International Journal of Mathematics
and Computer Science, 1(2006), 83-116

Efficient Ideal Reduction in Quadratic Fields

Michael J. Jacobson, Jr., Reginald E. Sawilla, Hugh C. Williams

Department of Mathematics and Statistics, Department of Computer Science
University of Calgary
Calgary, Alberta, Canada, T2N 1N4
e-mail:jacobs@cpsc.ucalgary.ca, reg@sawilla.com, williams@cpsc.ucalgary.ca

(Received September 1, 2005, Accepted October 17, 2005)

Abstract

This paper is a survey of all known ideal reduction algorithms for quadratic fields. As most of these algorithms were developed in the language of positive definite binary quadratic forms, we have generalized them to work with ideals of both positive and negative discriminant and, in the real case, developed a general method for computing the relative generator necessary for infrastructure computations. In addition, we present a new reduction method which combines ideas from the fastest known algorithms. The algorithms are implemented and compared in both reduction-only and multiplication-reduction settings. We show that Schönhage's algorithm is asymptotically the fastest reduction algorithm but not useful in practice, the formulation of NUCOMP from [6] is the fastest practical method when multiplying and reducing, and our new algorithm is the fastest method when only reduction is required.

1 Introduction

A quadratic number field is an extension of the field of rational numbers, formed by adjoining the square-root of a non-square integer. The rationals contain a special subset of numbers, namely the rational integers. The concept of an integer may be extended to quadratic number fields and, as with the rational integers, these integers have the algebraic structure of a ring.

Ideals of the ring of integers of a quadratic number field have many applications including: finding solutions to the Pell equation, calculating the fundamental unit and regulator of a real quadratic field, factoring integers, and cryptographic key agreement protocols. When ideals are multiplied in these applications, the bit length of the parameters characterizing the product grows quickly and soon becomes unmanageable.

Reduction is the process of finding an ideal with parameters of a bounded size that is equivalent to another ideal. There are two circumstances when reduction is required. Most often, it is required after multiplying two reduced ideals, for example, cryptographic public key exchange requires thousands of multiplications and reductions. In the second case, we are simply presented with an ideal to reduce without any extra knowledge about its composition.

Key words and phrases: Ideal reduction, quadratic field, cryptographic public key-exchange, simple continued fraction.
ISSN 1814-0424 © 2006, http://ijmcs.future-in-tech.net
The authors' research was supported by grants from NSERC of Canada and iCORE of Alberta

Many reduction algorithms have been proposed [3, 6, 11, 14, 15], some in the closely related setting of binary quadratic forms. In this paper, we give an extensive survey and comparison of reduction in real quadratic fields. All the algorithms in this paper, including those originally described in the language of forms, are presented in a unified setting using the language of ideals. In addition, our survey and analysis of existing reduction algorithms has led to the development of a new algorithm based on the ideas of several of the techniques presented in this work. This new reduction algorithm is the fastest method for reducing ideals in real quadratic fields when multiplication is not required.

Our focus in this paper is on the problem of ideal reduction in real quadratic fields. As there is not a unique reduced ideal per equivalence class in real quadratic fields but rather a cycle of them, some applications such as cryptographic key exchange require the relative generator of the ideal relating the unreduced ideal to an equivalent reduced ideal. Thus, all the algorithms presented include the necessary formulas for computing the relative generator.

In order to determine which algorithms work better in practice, a careful implementation of the algorithms was written in the C programming language utilizing the GNU multi-precision library. The calculations were precisely optimized to ensure that any duplication of computing effort was eliminated. All of the algorithms were tested on a common hardware and software platform to ensure fair results. Each algorithm was carefully implemented using the most efficient known open source software, namely, the GNU C programming language and the GNU multiprecision library GMP. This is the first implementation of many of these algorithms and the most extensive comparison to date of reduction algorithms for ideals of real quadratic fields.

Summarizing the results, we found that for practical purposes NUCOMP is the best algorithm to use when multiplication and reduction are both required. Through the work of Shanks, Atkin, van der Poorten, Jacobson, Scheidler and Williams we have the ability to compute a reduced product without actually computing with any parameters near the size of the unreduced product. If reduction without multiplication is required, the new reduction algorithm presented here is the fastest.

