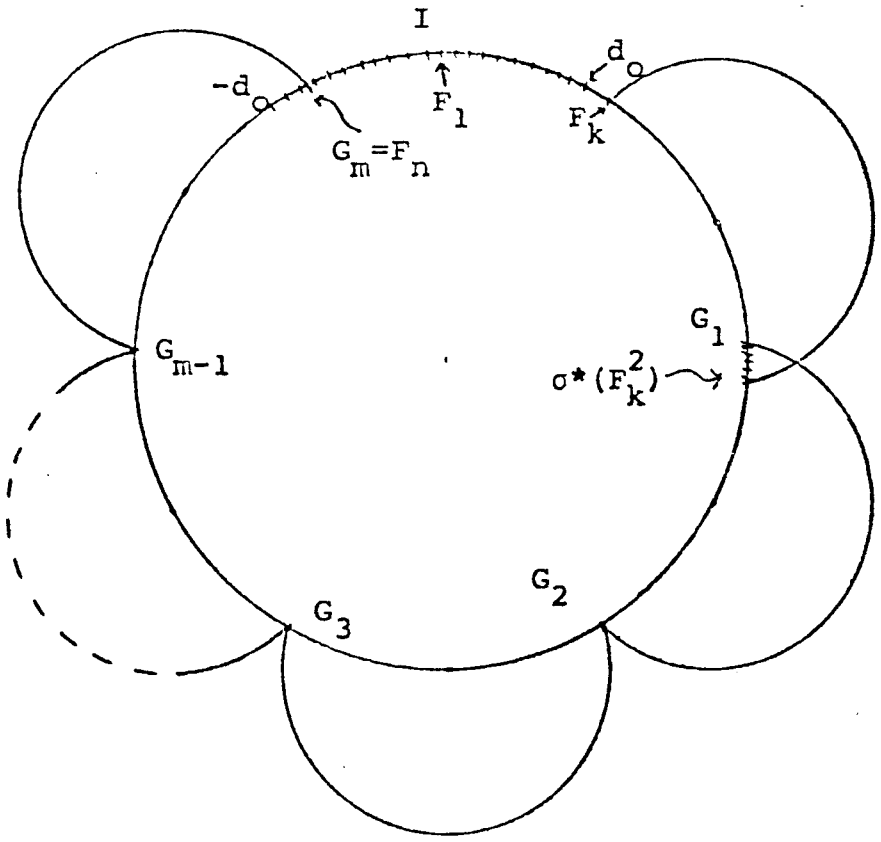
5. By searching into the neighbours of G find a form G_1 such that

$$2d_o - \tfrac{1}{2}\ell n\,(1+(1+\sqrt{5})\sqrt{D}\,) \quad \ell n 4D < d(1,G_1) < 2d_o - \tfrac{1}{2}\ell n(1+(1+\sqrt{5})\sqrt{D} \qquad (3.16)$$

6. Compute the sequence of forms $G_2, G_3, \ldots$ (using the algorithm of Corollary 1.1.11), where $G_{j+1} = \sigma^*(G_j \circ G_1)$ in conjunction with $d(I,G_j)$ ($d(I,G_j) = d(I,G_1) + d(I,G_j) + d(G_j \circ G_1, G_{j+1})$) until an integer m such that $G_m = F_n$ with $|n| < k$ is encountered. (This is taking "giant-steps" around the principal cycle in Shanks terminology).

7. $R = d(I,G_m) - d(I,F_n)$.

The steps of the algorithm may be illustrated diagramatically thus:



It can be shown that if G_i satisfy (3.16), then

$$d_o < d(G_i, G_{i+1}) < 2\,d_o.$$

Hence one of the jumps will fall into the distance $[-d_o, d_o]$ since each jump covers a distance less than $2\,d_o$. Moreover this will happen after at most $O(R/d_o) \approx O(D^{1/4})$ jumps, since each jump is greater than d_o.

ALGORITHM 3.1.21

INPUT : A non-square integer $D > 0$

OUTPUT : The regulator R of $\mathcal{O}_{2f}(k)$ with $D = f^2k$ and k square-free.

Begin

1. $b_o \leftarrow L\sqrt{D}$;
2. $c_o \leftarrow b_o^2 - D$;
3. $F_o \leftarrow (1, b_o, c_o)$;
4. $i \leftarrow 0$;
5. $d(I, F_o) \leftarrow 0$;
6. $d_o \leftarrow (6\sqrt{D}\ \ln\ 4D)^{1/2}$;
7. While $d(I, F_k) < d_o$ do

   Begin

8. $F_{k+1} \leftarrow \sigma(F_k)$;

   $d(I, F_{k+1}) \leftarrow d(I, F_k) + \frac{1}{2} \ln \left| \frac{(b_k + \sqrt{D})}{(b_k - \sqrt{D})} \right|$;

   $k \leftarrow k+1$;

   If < F_k is ambiguous> then Return $R = 2d(I, F_k)$;

9. end

   $G \leftarrow F_k$;

10. For i = 1 until k do

Begin

11. $F_{1-i} \leftarrow (c_i, b_i, a_i)$;

$d(I, F_{1-i}) \leftarrow -d(I, F_i)$;

12. end

Comment Use the algorithm of Corollary 1.1.11 for composition and reduction, and Corollary 3.1.15 to compute $d(I, G_1)$.

13. <Compute $G_1 = \sigma^*(G^2)$ and $d(I, G_1)$ in conjunction>;

14. while $d(I, G_1) > 2d_o - \frac{1}{2} \ell n(1 + (1 + \sqrt{5})\sqrt{D})$ do

Begin

15. G_1 <the neighbour (a,b,c) by last part to G_1>;

$$d(I, G_1) \leftarrow d(I, G_1) - \tfrac{1}{2} \ell n \left| \frac{b + \sqrt{D}}{b - \sqrt{D}} \right|$$

16. end

17. <Sort all F_i's to obtain a list L>;

$j \leftarrow 1$;

18. while <$G_j \notin L$ > do

Comment Use "binary search" to search if $G_j \in L$

Begin

19. <Compute $G_{j+1} \leftarrow \sigma^*(G_j \circ G_1)$ and $d(I,G_{j+1})$ in conjunction>;

Comment Use algorithm 0.3.6 for reduction and Corollary 3.1.15 to compute $d(I,G_{j+1})$

$J \leftarrow J+1$;

20. end;

<Let m be such that $G_j = F_m$>;

$R \leftarrow d(I,G_j) - d(I,F_m)$;

Return R;

end. □

THEOREM 3.1.22

The algorithm 3.1.21 terminates and correctly yields the regulator.

Proof

It is sufficient to prove that $d(G_j,G_{j+1}) < 2d_o$ for all j's, because then one of the jumps will fall into the interval $[-d_o,d_o]$. The rest of the correctness of the algorithm follows from Corollary 3.1.15 and Theorem 3.1.18.

First we have

$$d(I,G_1) < 2d_o - \tfrac{1}{2} \ln(1 + (1 + \sqrt{5})\sqrt{D}) \qquad (3.17)$$

Now using definition 3.1.14

$$d(G_n,G_{n+1}) = d(I,G_{n+1}) - d(I,G_n)$$

Now since

$$d(I,G_{n+1}) = d(I,G_1) + d(I,G_n) + d(G_n \circ G_1, \sigma^*(G_n \circ G_1))$$

we have

$$d(G_n, G_{n+1}) = d(I, G_1) + d(G_n \circ G_1, \sigma^*(G_n \circ G_1)) \qquad (3.18)$$

Hence from Proposition 3.1.20 and (3.17) we have

$$d(G_n, G_{n+1}) < 2d_o. \qquad \square$$

THEOREM 3.1.23

The algorithm 3.1.21 yields the regulator in $O(D^{1/4} M(\log D) \log D)$ elementary operations.

Proof

Steps 1-6 require only $O(M(\log D))$ elementary operations. To find a neighbour of a form (step 8) requires only $O(M(\log D))$ elementary operations.

Now using Proposition 3.1.16(ii) we have that the number of iterations in the loop 8-9 (similarly loop 10-12) is at most

$$2 d_o / \ell n\, 2 + 1 = O(D^{1/4} \log D).$$

Hence loop 8-9 requires $O(D^{1/4} \log D\, M(\log D))$ elementary operations. Also the loop 10-12 requires $O(D^{1/4} \log^2 D)$ elementary operations.

To compute $\sigma^*(G^2)$ by Corollary 1.1.11 requires $O(M(\log D) \log D)$ elementary operations.

Now from Corollary 3.1.15 we have

$$d(I,\sigma^*(G^2)) = 2\, d(1,G) + d(G^2,\sigma^*(G^2))$$

Now from Proposition 1.1.16(i) and Proposition 3.1.17 we have

$$d(I,G) < d_o + \tfrac{1}{2}\,\ell n + D \qquad \text{and}$$

$$d(G^2,\sigma^*(G^2)) < \tfrac{1}{2}\,\ell n(1 + (1+\sqrt{5})\sqrt{D}).$$

Hence we have

$$d(I,\sigma^*(G^2)) < 2\, d_o + \tfrac{1}{2}\,\ell n\, 4D + \tfrac{1}{2}\,\ell n(1+\sqrt{5})\sqrt{D}) \quad (3.19)$$

From (3.19), (3.18) and Proposition 3.1.16(ii) finding a form G_1 (loop 14-16)

$$d(I,G_1) < 2\, d_o - \tfrac{1}{2}\,\ell n(1 + (1+\sqrt{5})\sqrt{D})$$

requires at most $1 + 2\,(\tfrac{1}{2}\,\ell n\, 4D + \ell n\,(1+(1+\sqrt{5})\sqrt{D}))/\ell n 2 = O(\log D)$ iterations of the loop 14-16. Since step 15 requires only $O(M(\log D))$ elementary operations, the loop 14-16 requires $O(\log D\, M(\log D))$ elementary operations.

Since the number of F_i's is $O(D^{1/4} \log D)$, sorting requires $O(D^{1/4} \log^2 D)$ elementary operations (see [2], p.87). As step 18 to search whether G_j belong to L using binary search requires only $O(\log D)$ elementary operations (see [2], p. 114)

Now will be shown that $d(G_i,G_{i+1}) > d_o$

One can see that

$$d(I,G_1) > 2d_o - \tfrac{1}{2}\,\ell n(1 + (1+\sqrt{5})\sqrt{D}) - \tfrac{1}{2}\,\ell n\, 4D > d_o$$

using Proposition 3.1.16(i).

Moreover from (3.17)

$$d(G_j, G_{j+1}) = d(I, G_1) + d(G_j G_1, \sigma^*(G_n G_1)).$$

Hence we have

$$d(G_j, G_{j+1}) > d_o$$

Thus a match $G_j = F_m$ will be found as most after

$$1 + R/d_o < 1 + \frac{6\sqrt{D}\,\ell nD}{d_o} = O(D^{1/4})$$

iterations of the loop 18-20.

Hence, since step 19 requires O(M(log D)log D)) elementary operations (similarly with step 13), the loop 18-20 requires at most $O(D^{1/4}\ M(\log D)\ \log D)$ elementary operations.

From the above analysis we conclude that the algorithm yields the regulator in $O(D^{1/4}\ M(\log D)\ \log D)$ elementary operations in worst-case. □

Since ambiguous forms are very easy to identify, it may be helpful to test the ambiguity of G_i at step 19 in algorithm 3.1.21. If G_i is ambiguous, then $R = 2\ d(I, G_i)$, but the probability of finding the midpoint of the principal cycle by chance in this fashion is so small for large D that this modification is unlikely to have practical interest.

An iterative version of Lenstra's algorithm for computation of the regulator given below may have practical application. The method is the following:

Begin

k ← 0;

while the regulator is not computed>do

Begin

<Compute 2^k baby steps>

Comment. We find 2^k neighbours (F_i's) of I by either side

<Compute 2^k giant steps>

Comment We compute 2^k forms G_i, where G_i as in algorithm 3.1.21

If <a match ($G_m = F_n$) is found> then Return <the regulator>;

k ← k + 1

end

One using Proposition 3.1.16(ii) can show that 2^k baby steps cover distance $d_b > 2^k \ln 2$ and 2^k giant steps cover distance $d_g > 2^k d_o > 2^{2k} \ln 2$ (since each giant step is greater than d_o, see 3.1.21-23). If we find for some k a match is found then $R > 2^{k-1} \ln 2 + 2^{2(k-1)} \ln 2 > 2^{2(k-1)} \ln 2$ and thus $2^k = O(\sqrt{R})$.

Hence this modified version always requires $O(\sqrt{R})$ steps to compute the regulator, whilst algorithm 1.3.21 requires R/2 steps when $R < (6\sqrt{D} \ln 4D)^{1/2}$. Both algorithms have the same worst-case complexity time.

Lenstra ([25], p. 22) asserts that, using the same technique as in the algorithm 3.1.21 one can decide in $O(D^{1/4+\varepsilon})$ elementary operations whether or not two indefinite forms F,G are equivalent. Such an algorithm is given below.