2 Background and Relative Generator Calculation

The following background theory is well-known. Example sources are, [5] for abstract algebra, [16] for algebraic number theory and [17] for ideals of orders of quadratic fields. In this section we also determine the equivalence class of an ideal and provide a method of calculating the relative generator; that is, the principal ideal relating two equivalent ideals. As well, a method is introduced which converts algorithms for working with forms to those for ideals.

Definition 2.1. Let $D \in \mathbb{Z}$ be square-free and let $K = \mathbb{Q}(\sqrt{D})$. If $D > 0$, we say K is a *real quadratic field*; if $D < 0$, we say K is an *imaginary quadratic*

field.

Theorem 2.2. *The ring of integers of a quadratic field is given by* $\mathcal{O} = \mathbb{Z}[\omega]$ *where* $\omega = (\sigma - 1 + \sqrt{D})/\sigma$, $\sigma = 1$ *if* $D \equiv 2, 3 \pmod 4$, *and* $\sigma = 2$ *if* $D \equiv 1 \pmod 4$.

We denote the conjugate of an element α by $\bar{\alpha}$. The discriminant of $\mathbb{Q}(\sqrt{D})$ is calculated to be

$$\Delta = \begin{vmatrix} 1 & \omega \\ 1 & \bar{\omega} \end{vmatrix}^2 = \left(\frac{\sigma - 1 - \sqrt{D}}{\sigma} - \frac{\sigma - 1 + \sqrt{D}}{\sigma} \right)^2 = \left(\frac{2}{\sigma} \right)^2 D \ .$$

Each non-zero ideal $\mathfrak{a}$ of the ring of integers of a quadratic field may be written uniquely as the $\mathbb{Z}$ module

$$\mathfrak{a} = a\mathbb{Z} + (b + c\omega)\mathbb{Z}$$

where $a, b, c \in \mathbb{Z}$, $a, c > 0$, $0 \leq b < a$, $c|a$, $c|b$, $ac|\mathrm{N}(\,(b + c\omega)\,)$. Setting $Q = a\sigma/c$, $P = b\sigma/c + \sigma - 1$, $S = c$ we see that this is the same as

$$\mathfrak{a} = S\frac{Q}{\sigma}\mathbb{Z} + S\left(\frac{P + \sqrt{D}}{\sigma} \right)\mathbb{Z}$$

where Q/σ, P, $S \in \mathbb{Z}$, $\sigma Q | P^2 - D$ and $Q, P > 0$. Q/σ is the least positive rational integer in $\mathfrak{a}$; the norm of $\mathfrak{a}$ is easily calculated as SQ/σ. Since the only variables of ideals in this representation are Q, P and S, we use the notation $(S)(Q, P)$ to represent the ideal.

Definition 2.3. An ideal is *primitive* if it has no rational prime divisors.

We will usually work with ideals where $S = 1$. In this case, by the structure of $\mathcal{O}$ and the fact that Q/σ is the least element in $\mathfrak{a}$, the ideal is primitive, and we simply write (Q, P).

Since we are restricting our focus to quadratic fields, the task of computing the product of two ideals is quite simple. An algorithm of Shanks is described in [2] that gives the following method of computing the product of two primitive ideals of the same discriminant. To multiply $\mathfrak{a} = (Q_a, P_a)$ and $\mathfrak{b} = (Q_b, P_b)$ resulting in the product $(S)(Q_0, P_0)$, compute

$$\begin{aligned} &\sigma G = \gcd(Q_a, Q_b) = Q_a X + Q_b Y \ , \\ &\sigma S = \gcd(G, P_a + P_b) = GZ + (P_a + P_b)W \ , \\ &U \equiv (P_b - P_a)XZ - R_a W \pmod{Q_b/S} \ , \\ &Q_0 = \frac{Q_a Q_b}{\sigma S^2} \ , \\ &P_0 = \frac{Q_a U}{\sigma S} + P_a \ . \end{aligned} \tag{2.1}$$

Note. Throughout this paper, if the symbol R is used in the context of an ideal (Q, P), it always refers to the integer $(P^2 - D)/Q$. In cases where it is subscripted, the same subscripts apply to Q and P.

We need to compute the above greatest common divisor (GCD) because even though we are multiplying two primitive ideals, their product need not be primitive. The above calculations ensure that we have $\mathfrak{ab} = (S)(Q_0, P_0)$ with (Q_0, P_0) primitive.