We note that to decide equivalence between two forms F,G, it is sufficient to decide whether or not the form $F \circ G^{-1}$ belongs to the principal cycle. The following algorithm decides whether or not a reduced form Q with determinant $D > 0$ (non-square) belongs to the principal cycle.

1. Let $I = (1,b_o,c_o)$ form of the principal cycle. Compute the forms

$$F_{-k},\ldots,F_{-2},F_{-1},\ I = F_o,F_1,F_2,\ldots,F_k$$

and their distances as in algorithm 3.1.21. If Q is one of the F_i's, then Q belongs to principal cycle. If F_j is ambiguous for some j, then $\{F_{-j+1},\ldots,I = F_o,\ldots,F_j\}$ are all the forms of the principal cycle, and if Q is not one of the F_i's, then Q does not belong to the principal cycle. Assume then that no F_i is ambiguous and Q is not an F_i.

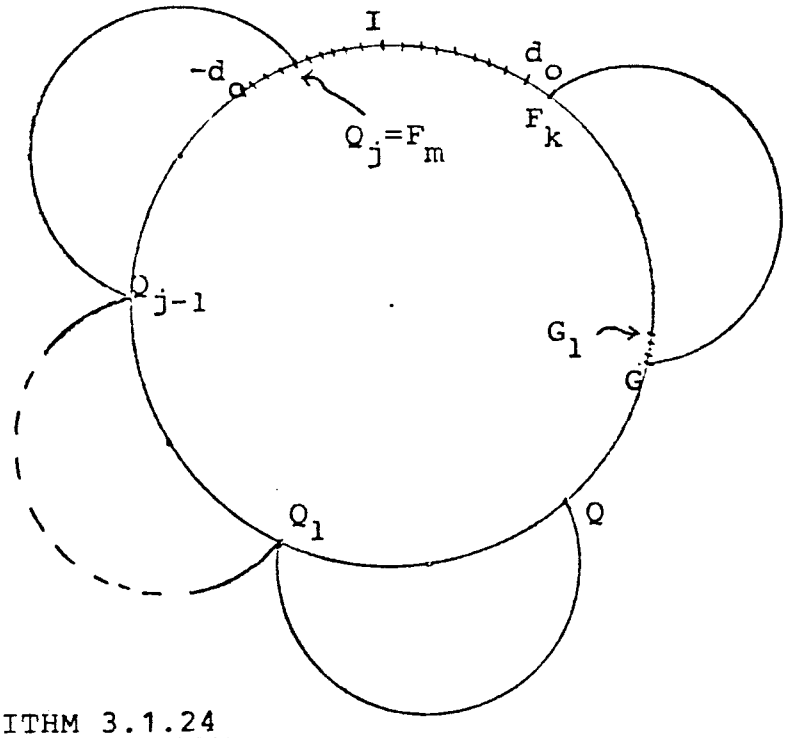
2. Compute $G = \sigma^*(F_k^2)$ and $d(I,G)$ in conjunction.

3. By searching the neighbours of G find (as in 3.1.21 (steps 14-16)) a form G_1 which satisfies

$$2d_o - \tfrac{1}{2}\,\ell n(1 + (1+\sqrt{5})\sqrt{D}) - \tfrac{1}{2}\,\ell n(4D) < d(I,G_1) < 2d_o - \tfrac{1}{2}\,\ell n(1 + (1+\sqrt{5})\sqrt{D})$$

4. Compute $Q_{j+1} = \sigma^*(Q_1 \circ G_1)$ for $j = 0,1,\ldots$ and $d(I,Q_j)$ until either for some $j, Q_j = F_m$ or $d(I,Q_j) > (6\sqrt{D}\ \ell n 4D)^{1/2}$. In the former case, Q belongs to the principal cycle, whilst in the latter case, it does not.

The steps of the algorithm when Q belongs to the principal cycle may be illustrated diagrammatically thus:



ALGORITHM 3.1.24

INPUT = A reduced form Q with determinant $D > 0$ (non-square)

OUTPUT = Whether or not Q belongs to the principal cycle of $C\ell(D)$.

Begin

$F_o \leftarrow (1, b_o, c_o)$ where $b_o = \lfloor \sqrt{D} \rfloor$ and $c_o = b_o^2 - D$;

$k \leftarrow 0$;

$d(I, F_o) \leftarrow 0$;

while $d(I, F_i) < d_o = (6\sqrt{D}\ \ell n\ 4D)^{1/2}$ do

Begin

$F_{k+1} \leftarrow \sigma(F_k)$;

$d(I, F_{k+1}) \leftarrow d(I, F_k) + \frac{1}{2}\ \ell n \left| \frac{b_k + \sqrt{D}}{b_k - \sqrt{D}} \right|$;

$k \leftarrow k+1$;

end

For $i = 2$ until $k+1$ do

Begin

$F_{-i+1} \leftarrow (c_i, b_i, a_i)$;

$d(I, F_{1-i}) \leftarrow -d(I, F_i)$;

end

If <$Q = F_i$ for some i> then Return "Q belongs to the principal cycle";

If <F_i is ambiguous for some i> then Return "Q does not belong to the principal cycle";

<Compute $G = \sigma^*(F_k^2)$ and $d(I,G)$ in conjunction>;

$G_1 \leftarrow G$;

while $d(I, G_1) > 2d_o - \frac{1}{2}\ \ell n(1 + (1 + \sqrt{5})\sqrt{D}$ do

Begin

$G_1 \leftarrow$ (the neighbour (a,b,c) by last part to G_1>;

$d(I, G_1) \leftarrow d(I, G_1) - \frac{1}{2}\ \ell n \left| \frac{b + \sqrt{D}}{b - \sqrt{D}} \right|$;

end

<Sort all F_i's to obtain a list L>;

$Q_o \leftarrow Q$;

$\bar{d}(I,Q_o) \leftarrow d_o$;

Comment For each i $\bar{d}(I,Q)$ is a lower bound for $d(I,Q)$

$j \leftarrow 1$;

while $<Q_j \notin L>$ do

Begin

if $\bar{d}(I,Q_j) > G\sqrt{D}\ \ell n\ 4D$ then

Return "Q does not belong to the principal class";

else

<Compute $Q_{j+1} = \sigma^*(Q_j.G_1)$ and $\bar{d}.(,Q_{j+1}) = d(I,G_1) + \bar{d}(I,Q_j) + d(Q_j \circ G_j, Q_{j+1})$ in conjunction>

$j \leftarrow j+1$;

end

Return "Q belongs to the principal class";

end. □

THEOREM 3.1.25

Algorithm 3.1.24 correctly decides whether a form Q belongs to the principal cycle in $O(D^{1/4} M(\log D) \log D)$ elementary operations.

Proof

Similar to that of Theorems 3.1.22 - 3.1.23. □

An application of the algorithm 3.1.24 will now be described. Using the following theorem one can decide whether the equation

$$T^2 - DU^2 = -1 \text{ (non-Pellian equation)}$$

is solvable or not.

THEOREM 3.1.26

The equation $T^2 - DU^2 = -1$ with $D > 0$ non-square has an integral solution if and only if $(1,0,-D) \approx (-1,0,D)$

Proof

($\Rightarrow$) Let (t,u) be a solution of $T^2 - DU^2 = -1$. Then

$$(1,0,-D) \to (-1,0,D) \text{ via } S = \begin{pmatrix} t & u \\ Du & -t \end{pmatrix}$$

and $\det(S) = -t^2 + Du^2 = 1$.

($\Leftarrow$) If $(1,0,-D) \approx (-1,0,D)$, then there exists a unimodular matrix $S = \begin{pmatrix} \kappa & \lambda \\ \mu & \nu \end{pmatrix}$ such that

$$\begin{pmatrix} \kappa & \mu \\ \lambda & \nu \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -D \end{pmatrix} \begin{pmatrix} \kappa & \lambda \\ \mu & \nu \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & D \end{pmatrix} \quad \text{which implies}$$

$$\begin{pmatrix} \kappa^2 - \mu^2 D & \kappa\lambda - \mu\nu D \\ \kappa\lambda - \mu\nu D & \lambda^2 - \nu^2 D \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & D \end{pmatrix}$$

Hence (κ,μ) is a solution of $T^2 - DU^2 = -1$. □

THEOREM 3.1.27

There exist an algorithm which decides whether or not the equation $T^2 - DU^2 = -1$ with $D > 0$ non-square has integral solutions in $O(D^{1/4} \log D\, M(\log D))$ elementary operations.

Proof

We use algorithm 3.1.24 to examine whether or not the form $\sigma^*(-1,0,D)$ belongs to the principal cycle. □

Also one can compute the forms of the principal cycle of $C\ell(D)$ using the algorithm 1.3.5 and if $\sigma^*(-1,0,D)$ belongs to the principal cycle, then using (3.9) can compute the matrices S_i and thus a matrix $S = \prod_i S_i$ such that

$$\sigma^*(1,0,-D) \to \sigma^*(-1,0,D) \text{ via } S$$

The reduction procedure (see 0.3.6) provides matrices S_1, S_2 such that

$$(1,0,-D) \to \sigma^*(1,0,-D) \text{ via } S_1$$

and $$(-1,0,D) \to \sigma^*(-1,0,D) \text{ via } S_2$$

Hence we have

$$(1,0,-D) \to (-1,0,D) \text{ via } S_1 S\, S_2^{-1} = \begin{pmatrix} \kappa & \lambda \\ \mu & \nu \end{pmatrix} .$$

Then (κ, μ) is a solution of the equations $T^2 - DU^2 = -1$ (see Theorem 3.1.26(⇐)).

Lagarias [21] gave an algorithm for deciding whether or not the non-Pellian equation has solution. His algorithm obtains a basis for the ambiguous forms using genus characters

and decides the equivalence of the forms $(1,0,-D)$, $(-1,0,D)$ by expressing them in terms of elements of the basis. His algorithm has worst-case complexity time $O(D^{1/4+\varepsilon})$. If complete factorization of D is provided and the G.R.H. is true, then his algorithm terminates in $O((\log D)^5 \log\log D \log\log\log D)$ elementary operations.

3.2. INFRASTRUCTURE OF $C\ell(D)$ WITH D NEGATIVE

The following theorem is due to Gauss ([16], A.172); it characterizes the number of reduced forms in a class of $C\ell(D)$ with $D < 0$.

THEOREM 3.21

Suppose $K \in C\ell(D)$ with $D < 0$. Then either K contains one reduced form or two opposite reduced ambiguous forms or two opposite reduced forms with equal the outer terms.

Proof

Let $F = (a,b,c)$ and $G = (A,B,C)$ be reduced forms in K. Then there exist a unimodular matrix S given by

$$S = \begin{pmatrix} \kappa & \lambda \\ \mu & \nu \end{pmatrix}$$

such that $S^T M_F S = M_G$.

Now with direct calculation of $S^T M_F = M_G S^{-1}$ we have:

$$A\nu - a\kappa = (B+b)\mu \tag{3.20}$$

$$c\mu + a\lambda = (B-b)\kappa \tag{3.21}$$

$$a\lambda + C\mu = (B-b)\nu \tag{3.22}$$

$$C\kappa - c\nu = (B+b)\lambda \tag{3.23}$$

Without loss of generality we may assume $|A| \geq |a|$. (3.24)

Also since $D < 0$ we have that $ac > 0$, $AC > 0$. Now using (3.20) and (3.22=

$$aA = aA(\kappa\nu - \lambda\mu) = A(A\nu^2 - C\mu^2 - 2B\mu\nu) = (Av - B)^2 + D\mu^2 > 0 \quad (3.25)$$

Hence a,c,A,G have the same sign.

Now using (3.24), (3.25) and the fact C is reduced we have

$$|2B\mu| = |A^2 + c\mu^2 - a| > \min(|A|,|C|)(\mu^2+\nu^2) - |a| \quad (3.26)$$

$$> |A|(\mu^2 + \nu^2 - 1) \quad (3.27)$$

$$> 2|B|(\mu^2 + \nu^2 - 1) \quad (3.28)$$

Hence $\mu^2 + \nu^2 - |\mu\nu| = (|\mu| - |\nu|)^2 + |\mu\nu| < 1$, which implies $(|\mu| - |\nu|)^2 + |\mu\nu| = 1$.

Now we have two cases.

Case $\mu = 0$

Then $\nu = 1$. From (3.26), (3.27) we have $|A| = |a|$.
If $\lambda = 0$, then from (3.23) $B = b$. Hence $F = G$.
If $\lambda \neq 0$, then from (3.23) we have $a\lambda = B - b$ and

$$|a| < |a\lambda| = |B-b| < 2\max\{|B|,|b|\} < |A| = |a| \quad (3.29)$$

Hence $|B-b| = 2\max\{|B|, |b|\}$, which implies $B = -b$.
And from (3.29) $2|B| = 2|b| = |a| = |A|$, which implies $F = \bar{G}$ and F,G ambiguous.