Definition 2.4. An ideal $\mathfrak{a} = (Q, P)$ of $\mathcal{O}$ is *reduced* if $\mathfrak{a}$ is primitive and there does not exist an element $\beta \neq 0 \in \mathfrak{a}$ such that $|\beta| < \mathrm{N}(\mathfrak{a})$ and $|\bar{\beta}| < \mathrm{N}(\mathfrak{a})$. For ideals of positive discriminant where $\alpha = (P + \sqrt{D})/Q$, this is equivalent to $\alpha > 1$ and $-1 < \bar{\alpha} < 0$. This is the case if $0 < P < \sqrt{D},\ 0 < Q < P + \sqrt{D}$ and $Q + P > \sqrt{D}$.

It is clear from the multiplication formula of Equation (2.1) that the magnitude of the parameters Q_0 and P_0 of the product ideal is $O(Q_a Q_b)$. When (Q_a, P_a) and (Q_b, P_b) are reduced, the magnitude of every ideal parameter is bounded by $\sqrt{D}$; hence, $Q_a Q_b$ is bounded by D. Informally, reduction is the process of finding an equivalent ideal with parameters bounded in magnitude by $\sqrt{D}$, thereby ensuring that computations involving ideal arithmetic are carried out with coefficients of bounded size.

Definition 2.5. We define $\mathfrak{a}$ to be a *fractional ideal* of $\mathcal{O}$ if $\mathfrak{a} = c^{-1}\mathfrak{b}$ for some (ordinary) ideal $\mathfrak{b}$ and non-zero $c \in \mathcal{O}$. To avoid confusion, an ideal that is not a fractional ideal is sometimes called an *integral ideal*. Note that every integral ideal is a fractional ideal.

Definition 2.6. Two fractional ideals $\mathfrak{a}$ and $\mathfrak{b}$ are said to be *equivalent* (written $\mathfrak{a} \sim \mathfrak{b}$) if $\left(\frac{\psi_1}{\psi_2}\right)\mathfrak{a} = \mathfrak{b}$ where $\psi_1, \psi_2 \in \mathcal{O}$, $\psi_2 \neq 0$.

Definition 2.7. Let $\mathfrak{a}$ and $\mathfrak{b}$ be equivalent ideals. The term *relative generator* refers to the number ψ generating a principal fractional ideal such that $(\psi)\mathfrak{a} = \mathfrak{b}$.

From this point forward, we shall consider all ideals to be integral with the exception of principal ideals given by a relative generator. We now derive formulas for computing Q_j, P_j, and R_j which, in subsequent sections, will allow us to perform many reduction steps without needing to calculate these parameters at each step. This will allow us to skip some calculations and create more efficient algorithms.

Definition 2.8. By $\mathrm{GL}(2, \mathbb{Z})$ we denote the *general linear* group of 2×2 integral matrices which are invertible. Over $\mathbb{Z}$ this simply means that they have determinant ± 1. $\mathrm{PGL}(2, \mathbb{Z})$ is the factor group $\mathrm{GL}(2, \mathbb{Z})/\pm I_2$. That is, $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} -a & -b \\ -c & -d \end{smallmatrix}\right)$ are equivalent in $\mathrm{PGL}(2, \mathbb{Z})$.

Let $\mathfrak{a}_i = (Q_i, P_i)$ and define α_i to be the corresponding quadratic irrational formed by the quotient of $\mathfrak{a}_i$'s two generators.

$$\alpha_i = \frac{(P_i + \sqrt{D})/\sigma}{Q_i/\sigma} = \frac{P_i + \sqrt{D}}{Q_i} \quad .$$

Then by [8, Prop. 3], two ideals $\mathfrak{a}_i$ and $\mathfrak{a}_j$ are equivalent if and only if there exists a matrix $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{PGL}(2,\mathbb{Z})$ such that $\alpha_j = \frac{a\alpha_i+b}{c\alpha_i+d}$. Further, the numbers $\psi \neq 0 \in K$ such that $\mathfrak{a}_j = \psi\mathfrak{a}_i$ are given by $\psi = \frac{Q_j}{Q_i}\frac{1}{c\alpha_i+d} = \varepsilon(c\bar{\alpha}_i + d)$ where $\varepsilon = ad - bc$.