Case $\nu = 0$

Then $\mu = 1$. From (3.26), we have $|C| = |A|$. From (3.27), we have $|A| = |a|$. If $\kappa = 0$, then from (3.20) we have $B = -b$. Hence $F = \bar{G}$ with the outer terms equal.

If $k \neq 1$, then using (3.20)

$$|B + b| > |a| = |A| > 2 \max \{|B|, |b|\}$$

which implies

$$|B| = |b|.$$

Hence either $F = G$ or $F = \bar{G}$ with the outer terms equal. □

From the previous theorem one can see that the inner structure of $C\ell(D)$ is much simpler in the case of negative determinant than the positive one. Hence it is much easier to decide equivalence between two definite forms.

THEOREM 3.22

To decide whether or not two definite reduced forms F,G are equivalent requires $O(\log ||Q|| M(\log ||Q||))$ elementary operation, where $||Q|| = \max \{||F||, ||G||\}$.

Proof

First we check if F,Q have the same determinant. It requires $O(M (\log ||Q||))$ elementary operations. If they have not the same determinant, then they are not equivalent.

Second we reduce F,G to F', G' respectively using the algorithm 0.32. It requires $O(\log ||Q|| M(\log ||Q||))$ elementary operations. If F' and G' are either identical or opposite and ambiguous or opposite with equal outer terms, then $F \approx G$, else $F \not\approx G$.

Hence we can decide equivalence in $O(\log ||Q|| (M(\log ||Q||)$ elementary operations. □

3.3. DEGENERATE FORMS

Although the degenerate forms do not form a group, since, composition is not defined for quadratic determinant, we describe and analyse an algorithm to decide whether or not two degenerate forms are equivalent. The following theorem is due to Gauss ([16], A.207).

THEOREM 3.3.1

Two reduced degenerate forms are equivalent if and only if they are identical.

Proof

Let $F_1 = (a_1,h,0)$, $F_2 = (a_2,h,0)$ be equivalent reduced forms with the same determinant $D = h^2$ and

$$F_1 \to F_2 \quad \text{via} \quad S = \begin{pmatrix} \kappa & \lambda \\ \mu & \nu \end{pmatrix} .$$

Then we have

$$a_2 = a_1\kappa^2 + 2\kappa\mu h \tag{3.30}$$

$$h = a_1\kappa\lambda + h(\kappa\nu + \lambda\mu) \tag{3.31}$$

$$0 = a_1\lambda^2 + 2h\lambda\nu \tag{3.32}$$

$$\kappa\nu - \mu\lambda = 1 \tag{3.33}$$

From (3.31) and (3.32) we have

$$-h\lambda(\kappa\nu-\mu\lambda) = h\lambda$$

and from (3.33)

$$-h\lambda = h\lambda \ .$$

Hence $\lambda = 0$ (The case $h = 0$ is trivial)

Now from (3.33) we have $kv = 1$ which implies $|k| = 1$. Hence from (3.30) we have

$$a_2 = a_1 + 2\kappa\mu h \tag{3.34}$$

Now since F_1, F_2 are reduced we have

$$\begin{aligned} &0 < a_1 < 2h-1 \\ \text{and} \quad &0 < a_2 < 2h-1 \end{aligned} \tag{3.35}$$

Hence from (3.34) we have

$$0 < a_1 + 2\kappa\mu h < 2h-1$$

and since (3.35) we have $\mu = 0$.

Hence from (3.30) we have $a_1 = a_2$ and thus $F_1 = F_2$. □

THEOREM 3.3.2

There exists an algorithm to decide equivalence between two degenerate forms F_1, F_2 in $O(M(\log \|F\|)\log\log \|F\|)$ elementary operations, where $\|F\| = \max \{\|F_1\|, \|F_2\|\}$.

Proof

First we check if F_1, F_2 have the same determinant. This can be done in $O(M(\log \|F\|))$ elementary operations. If F_1, F_2 have not the same determinant they are not equivalent.

If F_1, F_2 have the same determinant, then we reduced F_1, F_2 to F_1', F_2' respectively using the algorithm 0.3.9. This can be done in $O(M(\log \|F\|) \log \log \|F\|)$ elementary operations. Now if $F_1' = F_2'$, then F_1 and F_2 are equivalent. □

4.1. CLASS NUMBER OF IMAGINARY QUADRATIC FIELDS

A. GAUSS ALGORITHM

DEFINITION 4.1.1.

The number of classes of properly primitive positive definite forms of fixed determinent $D < 0$ is called the *class number* of $Cl(D)$ and it is denoted by $h(D)$.

Gauss ([16], A.173) gave two algorithms for finding the class number of $Cl(D)$ for $D < 0$. Both are based on counting a set of representatives for $Cl(D)$ and both run in time exponential in $\log |D|$.

METHOD 4.1.2.

Gauss's method makes use of the inequalities

$$0 < A < 2\sqrt{|D|/3}$$

$$2|B| < A$$

which every positive definite reduced form (A,B,C) satisfies and the fact that $D = B^2 - AC$, which implies

$$B^2 \equiv D \bmod A.$$

The method is the following:

1. For $0 < A < 2\sqrt{|D|/3}$ find all pairs (A,B) such that $X = B$ satisfies

$$X^2 \equiv D \bmod A \quad \text{and} \quad |x| < A/2 \tag{4.1}$$

2. For each such pair (A,B), compute a triple (A,B,C) where $C = (B^2 - D)/A$. Then we have a set of forms (A,B,C) with determinant D.
3. Reject the forms which are not properly primitive
4. If A divides 2B or A = C, then reject (A,-B,C) (since $(A,B,C) \approx (A,-B,C)$ by Theorem 3.2.1).
5. The number of remaining forms is the class number

It is known that the number of solutions of (4.1) is $D(2^{t_A})$, where t_A is the number of distinct prime divisors of A. (see Apostol [4], p. 120-122). Now since

$$t_A = O(\log A / \log \log A)$$

the total number of solutions of (4.1) is

$$\sum_{A=0}^{k} D(2^{t_A}) = \sum_{A=1}^{k} O(A) = O(|D|) \text{ with } k = 2\sqrt{|D|/3}\,.$$

One can see that the method then requires $O(|D|^{1+\varepsilon})$ elementary operations. More detailed analysis yields the bound $O(|D|M(\log|D|) \log \log |D|)$.

METHOD 4.1.3.

Since every positive definite form (A,B,C) satisfies

$$0 < A < C \quad \text{and} \quad |B| < \sqrt{|D|/3}$$

we have the following method for computation of the class number.

1. Construct a set S of forms with determinant D comprising forms of shape $(A, \pm B, C)$, where $0 < B < \sqrt{|D|/3}$ and (A,C) satisfies $AC = B^2 - D$ and $0 < A < C$.
2. Modify S by rejecting forms which are not reduced.
3. Modify S by rejecting forms which are not properly primitive.
4. If two forms in S are equivalent, reject one of them (as in 4.1.2,(4)).
5. The number of remaining forms is the class number.

It can be shown that step 1 requires $O(|D|M(\log |D|))$ elementary opations, which is the bottleneck of the algorithm.

The counting methods for computation of the class-number are inefficient when one computes h(D) for a particular D, but are relatively efficient when used to compute class numbers of determinant lying in some range. One can count triples (a,b,c) with $|2b| < a < c$, $a > 0$ and sort them by determinant. This method was used by Buell in [9].

B. DIRICHLET'S FORMULAE

Dirichlet (see Mathews [26], p. 235) gave a formula for h(D) with $D < 0$, as an expression of Jacobi symbols:

$$h(D) = \begin{cases} \dfrac{2\sqrt{|D|}}{\pi} \displaystyle\sum_{k=0}^{\infty} \left(\frac{D}{2k+1}\right) & \text{if } D < -1 \\ \dfrac{4}{\pi} \displaystyle\sum_{k=0}^{\infty} \left(\frac{-1}{2k+1}\right) = 1 & \text{if } D = -1 \end{cases} \tag{4.1}$$

Now using the facts

(i) $\left(\frac{D}{2k+1}\right) = \left(\frac{4D}{2k+1}\right)$, since $\left(\frac{4}{2k+1}\right) = \left(\frac{2}{2k+1}\right)^2 = 1$ and

(ii) $\left(\frac{4D}{n}\right) = 0$ for n even (Kronecker's symbol)

we have

$$\sum_{k=0}^{\infty} \frac{1}{2k+1} \left(\frac{D}{2k+1}\right) = \sum_{n=1}^{\infty} \left(\frac{4D}{n}\right) \frac{1}{n} = L(1, \chi_{4D}).$$

Hence using Theorem 2.2.3 we have

$$h(D) = \frac{2\sqrt{|D|}}{\pi} L(1, \chi_{4D}) = \frac{2\sqrt{|D|}}{\pi} \prod_{\substack{p=3 \\ \text{prime}}}^{\infty} \frac{p}{p - \left(\frac{4D}{p}\right)} \tag{4.2}$$

Moreover observing that $\left(\frac{1}{p}\right) = 1$ we have

$$h(D) = \frac{2\sqrt{|D|}}{\pi} \prod_{\substack{p=3 \\ \text{prime}}}^{\infty} \frac{p}{p - \left(\frac{D}{p}\right)} \tag{4.3}$$

or

$$h(D) = \frac{\sqrt{|D|}}{\pi} L(1, \chi_D) \left(2 - \left(\frac{D}{2}\right)\right) \text{ for } D \not\equiv 3 \pmod 4 \tag{4.4}$$

(The exception is due to the fact that Kronecker's symbol $\left(\frac{D}{p}\right)$ is not defined for $D \equiv 3 \pmod 4$)

Now if $D = D_o S^2 < 0$, then, $h(D)$ and $h(D_o)$ are related by the following formula: (Mathews [26], p. 166)

$$h(D) = w\, h(D_o) S \prod_{p|S} \left(\frac{p - \left(\frac{D_o}{p}\right)}{p}\right) \text{ for p odd prime} \tag{4.5}$$

where $w = 1$ except when $D_o = -1$ then $w = 1/2$.

Moreover for square $D < -1$ there is a formula (also due to Dirichlet) expressing $h(D)$ as a finite sum of Kronecker

symbols (see Venkov [40], p. 224):

$$h(D) = \begin{cases} \sum_{k=0}^{\lfloor |D|/2-1 \rfloor} \left(\frac{D}{2k+1}\right) & \text{if } D \equiv 1,2 \bmod 4 \\ \sum_{k=1}^{\lfloor |D|/2 \rfloor} \left(\frac{K}{-D}\right) & \text{if } D \equiv 3 \bmod 4 \end{cases} \tag{4.6}$$

Now one can compute the class number $h(D)$ for $D < 0$ using the formulae (4.5) and (4.6) in the following way:

THEOREM 4.1.4

The computation of $h(D)$, for $D < -1$, via the formulae (4.5) and (4.6) requires $O(|D| \log|D| M(\log|D|))$ elementary operations.

Proof

1. Factor D. Let $D = -D_o S^2$ and $D_o = 2^{b_o} p_1^{b_1} \ldots p_r^{b_r}$ with $b_i \in \{0,1\}$.

2. If $D_o \equiv 1,2 \bmod 4$, then compute

$$\left(\frac{D_o}{2k+1}\right) \text{ for } 0 < K < \lfloor |D_o|/2 - 1 \rfloor$$

in the following way: (see definition 2.2.1 and properties)

(i) $$\left(\frac{2k+1}{p_i}\right) \equiv (2k+1)^{(p_i-1)/2} \bmod p_i \quad \text{for } 1 < i < r$$

using Euler's criterion.

(ii) $\left(\frac{p_i}{2k+1}\right) = \left(\frac{2k+1}{p_i}\right) (-1)^{k(p_i-1)/2}$ for $1 \leq i \leq r$

(iii) $\left(\frac{-1}{2k+1}\right) = (-1)^k$ and $\left(\frac{2}{2k+1}\right) = (-1)^{k(k+1)/2}$

and hence

(iv) $\left(\frac{D_o}{2k+1}\right) = \left(\frac{-1}{2k+1}\right) \left(\frac{2}{2k+1}\right)^{b_o} \left(\frac{p_i}{2k+1}\right)^{b_1} \ldots \left(\frac{p_r}{2k+1}\right)^{b_r}$

if $D \equiv 3 \bmod 4$ compute $\left(\frac{k}{-D}\right)$ similarly.

3. Compute $h(D_o)$ using (4.6) (If $D_o = -1$, then $h(D_o) = 1$)

4. Compute $h(D)$ using (4.5) and $\left(\frac{D_o}{p}\right) \equiv D_o^{(p-1)/2} \bmod p$.

Now one can factor D in step 1 using Pollard's [30] method in $O(|D|^{1/4+\varepsilon})$ elementary operations.