Observation. If $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ is as above then $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)^{-1} = \varepsilon \left(\begin{smallmatrix} d & -b \\ -c & a \end{smallmatrix}\right)$ satisfies $\alpha_i = \frac{d\alpha_j-b}{-c\alpha_j+a}$ and $\psi^{-1} = \frac{Q_i}{Q_j}\frac{1}{\varepsilon(-c\alpha_j+a)} = a - c\bar{\alpha}_j$.

We may use this information to determine the equivalence class of a given ideal.

Proposition 2.9. *The equivalence class of an ideal (Q_i, P_i) is given by*

$$[(Q_i,P_i)] = \left\{ (Q_j,P_j) \;\middle|\; \begin{array}{l} Q_j = \varepsilon(d^2Q_i + 2cdP_i + c^2R_i), \\ P_j = \varepsilon(bdQ_i + (ad+bc)P_i + acR_i), \\ \varepsilon = ad - bc = \pm 1,\ a,b,c,d \in \mathbb{Z} \end{array} \right\} . \tag{2.2}$$

Proof. (Q_i, P_i) is equivalent to (Q_j, P_j) if, and only if, $\alpha_j = \frac{a\alpha_i+b}{c\alpha_i+d}$ for some matrix $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{PGL}(2,\mathbb{Z})$. Substituting $\alpha_i = \frac{P_i+\sqrt{D}}{Q_i}$ and simplifying yields the formulas for Q_j and P_j. □

To summarize, if $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{PGL}(2,\mathbb{Z})$ is such that $\alpha_j = \frac{a\alpha_i+b}{c\alpha_i+d}$, then

$$\begin{aligned} &(Q_i,P_i) \sim (Q_j,P_j)\ , \\ &(\psi)(Q_i,P_i) = (Q_j,P_j) \text{ where } \psi = \varepsilon(c\bar{\alpha}_i + d)\ , &(2.3)\\ &(Q_i,P_i) = (\psi^{-1})(Q_j,P_j) \text{ where } \psi^{-1} = a - c\bar{\alpha}_j\ . &(2.4) \end{aligned}$$

It will be convenient to simplify the calculation of Q_j, P_j, and R_j, given (Q_i, P_i) and γ, and generalize [6, Lemma 6.3] by defining

$$\begin{aligned} F &= dP_i + cR_i\ , & G &= dQ_i + cP_i\ , \\ F' &= bP_i + aR_i\ , & G' &= bQ_i + aP_i\ . \end{aligned}$$

Then,

$$\begin{aligned} Q_j &= \varepsilon(d^2Q_i + 2cdP_i + c^2R_i) & P_j &= \varepsilon(bdQ_i + (ad+bc)P_i + acR_i) \\ &= \varepsilon(cF + dG)\ , & &= \varepsilon(aF + bG)\ , \\ R_j &= \varepsilon(b^2Q_i + 2abP_i + a^2R_i) \\ &= \varepsilon(aF' + bG')\ . \end{aligned} \tag{2.5}$$

Q_j and P_j can be given expressly in terms of Q_i and P_i by making the

appropriate substitution for R_i:

$$\begin{aligned} Q_j &= \varepsilon\left(d^2Q_i + 2cdP_i + c^2\frac{P_i^2 - D}{Q_i}\right) \\ &= \varepsilon\frac{d^2Q_i^2 + 2cdQP_i + c^2P_i^2 - c^2D}{Q_i} \\ &= \varepsilon\frac{G^2 - c^2D}{Q_i} \ , \end{aligned} \tag{2.6}$$

$$\begin{aligned} P_j &= \varepsilon\left(bdQ_i + (ad+bc)P_i + ac\frac{P_i^2 - D}{Q_i}\right) \\ &= \varepsilon\frac{bdQ_i^2 + (ad+bc)QP_i + acP_i^2 - acD}{Q_i} \\ &= \varepsilon\frac{GG' - acD}{Q_i} \ . \end{aligned} \tag{2.7}$$

Alternatively, if Q_j has been computed, a computationally more efficient formula for P_j is

$$\begin{aligned} P_j &= \varepsilon(aF + bG) = \varepsilon\left(aF + \frac{ad-\varepsilon}{c}G\right) \\ &= \varepsilon\frac{a(cF+dG) - \varepsilon G}{c} = \frac{a\varepsilon(cF+dG) - \varepsilon^2 G}{c} \\ &= \frac{aQ_j - G}{c} \ . \end{aligned} \tag{2.8}$$

2.1 Relative Generator Computation For Any Reduction Algorithm

Given any reduction algorithm, we may compute the relative generator relating the unreduced and reduced ideals. To do this, let (Q_0, P_0) be the input to the algorithm and compute $R_0 = (P_0^2 - D)/Q_0$. Run through one reduction step in the algorithm yielding Q_1, P_1, and R_1. In Proposition 2.9, R_j is determined symmetrically to Q_j and so we solve the following system of four equations in the four unknowns a_0, b_0, c_0, and d_0.

$$\begin{aligned} Q_1 &= \varepsilon_0(d_0^2Q_0 + 2c_0d_0P_0 + c_0^2R_0) \\ P_1 &= \varepsilon_0(b_0d_0Q_0 + (a_0d_0 + b_0c_0)P_0 + a_0c_0R_0) \\ R_1 &= \varepsilon_0(b_0^2Q_0 + 2a_0b_0P_0 + a_0^2R_0) \\ \varepsilon_0 &= a_0d_0 - b_0c_0 = \pm 1, \ a_0, b_0, c_0, d_0 \in \mathbb{Z} \end{aligned}$$

This procedure is repeated with Q_2, P_2, and R_2, and so on until the ideal is reduced. Some of the unknowns may themselves be variables; however, as long as the same procedure is used in each step, the pattern for the variables a_i, b_i,

c_i, and d_i will remain the same. Suppose the ideal is reduced after j reduction steps, then the relative generator may be found by first multiplying

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a_{j-1} & b_{j-1} \\ c_{j-1} & d_{j-1} \end{pmatrix} \begin{pmatrix} a_{j-2} & b_{j-2} \\ c_{j-2} & d_{j-2} \end{pmatrix} \cdots \begin{pmatrix} a_0 & b_0 \\ c_0 & d_0 \end{pmatrix}$$

to yield a matrix M which encapsulates the entire reduction. With this matrix, Equation (2.3) is used to compute the relative generator.

2.2 Classical Reduction of Ideals (Lagrange)

To illustrate how one may use these techniques to compute the relative generator for any reduction algorithm we look at the case of Lagrange's reduction method utilizing simple continued fractions. Lagrange reduction of an ideal (Q_0, P_0) of positive discriminant is the process of developing the simple continued fraction expansion of $(P_0 + \sqrt{D})/Q_0$ [17]. To develop the simple continued fraction expansion of a real number α, we let $\alpha_0 = \alpha$, $q_i = \lfloor \alpha_i \rfloor$, and $\alpha_{i+1} = 1/(\alpha_i - q_i)$. Note that $\alpha_{i+1} > 1$ for all $i \geq 0$. Using these definitions, α may be written as

$$\alpha = q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{\ddots q_{n-1} + \cfrac{1}{\alpha_n}}}} \ .$$

We write the above compactly as

$$\alpha = [q_0, q_1, q_2, \ldots, q_{n-1}, \alpha_n] \ .$$

The variables q_i are termed *partial quotients*. If we define the sequences

$$\begin{array}{llll} A_{-2} = 0, & A_{-1} = 1, & A_i = q_i A_{i-1} + A_{i-2} \ , \\ B_{-2} = 1, & B_{-1} = 0, & B_i = q_i B_{i-1} + B_{i-2} \ , \end{array} \tag{2.9}$$

then the rational numbers $A_i/B_i = [q_0, q_1, \ldots, q_i]$ are termed the *convergents* of the simple continued fraction expansion of α. If α is rational then $\alpha = [q_0, q_1, \ldots, q_n]$ for some $n \in \mathbb{N}$. If α is irrational, then we have an infinite continued fraction and $\lim_{i \to \infty} A_i/B_i = \alpha$. For all i we know

$$A_{i-1}B_{i-2} - A_{i-2}B_{i-1} = (-1)^i \ . \tag{2.10}$$

With this continued fraction notation, the sequence of ideals $\{\mathfrak{a}_i = (Q_i, P_i)\}$ is generated by

$$\frac{P_0 + \sqrt{D}}{Q_0} = \left[q_0, q_1, \ldots, q_{i-1}, \frac{P_i + \sqrt{D}}{Q_i} \right]$$