Steps (i) - (iii) require $O(\log|D|M(\log|D|))$ elementary operations (see analysis of steps 11-12 of 2.1.11) and step (iv) requires only $O(\log|D|)$, since it is multiplication by ±1's and ±1's. Hence step 2 requires $O(|D|\log|D|M(\log|D|))$ elementary operations. Now step 2 requires $O(|D|)$ elementary operations for additions.

Step 4 requires $O(\log|D|/\log\log|D|)$ computations of the symbol $\left(\frac{D_o}{p}\right)$ and thus costs $O(\log^2|D|M(\log|D|))$ elementary operations. From the above analysis the theorem follows. □

Given h(D) the computation of Dirichlet's function $L(1,\chi_D)$ for $D \not\equiv 3 \pmod 4$ can be done using (4.4). Hence $L(1,\chi_D)$ can be computed in $O(|D|\log|D|M(\log|D|))$ elementary operations using the method for computation of h(D) given by Theorem 4.1.4.

C. SHANKS' ALGORITHM

Before describing Shanks' algorithm, it is necessary to state some basic background results.

Shanks' algorithm for computing the class number makes use of the group structure on $C\ell(D)$. An essential step is the consideration of a set of generations for $C\ell(D)$. The following theorems are used to give bounds on the amount of computation required to find such a generating set, with and without assumption of the GRH. The proofs are direct applications of results by Lagarias and Odlyzko [23], Lagarias, Montgomery and Odlyzko [22].

THEOREM 4.1.5

If the G.R.H is true, then there exists an effectively computable positive constant c_1 which does not depend on D such that the set of forms of shape

$$(p, B_p, (B_p^2 - D)/p) \tag{4.7}$$

where $p < c_1 \log^2|D|$ is prime and $B_p^2 \equiv D \pmod p$ generates $C\ell(D)$.

Proof

From Lagarias and Odlyzko [23], it follows that there exists a positive constant c_1, effectively computable, and independent of D, such that the class group of ideals of $\mathcal{O}_{2f}(d)$ where $D = f^2 d$, d square free, is generated by the set of non-principal ideals whose norm is a rational prime $p < c_1 \log^2 |D|$. Such ideals correspond to forms of shape $(p, B_p, (B_p^2 - D)/p)$ where $B_p^2 \equiv D \pmod{p}$ under the isomorphism of Theorem 1.2.4 and Theorem 4.1.5 follows immediately. (Also see Schoof [33], p. 179). □

THEOREM 4.1.6

There exist two positive constants c_1, c_2 which do not depend on D such that the set of forms of shape

$$(p, B_p, (B_p^2 - D)/p),$$

where $p < c_2 |D|^{c_3}$ is prime and $B^2 \equiv D \pmod{p}$, generates $C\ell(D)$.

Proof

It is an application of [22] and Theorem 1.2.4. □

The following theorem yields a relation between h(D) and D.

THEOREM 4.1.7

Suppose that p is an odd prime and that h(D) is the class number of $C\ell(D)$.

(i) If $p^{2k} \| D$ for some integer $k > 1$, then

$$p^{k-1}(p - (\frac{D_p}{p})) \mid h(D), \text{ where } D_p = D/p^{2k}.$$

(ii) If $p^{2k+1} \| D$ for some integer $k > 1$, then $p^k \mid h(D)$.

Proof

Let $D = p_1^{2a_1+1} \ldots p_r^{2ar+1} q_1^{2b_1} \ldots q_n^{2b_n} 2^{m+t}$ with $t = 1$ or 0, $D_o = p_1 \ldots p_r 2^t$ and $S^2 = \frac{D}{D_o}$ where p_i, q_i are distinct odd primes.

From (4.5) we have

$$h(D) = w\, h(D_o)\, p_1^{a_1} \ldots p_t^{a_r}\, q_1^{b_1} \ldots q_2^{b_n} 2^m \prod_{\substack{p \mid s \\ p \text{ odd}}} \frac{p - (\frac{D_o}{p})}{p}.$$

One can see that $(\frac{D_o}{p_i}) = 0$, hence

$$\prod_{\substack{p \mid s \\ p \text{ odd}}} \frac{p - (\frac{D_o}{p})}{p} = \prod_{i=1}^{n} (q_i - (\frac{D_o}{q_i}))/q_i$$

And finally

$$h(D) = wh(D_o)\, p_1^{a_1} \ldots p_r^{a_r}\, q_1^{b_1-1} \ldots q_n^{b_n-1} \prod_{i=1}^{n} (a_i - (\frac{D_o}{q_i})) \qquad (4.11)$$

and since $(\frac{D_o}{q}) = (\frac{D_o}{q})$ the theorem follows. □

Now some bounds on the class number of $Cl(D)$ are given.

PROPOSITION 4.1.9

Suppose that $h(D)$ is the class number of $C\ell(D)$ with $D < 0$. Then

$$h(D) < 2\sqrt{|D|} \ \ell n \ |4D|$$

Proof

By Theorem 2.2.4 we have

$$L(1,\chi_{4D}) < 3 \ \ell n \ |4D|$$

and from (4.2) follows that

$$h(D) < 2\sqrt{D} \ \ell n \ |4D| \ . \quad \square$$

THEOREM 4.1.10

Suppose that $L = (1,X_D)$ and $L' = \prod_{\substack{p<m \\ p \text{ prime}}} (p/(p - (\frac{D}{p})))$.

If the Generalized Riemann Hypothesis is true, then

$$\frac{L'}{L} = 1 + O(m^{-1/2}(\log |D| + \log m))$$

Proof

This is a result due to Odlyzko (see Monier [28], p. 3.8, Lenstra [25], p. 11). Also the constant of the O-symbol is $c_4 = 20$ (see [28], p. 3.11). $\square$

COROLLARY 4.1.11

Suppose that

$$\tilde{h} = \frac{2\sqrt{|D|}}{\pi} \prod_{p<m} \left(\frac{p}{p - (\frac{D}{p})}\right) \text{ and } h = \frac{2\sqrt{|D|}}{\pi} L(1,x_o).$$

If the G.R.H. is true, then

$$\varepsilon(m) = |\tilde{h} - h| = O(\sqrt{|D|} \log|D| m^{-1/2}(\log|D| + \log m))$$

Proof

We have $\varepsilon(m) = |\tilde{h} - h| = \frac{2\sqrt{|D|}}{\pi} |L - L'|$, where L, L' as in Theorem 4.1.10. By Theorems 4.1.10 and 2.2.4 we have

$$|L' - L| = L.O(m^{-1/2}(\log |D| + \log m))$$

Hence $\varepsilon(m) = O(\sqrt{|D|} \log |D| \, m^{-1/2}(\log |D| + \log m))$. Also the constant of the O-symbol is $c_5 = 14$ (it follows from c_4 and 2.2.4). □

DEFINITION 4.1.12

Suppose that G is a finite group. Then the smallest integer e such that

$$a^e = 1 \quad \forall \quad a \in G$$

is called the *exponent* of the group G.

Shanks' algorithm for computation of the class number h(D) of Cℓ(D) will now be described. We shall describe the algorithm in two parts. First we shall give an algorithm

computing the exponent of $Cl(D)$ and afterwards an algorithm computing the structure and the class number of $Cl(D)$.

Shanks' algorithm for computation of the exponent of $Cl(D)$, constructs a set of generators of $Cl(D)$ (using Theorem 4.1.5 or 4.1.6), computes the order of each generator and determines the exponent e of the form class group as the L.C.M. of these orders. The algorithm uses a special technique (Shanks' "baby-giant step" strategy) to find the order of each generator and computes the exponent e of $Cl(D)$ in $O(D^{1/5+\varepsilon})$ elementary operations under the assumption of the truth of G.R.H.

An outline of the algorithm is now given.

A. Compute an $\tilde{h}$ to approximate the class number $h(D)$ using the formula

$$h = \frac{2\sqrt{|D|}}{\pi} \prod_{\substack{p<m \\ \text{odd prime}}} \frac{p}{p - (\frac{D}{p})}$$

where $m = m(D)$ is a positive integer chosen so as to achieve optimal efficiency. If a bound of the approximation is $\varepsilon(m)$ (it can be computed using the formula of Corollary 4.1.11), then

$$\tilde{h} - \varepsilon(m) < h(D) < \tilde{h} + \varepsilon(m)$$

B. Initialise b to the largest factor of $h(D)$ (e.g. using Theorem 4.1.7).

C. Compute a set of generator $\{F_p\}$ of $C\ell(D)$ (as justified by Theorem 4.1.5), where $F_p = (p, B_p, *)$ for primes $p < c_1(\log D)^2$ and B_p is a solution of the equation

$$X^2 \equiv D \pmod{p} \quad 0 < X < p.$$

D. For each generator carry out D1 - D4.

D1. Compute the closest integer h^+ to $\tilde{h}$ such that $h^+ \equiv 0 \pmod{b}$ Observe that

$$|h^+ - h(D)| < \varepsilon(m) + b = \alpha$$

D2. If $F_p^{h^+} \not\approx I$, then by searching in the interval $(h^+-\alpha, h^++\alpha)$ find an integer $h' \equiv 0 \pmod{b}$ such that $F_p^{h'} \approx I$ (In this interval there exists at least one such integer, viz $h(D)$). Let h^+ become equal to h'.

D3. Using the relation $F_p^{h^+} \approx I$ compute the order e_p of the form F_p.

D4. Let b to become equal to $\mathrm{LCM}(b, e_p)$ (since $e_p | h(D)$).

E. The exponent e of C (D) is determined by

$$e = \mathrm{LCM}\ (\{e_p\}).$$

Some technical details of Shanks' algorithm will now be given. The following techniques are used to speed up the algorithm.

1. One can improve the approximation $\tilde{h}$ and simultaneously find a factor of h(D) (see step B above) using Theorem 4.1.7. One can compute the sets

$$P_1 = \{p < m : p^{2a_p} \| D \text{ for some integer } a_p > 1 \text{ and } p \text{ odd prime}\}$$

$$P_2 = \{p < m : p^{2a_p+1} \| D \text{ for some integer } a_p > 0 \text{ and } p \text{ odd prime}\}$$

and let

$$b = \prod_{p \in P_1} p^{\alpha_p - 1} \left(p - \left(\frac{D_p}{p}\right)\right) \prod_{p \in P_2} p^{\alpha_p}, \text{ where } D_p = D/p^{2a_p} \text{ and}$$

$\left(\frac{D_p}{p}\right)$ is the Legendre symbol.

Let $2^{2m+t} \| D$ with $t = 0,1$. If D is a square, then let b become equal to $2^{m-1}b$ otherwise let b become equal to $2^m b$. From Theorem 4.1.7 and (4.11) we have that $b \mid h(D)$.

2. One can compute $F_p^{h^+}$ (step D2) the following way. First express h^+ in binary

$$h^+ = \sum_{i=0}^{n} a_i 2^i, \text{ where } a_i \in \{0,1\}$$

and compute $F_p^{h^+}$ via the recursive relation

$$G_{i-1} = G_i^2 F_p^{a_{i-1}} \text{ for } i = n+1 \text{ until } i = 1$$

with $G_{n+1} = (1,0,-D)$. Then $F_p^{h^+} = G_o$.

3. The most difficult part of the algorithm is when $F^{h^+} \not\approx I$ (in step D2) and it is necessary to find an $h' \equiv 0 \pmod{b}$ such that $f^{h'} \approx I$ by searching in the range

$$h^+ - \alpha < h' < h^+ + \alpha. \qquad (4.12)$$

(There is at least one such integer in the interval (4.12), viz h(D)). A direct search into the interval (4.12) would require $O(\varepsilon(m) + b)$ compositions, but this can be reduced to $O(\sqrt{\varepsilon(m)})$ by applying the baby-giant step strategy.

Suppose that $h' = h^+ + \varepsilon$. Then from (4.12) we have

$$|\varepsilon| = |h'-h^+| < |\tilde{h}-h^+| + |h'-\tilde{h}| < \varepsilon(m) + b$$

Now let $s = \lceil \sqrt{\varepsilon(m)/b + 1} \rceil$. Since $b|h^+$ and we want $b|h'$ we have that $b|\varepsilon$. Hence ε can be written

$$\varepsilon = 2rsb + tb \text{ for some integers } |r| < s/2 \text{ and } |t| < s$$

Let $g = 2sb$.

First compute

$$(F_p^b)^t \text{ for } 0 < t < s \quad \text{(baby steps)}$$

and

$$(F_p^b)^{-t} = \overline{(F_p^b)^t} \quad \text{for } 0 < t < s$$

since the inverse of a form is its opposite (see 1.2.2).

Now compute

$$F^{h^+ + rg} \text{ for } -\frac{s}{2} < r < \frac{s}{2} \quad \text{(giant steps)}$$

Now we have two cases.

(i) If $|\varepsilon| < \lceil \sqrt{\varepsilon(m) + b} \rceil$, then $\varepsilon = tb$ for some $|t| < s$ and thus a match will be found $F_p^{h^+} \approx F_p^{-tb}$. Hence $h' = h^+ + tb$ and $F_p^{h'} \approx I$.