Algorithm 2.1 Rho(ρ) — one continued fraction expansion step

Input: (Q_i, P_i), D
Output: $\rho(Q_i, P_i) = (Q_{i+1}, P_{i+1})$

$$q_i \leftarrow \left\lfloor \frac{P_i + \sqrt{D}}{Q_i} \right\rfloor$$
$$P_{i+1} \leftarrow q_i Q_i - P_i$$
$$Q_{i+1} \leftarrow \frac{D - P_{i+1}^2}{Q_i} = q_i(P_i - P_{i+1}) - R_i$$

return (Q_{i+1}, P_{i+1})

where the values q_i, Q_i, P_i are obtained from Algorithm 2.1 below.

Since $R_{i+1} = (P_{i+1}^2 - D)/Q_{i+1}$, we have $R_{i+1} = -Q_i$. Tenner [4] avoids the division in the formula for Q_{i+1} in Algorithm 2.1 by substituting $P_{i+1} = q_i Q_i - P_i$ into the formula for Q_{i+1} as shown. Matching up coefficients with the equivalence class given in Equation (2.2), we determine that $\left(\begin{smallmatrix} a_i & b_i \\ c_i & d_i \end{smallmatrix}\right) = \left(\begin{smallmatrix} 0 & -1 \\ -1 & q_i \end{smallmatrix}\right)$ and $\varepsilon_i = -1$. Now note that

$$\begin{pmatrix} 0 & -1 \\ -1 & q_{i-1} \end{pmatrix} \begin{pmatrix} 0 & -1 \\ -1 & q_{i-2} \end{pmatrix} \cdots \begin{pmatrix} 0 & -1 \\ -1 & q_0 \end{pmatrix} = \begin{pmatrix} B_{i-2} & -A_{i-2} \\ -B_{i-1} & A_{i-1} \end{pmatrix} , \qquad (2.11)$$

Hence, by Equation (2.3), $(\psi_i)\mathfrak{a}_0 = \mathfrak{a}_i$ where $\psi_i = \varepsilon(A_{i-1} - B_{i-1}\frac{P_0 - \sqrt{D}}{Q_0})$. Due to the reduction process, the magnitude of the parameters Q_i and P_i is less than that of the initial parameters Q_0 and P_0. Hence, it is computationally more efficient to compute $\psi_i^{-1} = B_{i-2} + B_{i-1}\frac{P_i - \sqrt{D}}{Q_i}$. Note also that this computation does not require the $\{A_i\}$ sequence but only the $\{B_i\}$ sequence.

By the previous discussion of continued fractions, $\alpha_i = \frac{P_i + \sqrt{D}}{Q_i} > 1$ for $i > 0$. This is the first condition of the reduction criterion and guarantees that $Q_i > 0$ if $|P_i| < \sqrt{D}$, and P_i and Q_i have the same sign if $|P_i| > \sqrt{D}$. From the latter fact we may conclude that if $Q_i > 0$ and $P_i < \sqrt{D}$, then $|P_i| < \sqrt{D}$. Under these conditions, $\bar{\alpha}_i < 0$; hence, by Theorem 4.2 of [17], (Q_i, P_i) is reduced. This provides a simplified reduction criterion for use with this method of reduction. We emphasize that this criterion may only be used if $\alpha_i > 1$ which is only generally guaranteed for $i > 0$. One further observation we may make from the formula for Q_{i+1} is that the sign of Q_{i+1} is the opposite of Q_i when $|P_{i+1}| > \sqrt{D}$ and remains the same if $|P_{i+1}| < \sqrt{D}$.

We are now in a position to present Algorithm 2.2 which is our implementation of Lagrange reduction. The computations for Q_{i+1} and P_{i+1} utilize efficiencies discovered by Tenner. The formula for Q_{i+1} was discussed previously and P_{i+1} is obtained by solving the remainder of $\frac{P_i + \lfloor \sqrt{D} \rfloor}{Q_i}$, namely $r_i = P_i + \left\lfloor \sqrt{D} \right\rfloor - q_i Q_i$, for $q_i Q_i - P_i$.

We will see in later sections that many of the faster reduction algorithms do

Algorithm 2.2 Lagrange Reduction

Input: $\mathfrak{a}_0 = (Q_0, P_0)$, D
Output: $\mathfrak{a}_i = (Q_i, P_i)$, Ψ_i, such that $\mathfrak{a}_i = (\Psi_i)\mathfrak{a}_0$ and $\mathfrak{a}_i$ is reduced
Note: The $\{A_i\}$ do not need to be calculated unless they are required by an external algorithm.