(ii) If $|\varepsilon| > \lceil \sqrt{\varepsilon(m) + b} \rceil$, then $\varepsilon = rg + tb$ for some $|r| < \frac{s}{2}$ and $|t| < s$ and thus a match will be found

$$F_p^{h^+ + rg} \approx F_p^{-tb}.$$

Hence $h' = h^+ + rg + tb$ and $F_p^{h'} \approx I$.

Let h^+ to become equal to h'.

4. One can compute the order of a form F_p (step D3) in the following way. We have that $F_p^{h^+} \approx I$. Then we factor h^+ using Pollard's[30] method. Let $h^+ = \prod_i q_i^{a_i}$ and $t_i = h^+/p_i^{a_i}$. If for every prime divisor a_i of $\tilde{h}$ one finds the smallest integer s_i such that:

$$(F_p^{t_i})^{q_i^{s_i}} \approx I$$

then the order e_p of the form F_p is

$$e_p = \prod_i q_i^{s_i}$$

Some remarks about Shanks' algorithm.

- Whenever two forms are composed, the composition is followed by reduction of the composed form (use algorithm of Corollary 1.1.11).
- To decide equivalence of two forms, the algorithm of Theorem 3.2.2 is used.

- To solve the equation $X^2 \equiv D$ the method of Tonelli-Shanks (see 0.2.5 - 0.2.6) is used.

ALGORITHM 4.1.13

INPUT : The determinant $D < 0$, an integer m bound for the approximation $\tilde{h}$ of $h(D)$.

OUTPUT : The exponent of $Cl(D)$ and a generating set G_p for every p-Sylow subgroup of $Cl(D)$ together with the orders of the generators.

Begin

1. $\tilde{h} \leftarrow \frac{2\sqrt{|D|}}{\pi} \prod_{\substack{p<m \\ p \text{ odd prime}}} \frac{p}{p - (\frac{D}{p})}$

$\varepsilon(m) \leftarrow C_5 \sqrt{|D|} \log |D| m^{-1/2} (\log |D| + \log m)$

Comment For the constant C_5 see Corollary 4.1.11

$b \leftarrow 1$: $G_p \leftarrow \emptyset$;

Comment We shall make use of Theorem 4.1.7 (see 1 above 4.1.13)

2. For <all add primes $p < m$> do

Begin

If <$p^{2a_p} || D$ for some a_p> > then

Begin

$D_p \leftarrow D/p^{2a_p}$;

$b \leftarrow b p^{a_p - 1} (p - (\frac{D_p}{p}))$;

end

If $\langle p^{2a_p+1} \| D$ for some $a_p > 0 \rangle$ then

$b \leftarrow b.p^{a_p}$;

3. end

If $\langle 2^{2m+t} \| D$ with $t = 0,1 \rangle$ then

If $\langle |D|$ is a square$\rangle$ then

$b \leftarrow 2^{m-1}b$;

else

$b \leftarrow 2^{m}b$;

$p' \leftarrow 2$;

while $p < C_1 \log^2 |D|$ do

4. Begin

<Compute h^+ the closest integer to $\tilde{h}$ such that

$h^+ \equiv 0 \pmod{b}$>;

5. Comment Construct a form $F_p = (p, B_p, C_p)$

<Find the smallest prime p such that $(\frac{D}{p}) = 1$ and $p > p'$>;

$p' \leftarrow p$;

<Solve the equation $X^2 \equiv D \pmod{p}$ $0 < x < p$;

$B_p \leftarrow x$;

$C_p \leftarrow (B_p^2 - D)/p$;

$F_p \leftarrow (p, B_p, C_p)$;

6. <Reduce F_p>;

Comment $F_p^{h^+}$ will be computed by expressing h^+ in binary

7. <Compute a_i's $\in \{0,1\}$ such that $h^+ = \sum_{i=0}^{n} a_i 2^i$>;

$G_{n+1} \leftarrow (1,0,-D)$:

8. For i = n+1 until 1 do

Begin

$G_{i-1} \leftarrow (G_i)^2 \; F_p^{a_{i-1}}$;

9. end

$F_p^{h^+} \leftarrow G_o$;

$s \leftarrow \lceil \sqrt{\varepsilon(m)/b+1} \rceil$

If $\langle F_p^{h^+} \neq (1,0,-D) \rangle$ do

Begin

10. <Compute $(F_p^b)^t$ for all $|t| < s$>;

Comment This can be done by computing $(F_p^b)^t$ for $0 < t < s$ and finding $(F_p^b)^{-t}$ for $0 < t < s$ using the relation $F_p^{-bt} = \overline{(F_p^{bt})}$

<Sort the list $L = \{F_p^{bt} : |t| < s\}$>;

$g \leftarrow 2bs$;

$r \leftarrow 0$;

$G \leftarrow F_p^{h^+}$;

11. While <G or $\bar{G}$ is not in the list L> do

Comment To find whether or not G or $\bar{G}$ belongs to the list L use binary search (see [2])

Begin

$G \leftarrow G.F^g$

$r \leftarrow r+1$;

12. end

<Let $G = F_p^{-bt}$ for some $|t| < s$>;

$h^+ \leftarrow h^+ + rg + bt$;

end

Comment The order e_p of F_p is computed

13. <Factor h^+ and let $h^+ = q_1^{a_1} \ldots q_r^{a_r}$>;

$e_p \leftarrow 1$;

14. For i = 1 until r do

Begin

$t_i \leftarrow h^+/q_i^{a_i}$;

$G \leftarrow F_p^{t_i}$: $s_i \leftarrow 0$;

while <$G \neq I$> do

15. Begin

$G \leftarrow G^{q_i}$;

$s_i \leftarrow s_i + 1$;

16. end

$e_p \leftarrow e_p . q_i^{s_i}$;

<Insert in the list G_q the element $(F_p^{t_i}, q_i^{s_i})$>;

17. end

$b \leftarrow$ L.C.M.(b, e_p);

$e \leftarrow$ L.C.M.(e, e_p);

18. end.

Return e, <G_q for every distinct prime q which divide e>;

end. □

THEOREM 4.1.14

If G.R.H. is true, then algorithm 4.1.13 correctly computes the exponent e of $Cl(D)$.

Proof

It follows from 4.1.5 and the description of the algorithm (above 4.1.13). □

THEOREM 4.1.15

If G.R.H. is true and m is suitably chosen, then algorithm 4.1.13 terminates in $O(|D|^{1/5}(\log|D|)^4 M(\log|D|))$ elementary operations.

Proof

The parts of the algorithm which involve m are analysed to determine the optimal choice of m.

Since the number of primes less than m is $O(m/\log m)$ by prime number theorem, the computation of step 1 requires $O(m/\log m)$ computations of Legendre symbols and $O(m/\log m)$ multiplications. To find all primes less than m requires m applications of a primality test. Using Miller's [27] test, which tests a integer k for primality in $O(\log k)^4)$ elementary operations on the assumption of the G.R.H, this can be done in $O(\sum_{k=1}^{m} (\log k)^4)$ elementary operations. Also the computation of the Legendre symbol $(\frac{D}{p})$ requires $O(M(\log|D|) + \log p\ M(\log p))$ elementary operations (see Theorem 2.1.11, analysis of steps 11-12). Hence step 1 requires

$$\max\{O(m(\log m)/\log m),\ O(m(M(\log|D|)/\log m + \sum_{p<m} \log p\ M(\log p)),\ O(\sum_{k=1}^{m} (\log k)^4)\}$$

elementary operations. This can be simplified to

$$\max\{O(m(M(\log|D|)/\log m + m\,M(\log m)), O(m(\log m)^4)\} =$$

$$= O(m.\max\{M(\log|D|)/\log m, (\log m)^4\})$$

Now loop 2-3 requires only $(m \log|D|/\log m)$ elementary operations for divisions.

Step 10 requires s compositions and by Corollary 1.1.11 requires $O(s \log|D| M(\log|D|))$ elementary operations. To sort the list L takes $O(s \log s)$ elementary operations (see [2], p. 87). To search (step 11) if G or $\overline{G}$ belongs to the list L costs $O(\log s)$ elementary operations, using binary search (see [2], p. 114). Since a G or $\overline{G}$ will be found in the list L for some r with $|r| < |s|$, the loop 11-12 requires at most s compositions and thus $O(s \log |D|\ M(\log|D|))$ elementary operations. Since $s \leq \sqrt{\varepsilon(m)} = O(|D|^{1/4} m^{-1/4} (\log|D| (\log|D| + \log m))^{1/2})$ (by Corollary 4.1.11) we have that steps 10-12 require $O(|D|^{1/4} \log^2 |D| M(\log D) m^{-1/4})$ elementary operations (using $m < |D|$).

Since the parts of the algorithm analysed above are the only parts depending on m and the loop 4-18 is iterated $O(\log^2|D|)$ times, we conclude that the optimal choice of m is $m = |D|^{1/5}$. With this choice of m step 1 requires $O(|D|^{1/5} (\log|D|)^4)$ elementary operations and steps 10-12 require $O(|D|^{1/5} \log^2 |D| M(\log|D|))$ elementary operations.

Now the rest of the algorithm will be analysed. In steps 5-6 to construct a form $F_p = (p, B_p, C_p)$ we solve the equation

$$X^2 \equiv D \pmod{p} \qquad 0 < B_p < p$$

by reducing D (mod p) and applying Theorems 0.2.5 and 0.2.6. Since $p < C_1 \log^2|D|$, this construction requires $O(M\log|D|))$ elementary operations.

From the fact that $B_p < p < c \log^2|D|$ one can show that $\log \|F_p\| = O(\log|D|)$. Hence the reduction of F_p (step 6) requires by Theorem 0.3.7 $O(\log \|F_p\| M(\log \|F_p\|)) = O(\log |D|M(\log|D|))$ elementary operations. The computation of $F_p^{h^+}$ requires n+1 compositions and since

$$h^+ < \tilde{h} + b = O(\sqrt{|D|} \log|D|)$$

we have $n = O(\log |D|)$. Hence to compute $F_p^{h^+}$ using the algorithm of Corollary 1.1.11 requires $O(n \log |D|M(\log|D|)) = O(\log^2|D|M(\log|D|))$ elementary operations.

Step 13 requires $O(|D|^{1/8+\varepsilon})$ elementary operations using Pollard's factorization algorithm, since $h^+ = O(\sqrt{|D|} \log |D|)$. Loop 15-16 requires s_i compositions and since $h^+ = \prod_i q_i^{a_i}$ and $s_i < a_i$, we have $s_i = O(\log|D|)$. Hence loop 15-16 requires $O(s_i \log |D|M(\log |D|)) = O(\log^2|D|M(\log|D|))$ elementary operations (by Corollary 1.1.11). Loop 14-17 requires r iterations and since $r = O(\log h^+/\log\log h^+) = O(\log |D|)$, it terminates in $O(\log^3|D|M(\log|D|))$ elementary operations.

Finally loop 4-18 requires $O(\log^2|D|)$ iterations and from the above analysis, it terminates after $O(|D|^{1/5} \log^4|D|M(\log|D|))$ elementary operations in worst-case. Hence the theorem follows. □

REMARKS

1. In practice, it may be unnecessary to reduce the form F_p at step 6. In algorithm 4.1.13, if D is large enough to ensure that $|D|/\log^4|D| > C_1$, then choosing B_p such that $0 < |B_p| < p/2$ will ensure that F_p is reduced.

2. If one wants to implement algorithm 4.1.13, then it is necessary to know the constant C_1 of Theorem 4.1.5. This constant can be determined from [23], but we do not know its value. For practical purposes, very few forms are generally sufficient to generate $C\ell(D)$ (see [34] and Remark 4 below).

3. If one wants to compute the exponent (and further the class number) of $C\ell(D)$ without the assumption of unproven hypothesis one has to search in the interval $(0, 2\sqrt{|D|}\ell n|4D|)$ (see the bound on h(D) given by Theorem 4.1.9) using the "baby-giant step" strategy to find an h' such that $F_p^{h'} \approx I$. This will require $O(|D|^{1/4+\varepsilon})$ operations. Since the number of generators for which the order must be computed is $O(|D|^{C_3})$ (by Theorem 4.1.6), the algorithm will terminate in $O(|D|^{C_3+1/4+\varepsilon})$ elementary operations. It is known that $C_3 > \frac{1}{4}$ (see [22], [23]). This worst-case complexity bound seems to be unrealistic, but as stated in [22] it is not easy to prove a better bound on the number of generators of $\mathcal{D}_{2f}(d)$ unconditionally. (Observe that in the above version of the algorithm 4.1.13 we do not compute an approximation $\tilde{h}$ to h(D), since there is no known unconditional bound for the quantity $|\tilde{h} - h(D)|$ better than the absolute bound on h(D)!)

4. Algorithm 4.1.13 was designed by Shanks as a heuristic algorithm for computing the class number. There are two ways in which algorithm 4.1.13 may give information about the class number.

(i) If b is sufficiently large, there may, at some point in algorithm 4.1.13, be a unique value $h^* \equiv 0 \pmod{b}$ in the range

$$\tilde{h} - \varepsilon(m) < h^* < \tilde{h} + \varepsilon(m)$$

which is probably (and subject to the truth of the G.R.H. necessarily) the class number. In practice if this test is used the computation of the order of a few generators of $C\ell(D)$ is usually sufficient to determine the class number, as remarked by Shanks [34] and Shoof [33].

(ii) If all the p-Sylow subgroups of $C\ell(D)$ are cyclic, then $e = h(D)$, where e is the exponent of $C\ell(D)$. In practice, the p-Sylow subgroups for p odd prime, are almost always cyclic (few examples of class groups with non-cyclic p-Sylow subgroups are known! see [33]), so that almost always

$$h(D) = 2^s.e$$

The above methods for computing the class number are generally effective and fail with low probability. When the class group is irregular, the algorithm for determining the group structure may be required.

Those are standard algorithms (see e.g. Sims [39]) for finding the structure of a finite abelian group; it suits our purpose to describe an algorithm specially suited for use in conjunction with algorithm 4.1.13. Subject to the G.R.H, we describe an algorithm requiring $O(|D|^{1/4})$ elementary operations which determines the complete structure of $Cl(D)$. Since algorithm 4.1.13 outputs a generating set for each p-Sylow subgroup of $Cl(D)$ and there are at most $O(\log h(D)/\log\log h(D)) = O(\log|D|/\log\log D)$ p-Sylow subgroups of $Cl(D)$, it suffices to describe an algorithm which computes a basis given a generating set for a finite abelian group.

First we give some notations and results, basic background for the algorithm.

NOTATION 4.1.16

Suppose that H is group. If $\{b_1,b_2,\ldots,b_n\}$ generates H, we write that $H = \langle b_1,b_2,\ldots,b_n\rangle$ and if $\{b_1,b_2,\ldots,b_n\}$ is a basis for H, we write $H = \langle\langle b_1,b_2,\ldots,b_n\rangle\rangle$.

THEOREM 4.1.17

Let $H = \langle\langle b_1,b_2,\ldots,b_n\rangle\rangle$, $H_1 = \langle H,x\rangle$ be abelian groups, where b_i has order p^i for $1 \leq i \leq n$ and x has order p^h for p prime. If R is the set of relations of the form

$$x^{\beta} = \prod_{i=1}^{n} b_i^{\gamma_i} \tag{4.13}$$

where $0 < \gamma_i < p^{\alpha_i}$ for $1 < i < n$ and $1 < \beta < p^h$. Then

(i) $R \neq \emptyset$

(ii) Let $$x^{\beta^*} = \prod_{i=1}^{n} b_i^{\delta_i} \tag{4.14}$$

be a relation of R, where β^* is the smallest exponent of x which appears in R. Then

$$R^* = \{x^{\beta^*} = \prod b_i^{\delta_i},\ b_i^{p^{\alpha_i}} = 1 \text{ for } 1 < i < n, x^{p^h} = 1\}$$

is a set of defining relations of H_1.

(iii) For some $0 < k < h$, we have that $\beta^* = p^k$.

Proof

(i) We have that $R \not\emptyset \emptyset$, because $x^{p^h} = \prod_{i=1}^{n} b_i^{o} \in R$.

(ii) Suppose that (4.13) is a relation in R. Then we choose an integer m such that

$$0 < \beta - m\beta^* < \beta^* \tag{4.15}$$

By raising both of the sides of (4.14) to the exponent -m and multiplying by (4.13) we have

$$x^{\beta - m\beta^*} = \prod_{i=1}^{n} b_i^{\gamma_i - m\delta_i}$$

Since β^* is the smallest exponent of x in relation of this form, from (4.15) we have that $\beta - mb^* = 0$ and thus $\beta^* | \beta$.

Now since $\{b_1, b_2, \ldots, b_n\}$ is a basis and thus the b_i's are independent we have that R is the set of the relations of the form

$$x^{\beta^* t} = \prod_{i=1}^{n} b_i^{\delta_i t} \quad \text{for } 1 \leq \beta^* t < p^h$$

Hence easily we have that R^* is a set of defining relations of H_1.

(iii) Since $x^{p^h} = \prod_{i=1}^{n} b_i^{o}$ is a relation of R, we have that $\beta^* | p^h$ and thus $\beta^* = p^k$ for some $0 \leq k < h$. □

A procedure (based on 4.1.17) computing the complete structure of a finite abelian p-group G_p will now be described. Suppose that $\{x_1, \ldots, x_g\}$ is a set of generators of G_p. The structure of G_p will be computed by computing recursively a basis of the subgroup H_{s+1} of G_p, then

$$H_{s+1} = \langle H_s, x_{s+1} \rangle \text{ for } 0 \leq s < g-1 \text{ and } H_o = \{1\}$$

Suppose that $H_s = \langle\langle b_1, b_2, \ldots, b_n \rangle\rangle$, where b_i has order p^{α_i} for $1 \leq i \leq n$ and x_{s+1} has order p^h. First we compute β^*, δ_i for $1 \leq i \leq n$ such that

$$x_{s+1}^{\beta^*} = \prod_{i=1}^{n} b_i^{\delta_i} \tag{4.16}$$

and β^* is as in Theorem 4.1.18. Then $\beta^* = p^k$.

Case k = h

Since the set R* of defining relations of H_{s+1} is

$$\{x_{s+1}^{p^h} = 1,\ b_i^{p^{\alpha_i}} = 1,\ 1 \leq i \leq h\}$$

we have that $H_{s+1} = \langle\langle b_1,\ldots,b_n,b_{n+1} = x_{s+1}\rangle\rangle$

Case 0 < k < h

Let $\gcd(\beta^*,\delta_1,\ldots,\delta_n) = p^r$ with $0 \leq r \leq k$

$$u = x_{s+1}^{\beta'} \prod_{i=1}^{n} b_i^{\gamma_i} \text{ with } \beta' = \beta^*/p^r \text{ and } \gamma_i = -\delta_i/p^r$$

Then u has order p^r, since

$$u^{p^r} = x_{s+1}^{\beta^*} \prod_{i=1}^{n} b_i^{-\delta_i} = 1$$

and β^* is the smallest exponent of the relations of R.

Subcase r = k

Then $\beta' = 1$ and $u = x_{s+1} \prod_{i=1}^{n} b_i^{\gamma_i}$. Then $H_{s+1} = \langle H_s, x_{s+1}\rangle = \langle H_s, u\rangle$ and the set R* of the defining relation of H_{s+1} becomes

$$\{u^{p^k} = 1,\ b_i^{p^{\alpha_i}} = 1,\ 1 \leq i \leq n\}$$

Hence we have $H_{s+1} = \langle\langle b_1,\ldots,b_n,b_{n+1} = u\rangle\rangle$.

If $0 \leq r < k$, then for some j, $1 \leq j \leq n$, we have that

$$\gcd(\gamma_j,p) = 1.$$

Then there exist integers λ, ν such that

$$\lambda\gamma_3 + \nu p^{\alpha_j} = 1$$

Hence we have

$$b_j = b_j^{\lambda\gamma_j + \nu p^{\alpha_j}} = u^{\lambda}\, x_{s+1}^{-\lambda\beta'} \prod_{\substack{i=1 \\ i\neq j}}^{n} b_i^{-\lambda\gamma_i}$$

and thus $b_j \in \langle u, x_{s+1}, b_1, \ldots, b_{j-1}, b_{j+1}, \ldots, b_n \rangle = H_{s+1}$.

Sub-case r = 0

Then we have $u = 1$ and $b_j \in \langle H_s', x_{s+1} \rangle = H_{s+1}$ with $H_s' = \langle\langle b_1, \ldots, b_{j-1}, b_{j+1}, \ldots, b_n \rangle\rangle$. Hence it suffices to recursively compute a basis $H_{s+1} = \langle H_s', x_{s+1} \rangle$.

Sub-case r $\neq$ 0,k

Then using the definition of u and β^*, it can be shown that the set $\{u, b_1, \ldots, b_{j-1}, b_{j+1}, \ldots, b_n\}$ has independent elements. Hence it suffices to recursively compute a basis of $H_{s+1} = \langle H_s'', x_{s+1} \rangle$ with $H_s'' = \langle\langle u, b_1, \ldots, b_{j-1}, b_{j+1}, \ldots, b_n \rangle\rangle$.

To justify the above procedure, it suffices to show that the recursive calls lead to simpler subproblems. It is clear that H_s' is a proper subgroup of H_s, whence $|H_s'| < \frac{1}{p} |H_s|$. Moreover

$$|H_s''| = p^r \prod_{\substack{i=1 \\ i\neq s}}^{n} p^{\alpha_i} < \frac{1}{p} |H_s| \qquad (4.17)$$

since $|H_s| = \prod_{i=1}^{n} p^{\alpha_i}$ and $p^r < p^{\alpha_j}$ from the facts $p^r | \delta_j$ and $\delta_j < p^{\alpha_j}$. Hence to find a basis for H_{s+1} requires at most t recursive calls of the procedure, where $|G_p| = p^t$.

To find a basis of G_p, g applications of the above procedure are required.

It remains to give an efficient way of computing the relation (4.16). We shall use the baby-giant step strategy. First we compute for $1 < i < n$

$$b_i^{-t_i} \quad \text{for all } 0 < t_i < \lceil p^{\alpha_i/2} \rceil \text{ (baby steps).}$$

Let now L be a list of all possible products of $b_i^{t_i}$'s. Then we can see that $\#L = \alpha(\prod_{i=1}^{n} p^{\alpha_i})^{1/2})$.

Now we compute the list L* such that

$$L^* = \{ax^{p^k} : a \in L, 0 < k < h\}.$$

Then one can see that $\#L^* = (\#L).(h+1)$. Moreover we compute for $1 < i < n$

$$b_i^{c_i d_i} \quad \text{for all } 0 < c_i < \lceil p^{\alpha_{1/2}} \rceil \text{ with } d_i = \lceil p^{\alpha_{i/2}} \rceil$$

(giant steps)

Now let L** be a list containing all the possible products of $b_i^{c_i d_i}$'s. Then we have $\#L^{**} = O((\prod_{i=1}^{n} p^{\alpha_i})^{1/2})$. Now for every element w of L** we search if w belongs to L* (using binary search). Hence we shall find a set of relations of the form

$$\prod_{i=1}^{n} b_i^{c_i d_i} = x_{s+1}^{p^k} \prod_{i=1}^{n} b_i^{-t_i}$$

for some $0 \leq c_i < \lceil p^{\alpha_1/2} \rceil$ $0 \leq t_i < \lceil p^{\alpha_i/2} \rceil$ and $0 \leq k \leq h$. Choosing the relation with the smallest k we have the relation (4.16).

From the above one can see that the computation of the relations requires $O(h(\#L)) = O(h(\prod_{i=1}^{n} p^{\alpha_i})^{1/2})$ multiplications and since $h = O(\log|G|)$ and $\prod_{i=1}^{n} p^{\alpha_i} \leq |G|$, we have that the computation of the relation requires $O(\log|G|\sqrt{|G|})$ multiplications.

And now explicitly the algorithm.

ALGORITHM 4.1.1[8]

INPUT : A set of generating sets $S_p = \{x_1,\ldots,x_g\}$ for every p-Sylow subgroup of an finite abelian group G and their orders.

OUTPUT : A basis for every p-Sylow subgroup of G.

Procedure BASIS ($H_s = \langle\langle b_1,b_2,\ldots,b_n\rangle\rangle$, x_{s+1}, $p^{\alpha_1},\ldots,p^{\alpha_n},p^h$)

Comment This procedure computes a basis of the p-group $H_{s+1} = \langle H_s, x_{s+1}\rangle$. Also p^{α_i} is the order of b_i for $1 \leq i \leq n$ and p^h the order of x_{s+1}.

Begin

1. <Compute the relation $X_{s+1}^{\beta^*} = \prod_{i=1}^{n} b_i^{\delta_i}$ as in (4.16)>;

Comment This can be done with the baby-giant step strategy described above

$k \leftarrow \log_p \beta^*$

Case k of

Begin

0 : Return $H_{s+1} = \langle\langle b_1, b_2, \ldots, b_n \rangle\rangle$;

h : Return $H_{s+1} = \langle\langle b_1, b_2, \ldots, b_n, x_{s+1} \rangle\rangle$;

2. else $r \leftarrow \log_p (\gcd(\beta^*, \delta_1, \ldots, \delta_n)$;

3. $u \leftarrow x_{s+1}^{\beta^*/p^r} \prod_{i=1}^{n} b_i^{-\delta_i/p^r}$;

Case r of

Begin

k : Return $H_s = \langle\langle b_1, b_2, \ldots, b_n, u \rangle\rangle$

0 : BASIS $(H_s' = \langle\langle b_1, \ldots, b_{j-1}, b_{j+1}, \ldots, b_n \rangle\rangle, x_{s+1},$
$p^{\alpha_1}, \ldots, p^{\alpha_j - 1}, p^{\alpha_j + 1}, \ldots, p^{\alpha_n}, p^h)$

else

5. BASIS $(H_s'' = \langle\langle b_1, \ldots, b_{j-1}, b_{j+1}, \ldots, b_n, u \rangle\rangle,$
$x_{s+1}, p^{\alpha_1}, \ldots, p^{\alpha_{j-1}}, p^{\alpha_{j+1}}, \ldots, p^{\alpha_n}, p^r)$

end

Begin

6. For <each p-Sylow subgroup G_p of G> do

Begin

7. <Let $G_p = \langle x_1,\ldots,x_g \rangle$ >;

$H_o \leftarrow 1$;

8. For s = 0 until g-1 do

BASIS (H_s, x_{s+1})

$B_p \leftarrow H_g$

9. end

Return $\langle B_p$ for each $G_p \rangle$;

end

THEOREM 4.1.19

Algorithm 4.1.18 correctly computes bases of all the p-Sylow subgroups of G.

Proof

It follows from the description above 4.1.19. □

We shall analyse algorithm 4.1.18 in the case G = Cℓ(D).

THEOREM 4.1.20

There exists an algorithm which computes bases for all the p-Sylow subgroups of Cℓ(D) with D < 0 in $O(|D|^{1/4}(\log|D|)^7 M(\log|D|))$ elementary operations on the assumption of the truth of the G.R.H.

Proof

First we compute a generating set G_p for every p-Sylow subgroup of $C\ell(D)$ and the orders of the generators using algorithm 4.1.13. After we compute bases for every p-Sylow subgroup of $C\ell(D)$ using algorithm 4.1.19.

First the procedure BASIS will be analysed. Step 1 requires $O(\log(h(D))\sqrt{h(D)})$ compositions, since $h(D) = |C\ell(D)|$ (see description above 4.1.19). Since $h(D) = O(\sqrt{|D|}\log|D|)$ by Proposition 4.1.9 and composition (following by reduction) requires $O(\log|D|\ M(\log|D|))$ elementary operations by Corollary 1.1.11, we have that step 1 requires $O(|D|^{1/4}(\log|D|)^2\ M(\log|D|))$ elementary operations.

Since $\beta^* < h(D)$, $\prod_{i=1}^{n} \delta_i < h(D)$ and $n < g = O(\log^2|D|)$ by Theorem 4.1.5), we have that step 2 requires $O(\log^2|D|M(\log|D|)\log\log|D|)$ elementary operations for n applications of the Euclidean algorithm (see Theorem 0.2.3).

One can compute u by expressing $\beta' = \beta^*/p^r, \gamma_i = -\delta_i/p^r$ for $1 \leq i \leq n$, in binary (see steps 7-9 of 4.1.13). This requires $O(\log\beta' + \sum_{i=1}^{n}\log\gamma_i)$ compositions and since $\prod_{i=1}^{n}\gamma_i < h(D) = O(\sqrt{|D|}\log|D|)$, it requires $O((\log|D|)^2M(\log|D|))$ elementary operations (by Corollary 1.1.11).

From (4.17) (see comment below it) the procedure BASIS is called recursively at most $t = O(\log^2|D|)$ times (steps 4-5).

From the above analysis we conclude that the procedure BASIS requires $O(|D|^{1/4}(\log|D|)^4 M(\log D))$ elementary operations in worst-case.

The main algorithm will be analysed now. To find the generating sets G_p and the order of the generators using algorithm 4.1.13, it requires $O(|D|^{1/5} \log|D| M(\log|D|))$ elementary operations.

To find a basis for a p-Sylow subgroup (steps 8-9) requires $g = O(\log^2|D|)$ applications of the procedure BASIS and thus requires $O(|D|^{1/4}(\log|D|)^6 M(\log|D|))$ elementary operations. Loop 6-9 requires $O(\lg e/\log\log e)$ iterations (the number of distinct prime divisors of the exponent e of $Cl(D)$) and thus $O(|D|^{1/4}(\log|D|)^7 M(\log|D|))$ elementary operations, since $e < h(D)$. □

Remarks

1. A more detailed analysis may reduce the exponent 7 of the running time of the algorithm. The presence of the term $|D|^{1/4}$ in the running time of the algorithm discourages us from doing it.

2. One may use Sims' method ([39]) for the computation of the bases above, which is rather inefficient. His method has running time $O(|D|^{1+\varepsilon})$ but requires $O(|D|^c)$, $c > 1$ bits for memory which may cause overflow.

If a basis $\{b_1, b_2, \ldots, b_n\}$ for the p-Sylow subgroup $Cl_p(D)$ of $Cl(D)$ is known and b_i has order p^{α_i} for $1 \leq i \leq n$, then

$$Cl_p(D) = G(p^{\alpha_1}) \times G(p^{\alpha_2}) \times \ldots \times G(p^{\alpha_n}) \text{ and}$$

$$|Cl_p(D)| = \prod_{i=1}^{n} p^{\alpha_i}$$

where $G(p^{\alpha_i})$ is a cyclic group of order p^{α_i}.

Since moreover $Cl(D) = \prod_p Cl_p(D)$ the class number $h(D)$ (the order of $Cl(D)$ is the product of the orders of all the p-Sylow subgroups $Cl_p(D)$ of $Cl(D)$. Hence we have the following theorem.

THEOREM 4.1.22

There exists an algorithm which computes the class number $h(D)$ in $O(|D|^{1/4}(\log|D|)^7 M(\log D))$ elementary operations on the assumption of the truth of G.R.H. □

TABLE II

CLASS NUMBER

$-1 < D < -50$

-D	h(D)	-D	h(D)	-D	h(D)
1	1	18	2	35	6
2	1	19	3	36	4
3	1	20	4	37	2
4	1	21	4	38	6
5	2	22	2	39	4
6	2	23	3	40	4
7	1	24	4	41	8
8	2	25	2	42	4
9	2	26	6	43	3
10	2	27	3	44	6
11	3	28	2	45	4
12	2	29	6	46	4
13	2	30	4	47	5
14	4	31	3	48	4
15	2	32	4	49	4
16	2	33	4	50	6
17	4	34	4		

$1 < D < 50$ non-square

D	h(D)	D	h(D)	D	h(D)
2	1	19	2	35	4
3	2	20	2	37	3
5	1	21	2	38	2
6	2	22	2	39	4
7	2	23	2	40	4
8	2	24	4	41	1
10	2	26	2	42	4
11	2	27	2	43	2
12	2	28	2	44	2
13	1	29	1	45	2
14	2	30	4	46	2
15	4	31	2	47	2
17	1	32	2	48	4
18	2	33	2	50	2
		34	4		

4.2. CLASS NUMBER OF REAL QUADRATIC FIELDS

A. GAUSS METHOD

DEFINITION 4.2.1

The number of classes of properly primitive forms of fixed non-quadratic determinant $D > 0$ is called the *class number* and is denoted by $h(D)$.

Gauss derived counting methods for the computation of $h(D)$ for $D > 0$ are similar to those described entirely for $D < 0$. The difficulty in the case of $D > 0$ is that equivalent forms are not as easily recognized as in the case of $D < 0$. Hence it is necessary to compute the period of reduced forms to reject equivalent ones.

METHOD 4.2.2

Gauss' first counting method makes use of the fact that every reduced indefinite form satisfies:

$$|\sqrt{D} - |A|| < B < \sqrt{D} \quad \text{and} \quad |A| < 2\sqrt{D} .$$

The method is:

1. For $|A| < 2\sqrt{D}$ find all pairs (A,B) such that $X = B$ satisfies

   $$X^2 \equiv D \bmod A \qquad |\sqrt{D} - |A|| < x < \sqrt{D}$$

   and for each pair (A,B), let (A,B,C) be a form where $C = (B^2 - D)/A$.

2. Reject th4 forms which are not reduced.

3. Reject the forms which are not properly primitive.

4. Partition the remaining forms into equivalence classes by computing cycles of reduced forms as appropriate and select a representative from each equivalence class.

5. The number of remaining forms is the class-number h(D).

The method requires $O(D^{1+\varepsilon})$ elementary operations. More detailed analysis shows that the algorithm terminates in $O(D(\log D)^3 M(\log D))$ elementary operations.

METHOD 4.2.3

Gauss' second counting method for computation of h(D), for $D > 0$ is:

1. For $0 < B < \sqrt{D}$ factor $B^2 - D = AC$ and for all possible pairs (A,C) let (A,B,C), (C,B,A) be a form.

2. Reject the forms which are not reduced.

3. Reject the forms which are not properly primitive.

4. Reject equivalent forms (as in step 5 of 4.2.2)

5. The number of the remaining forms is the class-number h(D).

One can see with detailed analysis that the method yields the class number in $O(D(\log D)^3 M(\log D))$ elementary operations.

B. DIRICHLET'S FORMULAE

Dirichlet gave the following formula for $h(D)$, with $D > 0$ (see Mathews [26], 238)

$$h(D) = \frac{2\sqrt{|D|}}{R} \sum_{k=0}^{\infty} \frac{1}{2k+1} \left(\frac{D}{2k+1}\right) \tag{4.18}$$

where R is the regulator.

Now since

(i) $\left(\frac{D}{2k+1}\right) = \left(\frac{4D}{2k+1}\right)$ and

(ii) $\left(\frac{4D}{n}\right) = 0$ for n even (Kronecker's symbol)

we have $\sum_{k=0}^{\infty} \frac{1}{2k+1} \left(\frac{D}{2k+1}\right) = \sum_{n=1}^{\infty} \left(\frac{4D}{n}\right) \frac{1}{n} = L(1, \chi_{4D})$

Hence using the Theorem 2.2.3 we have:

$$h(D) = \frac{2\sqrt{D}}{R} L(1, \chi_{4D}) = \frac{2\sqrt{D}}{R} \prod_{\substack{p=2 \\ p \text{ prime}}}^{\infty} \frac{p}{p - \left(\frac{4D}{p}\right)} \tag{4.19}$$

Moreover since $\left(\frac{4}{p}\right) = 1$ and $\left(\frac{4}{2}\right) = 0$ we have

$$h(D) = \frac{2\sqrt{D}}{R} \prod_{p=3}^{\infty} \frac{p}{p - \left(\frac{D}{p}\right)} \tag{4.20}$$

and if $D \not\equiv 3 \bmod 4$, then we have

$$h(D) = \frac{\sqrt{D}}{R} L(1, \chi_D) \left(2 - \left(\frac{D}{2}\right)\right)$$

Also the formula relating to the class number h(D) and $h(D_o)$ where $D = D_o S^2$ is:

$$h(D) = h(D_o)\, S \frac{R}{R'} \prod_{p|S} \frac{p - (\frac{D}{p})}{p} \qquad (4.21)$$

where R, R' are the regulators of the orders $\mathcal{O}_{2f}(d)$, $\mathcal{O}_{2f'}(d)$ respectively with $D = f^2 d$ and $D_o = (f')^2 d$ (see Mathews [26], p. 166).

Moreover there is a formula given by Dirichlet expressed in finite sum of Kronecker's symbols for D square free

$$h(D) = \begin{cases} \dfrac{-2 - (\frac{2}{D})}{R} \sum_{k=1}^{D-1} (\frac{K}{D}) \log \sin \dfrac{\pi k}{D} & \text{for } D \equiv 1 \bmod 4 \\ -\dfrac{1}{R} \sum_{k=1}^{2D-1} (\frac{D}{2k+1}) \log \sin \dfrac{\pi(2k+1)}{4D} & \text{for } D \equiv 2,3 \bmod 4 \end{cases} \qquad (4.22)$$

(see Venkov [40], p. 230)

Thus to compute the class number h(D) using the formulae (4.21), (4.22), requires O(D log D M(log D)) elementary operations in worst-case using one of the methods of the section 3.1 for the computation of the regulator and computing the Kronecker's symbols as in Theorem 4.1.4.

C. SHANKS ALGORITHM

Shanks' algorithm for the computation of the class number with D > 0 differs from algorithm 4.1.13 - 4.1.19 in the following respects.

1. The approximation $\tilde{h}$ of $h(D)$ is computed via the formula (4.20). To find an approximation, the computation of the regulator R of $\mathcal{O}_{2f}(d)$ with $D = f^2 d$ is necessary. To compute R one can use Lenstra-Shanks algorithm (section 3.1).

2. In algorithms 4.1.13 and 4.1.19 equivalent forms are easily recognized, since they are definite by using Theorem 3.2.2. But when $D > 0$ to decide equivalence of two indefinite forms F, Q it is necessary to use algorithm 3.1.24 to decide whether or not QF^{-1} is principal.

Hence if one modifies the algorithms 4.1.13 and 4.1.19 in respect with (1) and (2) then similarly with the case $D < 0$ we have the following theorems.

THEOREM 4.2.4.

Assuming the truth of G.R.H. there exists an algorithm which computes the complete structure of $C\ell(D)$ in $O(D^{1/2+\varepsilon})$ elementary operations, when $D > 0$, non-square

Proof

The bottleneck of the method (described above) is the computation of the relation (4.16). It requires $O(D^{1/4+\varepsilon})$ applications of the equivalence procedure (algorithm 3.1). Hence the computation of the complete structure of $C\ell(D)$ requires $O(D^{1/2+\varepsilon})$ elementary operations. □

COROLLARY 4.2.5

Assuming the truth of G.R.H. there exists an algorithm which computes the class-number $h(D)$ of $Cl(D)$ with $D > 0$ non-square in $O(D^{1/2+\varepsilon})$ elementary operations.

Proof

See remark above the Theorem 4.1.22. □

5.1. ALGORITHM CLASNO

DEFINITION 5.1.1.

A class of $C\ell(D)$ which contains an ambiguous form is called an *ambiguous class*.

PROPOSITION 5.1.2.

A class K of $C\ell(D)$ is ambiguous if and only if

$$K^2 = I$$

where I is the principal class of $C\ell(D)$.

Proof

($\Rightarrow$) Let (A,B,C) be an ambiguous form in K. Then $A \mid 2B$.

Let $2B = -mA$.

Then

$$(A,B,C) \to (A,-B,C) \text{ via } \begin{pmatrix} 1 & m \\ 0 & 1 \end{pmatrix}$$

and thus $(A,B,C) \approx (A,-B,C)$.

Now since $(A,-B,C) \in K^{-1}$, we have

$$K = K^{-1}.$$

Now we have

$$K^2 = K \circ K = K \circ K^{-1} = I$$

($\Leftarrow$) If $K^2 = I$, then

$$K^{-1} = K^{-1} \circ I = K^{-1} \circ K^2 = K$$

Hence every form (A,B,C) in k is equivalent to its inverse (A,-B,C) and thus by Gauss [16] (A.162) there exists an ambiguous form equivalent to (A,B,C). □

PROPOSITION 5.1.3.

The reduced forms of an ambiguous class of $C\ell(D)$ with $D < 0$ are of the shape (a,o,c), (a,b,a), (2b,b,c).

Proof

Let K be an ambiguous class of $C\ell(D)$ and (a,b,c) a reduced form in K. Then $(a,-b,c) \in K$, since K ambiguous. Moreover (a,-b,c) is also reduced.

Now from Theorem 3.2.1 we have that either a = c or $a|2b$.

If a = c, then the class K contains only the reduced forms (a,b,a), (a,-b,a).

If $a|2b$, then since (a,b,c) is reduced we have

$$|2b| < |a|$$

and thus

$$b = 0 \quad \text{or} \quad 2b = a.$$

Hence the class K contains in this case only the reduced form (a,o,c) or the reduced forms (2b,b,c), (2b,-b,c). □

PROPOSITION 5.1.4

Suppose that e is the exponent of $C\ell(-D)$ with $D > 0$. If e is odd, then D is a prime-power.

Proof.

Assume that D is not a prime-power. Then there exist integers f, g such that $f \neq 1$, $g \neq 1$, $f < g$, $D = f.g$ and $\gcd(f,g) = 1$. Now the form $(f,o,-g)$ is a properly primitive ambiguous form with determinant -D. Moreover using the fact that $(f,o,-g)$ is reduced one can show that $(f,o,-g) \not\approx (1,0,D) = I$.

By Proposition 5.1.1 we have

$$(f,o,-g)^2 \approx I$$

hence $2|e$, and the exponent is even. □

A version of the algorithm CLASSNO (see Shanks [34]) for factorization of an integer D with running time $O(|D|^{1/5+\varepsilon})$ elementary operations is the following.

THEOREM 5.1.5.

There exists an algorithm which factors a positive integer D in two factors f,g with $\gcd(f,g) = 1$ and if D is not prime-power, then yields $f \neq 1$, $g \neq 1$. This algorithm (CLASSNO) factorizes D in $O(|D|^{1/5}(\log|D|)^4\, M(\log|D|))$ elementary operations.

Proof

First we compute the exponent e and a set G_2 of generators and their order of the 2-Sylow subgroup of $C\ell(-D)$ using algorithm 4.1.13. If the exponent is odd then by Proposition 5.1.4 we have that D is a prime power. Otherwise for a $Q \in G_2$ with order 2^{j+1} we compute the form $G = Q^{2^j}$ using the algorithm of Corollary 1.1.11. Then one can see that $G^2 \approx I$ and by 5.1.2 we have that G belongs to an ambiguous class. Also G is reduced (by 1.1.11). Now by Theorem 5.1.3 we have that G is of the shape either (a,o,c) or (2b,b,c) or (a,b,a). Hence we have either

$$-D = ac \tag{5.1}$$

$$\text{or} \quad -D = b^2 - 2bc = b(b-2c) \tag{5.2}$$

$$\text{or} \quad -D = b^2 - a^2 = (b-a)(b+a) \tag{5.3}$$

Moreover using the fact that Q properly primitive, one can show that the factors of D given by (5.1) - (5.3) are coprime.

It is possible (5.2) to give the trivial factorization 1.(-D). Using the fact that G is a non-principal properly primitive reduced form, one can show that trivial factorization could happen only when $D \equiv 1 \pmod 4$, then $Q = (2,1,(D+1)/2)$ and the trivial factorization is given by (5.2).
Hence in the case $D \equiv 1 \pmod 4$ if (5.2) yields a trivial factorization, then we use other forms of G_2 to compute

ambiguous forms. If all the forms of G_2 lead to the ambiguous form $(2,\pm,(D+1)/2)$, then it can be shown that in this case D is prime power. (This follows from the theorem on generators of the ambiguous classes of $C\ell(D)$ cited by Lagarias in [21], p. 500).

We shall now analyse the algorithm. First, to compute the exponent of $C\ell(-D)$ and G_2 (by Theorem 4.1.15) requires $O(D^{1/5}(\log D)^4 M(\log D))$ elementary operations.

Suppose that $e = 2^s.t$ and $2 \nmid t$.

Then to compute to form G requires j+1 compositions. Since

$$J < s < \log e < \log(h(-D)) = O(\log D)$$

we have that the computation of an ambiguous form requires $O(J \log D\, M(\log D)) = O((\log D)^2\, M(\log D))$ elementary operations by Corollary 1.1.11. And in the case of trivial factorization we have to compute at most $\#G_2 = D(\log^2 D)$ ambiguous forms, hence the bottleneck of this step is $O((\log D)^4 M(\log D))$ elementary operations. From the above analysis the running time of the algorithm follows. □

For alternative approaches to factorization via quadratic forms see Lehmers' method [24] (an $O(\sqrt{D})$ method but the facilities of the University of California make it competitive!), SQUFOF (Shanks-unpublished: see Monier [28] - an $O(D^{1/4+\varepsilon})$ (expected time) simple method using searching on the principal cycle), Schnorr [31] (a probabilistic version of the algorithm CLASSNO).

REFERENCES

[1] L. Adleman, G.Miller, and K. Manders, "On taking roots in finite fields", 18th IEEE Symposium on Foundations of Computer Science, 1977.

[2] A. Aho, J. Hopcroft, and J. Ullman, "The design and Analysis of Computer algorithms" Addison-Wesley, Reading, Mass. 1974.

[3] N. Ankeny, " The least quadratic nonresidue", Ann. of Math. (2) 55 (1952), p. 65-72.

[4] Apostol, T.M., "Introduction to Analytic Number Theory", Springer-Verlag, New York, (1976).

[5] Ayoub, R., "An introduction to the analytic theory of numbers" American Math. Soc., Rhode Island, (1963).

[6] Beynon, W.M., "On Raney's binary encoding for continued fractions", Theory of computation, U.W. Rep. 34 (1981).

[7] Beynon, W.M., Iliopoulos, C.S., "Gauss' algorithm for the solution of Quadratic Diophantine equations" Theory of Computation, U.W. Rep. 37 (1981).

[8] Borevich, Z.I., Shafarevich, I.R., " Number theory", Academic Press, 1966.

[9] Buell, D.A., "Class groups of quadratic fields", Math. Comp. 30 (1976), pp. 610-623.

[10] Burgess, D.A., "The distribution of quadratic residues and non-residues", Mathematica 4 (1957), pp. 106-112.

[11] Burgess, D.A., "On characters sums and L-series", Proc. London Math. Soc. (3) 12 (1962), pp. 193-206.

[12] Cassels, J.W.S., "Rational quadratic forms", Academic Press, London, (1978).

[13] Cohn, H., "A second course in number theory", Wiley, New York, 1962.

[14] Dickson, L.E., "History of the theory of numbers" Volumes I-III, Chelsea, New York, (1966).

[15] Dickson, L.E., "Introduction to the theory of numbers", Univ. of Chicago Press, Chicago, 1929.

[16] Gauss, C.F., "Disquisitiones Arithmeticae", 1801; English transl., Yale Univ. Press, New Haven, Conn. 1966.

[17] Hall, M., Jr., "The theory of groups", Macmillan, New York, 1959.

[18] Iliopoulos, C.S., "Analysis of an algorithm for composition of binary quadratic forms" (to appear in J. Algorithms).

[19] Knuth, D., "Seminumerical algorithms", Addision-Wesley, Reading, Mass., 1969.

[20] Lagarias, J.C., "Worst-case Complexity bounds for Algorithms in the theory of integral quadratic forms", J. Algorithms 1 (1980), pp. 142-186.

[21] Lagarias, J.C., "On the computational complexity of determining the solvability or unsolvability of the equation $x^2 - Dy^2 = -1$", Trans. Amer. Math. Soc., 260 (2) 1980, pp.485-508.

[22] Lagarias, J.C., Montgomery, H.L., Odlyzko, A.M., "A bound for the least prime ideal in the Chebotarev Density theorem" Inv. Math. 54, (1979), 271-296.

[23] Lagarias, J.C., Odlyzko, A.M., "Effective versions of the Chebotarev density theorem" in "Algebraic Number Fields" (A. Fröhlich, Ed.), pp. 409-464, Proceedings, 1973 Durham Symposium, Academic Press, New York, 1977.

[24] Lehmer, D.H., and Emma, "A new factorization technique using quadratic forms", Math. of Comp., 28 (126) 1974 pp. 625-635.

[25] Lenstra, H.W., Jr., "On the calculation of Regulators and class numbers of quadratic fields" Univ. of Amsterdam, Rep. 80-08, (Preprint), October (1980).

[26] Mathews, G.B., "Theory of Numbers" 2nd ed., Chelsea, New York, 1961 (Reprinted)

[27] Miller, G., "Riemann's hypothesis and tests for primality", J. Comput. System Sci. 13 (1976), 300-317.

[28] Monier,L., "Algorithmes de factorisation d'entiers, Thèse, 3^{me} cycle, Orsay, 1980.

[29] Pall, G., "Some aspects of Gaussian composition", Acta Arithmetica 24 (1973), pp.401-409.

[30] Pollard, J.M., "Theorems on factorization and primality testing", Proc. Cambridge Philos. Soc. 76 (1974) pp. 521-528.

[31] Schnor, C.P., "Refined analysis and some improvements in some factoring algorithms" Stanford University Technical Report (1980).

[32] Schönhage, A., "Schnelle berechnung von Kettenbruchen-twicklungen" Acta Informatica 1, (1971), pp. 139-144.

[33] Scoof, R.J., "Quadratic fields and factorization" in Studieweek Getaltheorie en computers, Univ. of Amsterdam, Mathematisch centrum (1980).

[34] Shanks, D., "Class number, a theory of factorization and genera" in 1969 Number Theory Institute ; Proc. Symp. Pure Math. 20 (1971), 415-440.

[35] Shanks, D., "Five number theoretic algorithms" in Proceedings, 2nd Manitoba Conference on Numerical Mathematics, 1972", pp. 51-70.

[36] Shanks, D., "Infrastructure of the real quadratic fields and its applications" in "Proceedings, 1972 Number theory Conference, University of Colorado, Boulder, 1972", pp. 217-224.

[37] Shanks, D., "Gauss ternary form reduction and the 2-Sylow subgroup" Math. Comp. 25 (1971), pp. 837-853.

[38] Smith, H.J.S., "Report on the Theory of Numbers", Chelsea, New York (1965).

[39] Sims, C.C., "The influence of computers on algebra" in Proc. of Symposia in App. Math., Amer. Math. Soc., (1974), pp. 13-30.

[40] Venkov, B.A., "Elementary number theory", Engl. trans. (by Alderson, H.) Wolters-Noordhoff publishing groningen, The Netherlands, (1970).