[Initialization]
$i \leftarrow 0$
$A_{-2} \leftarrow 0$, $A_{-1} \leftarrow 1$, $B_{-2} \leftarrow 1$, $B_{-1} \leftarrow 0$
$R_0 \leftarrow \dfrac{P_0^2 - D}{Q_0}$
[Reduction]
while (Q_i, P_i) is not reduced **do**
  $q_i \leftarrow \left\lfloor \dfrac{P_i + \lfloor\sqrt{D}\rfloor}{Q_i} \right\rfloor$, $r_i \leftarrow P_i + \left\lfloor \sqrt{D} \right\rfloor - q_i Q_i$ (simultaneously)
  $R_{i+1} \leftarrow -Q_i$
  $P_{i+1} \leftarrow \left\lfloor \sqrt{D} \right\rfloor - r_i$
  $Q_{i+1} \leftarrow q_i(P_i - P_{i+1}) - R_i$
  $A_i \leftarrow q_i A_{i-1} + A_{i-2}$, $B_i \leftarrow q_i B_{i-1} + B_{i-2}$
  $i \leftarrow i + 1$
end while
[Compute relative generator]
$\Psi_i^{-1} \leftarrow B_{i-2} + B_{i-1}\left(\dfrac{P_i - \sqrt{D}}{Q_i}\right)$
return (Q_i, P_i), Ψ_i

not completely reduce the ideal but come within a few Lagrange steps of being reduced. Therefore, the Lagrange algorithm is often used after the completion of other reduction algorithms to finish the reduction. The following lemma is an important technical result that simplifies the computation of the relative generator in these cases. To simplify the proof of the lemma we first define the following function.

Definition 2.10. The function λ_M, where $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{PGL}(2, \mathbb{Z})$, represents the action of the matrix M on the irrational number α_0 corresponding to an ideal (Q_0, P_0). This action produces the irrational number α_i corresponding to an ideal (Q_i, P_i).

$$\begin{aligned} \lambda_M : \quad & \mathbb{Q}(\sqrt{D}) \to \mathbb{Q}(\sqrt{D}) \\ & \frac{P_0 + \sqrt{D}}{Q_0} \mapsto \frac{\varepsilon(bdQ_0 + (ad+bc)P_0 + acR_0) + \sqrt{D}}{\varepsilon(d^2Q_0 + 2cdP_0 + c^2R_0)} = \frac{P_i + \sqrt{D}}{Q_i} \end{aligned}$$

with $\varepsilon = \det M$.

The proof that $\Lambda = \{\lambda_M \mid M \in \mathrm{PGL}(2,\mathbb{Z})\}$ forms a group under function composition with identity element λ_I where $I = \pm\left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)$, is straight-forward. Composition in Λ is given by $\lambda_{M_1} \circ \lambda_{M_2} = \lambda_{M_2 M_1}$.

Lemma 2.11. *If* $(Q,P) = \left(a - c\left(\frac{P'-\sqrt{D}}{Q'}\right)\right)(Q',P')$, *then calling Algorithm 2.2 with* (Q',P') *and initializing* $B_{-2} \leftarrow a$, $B_{-1} \leftarrow -c$ *produces* (Q'',P'') *and* Ψ *such that* $(Q'',P'') = (\Psi)(Q,P)$.

Proof. Let α, α', and α'' be irrational numbers corresponding to the ideals (Q,P), (Q',P'), and (Q'',P'') respectively. Let $M' = \left(\begin{smallmatrix}a & b\\ c & d\end{smallmatrix}\right)$ be such that $\alpha' = \lambda_{M'}(\alpha)$ and let $\alpha'' = \lambda_{M''}(\alpha')$ where $M'' = \left(\begin{smallmatrix}B_{i-2} & -A_{i-2}\\ -B_{i-1} & A_{i-1}\end{smallmatrix}\right)$ (recall Equation (2.11)) and the matrix entries are obtained from Algorithm 2.2 on input of (Q',P'). This implies that

$$\begin{aligned}\alpha'' &= \lambda_{M''}(\lambda_{M'}(\alpha))\\ &= \lambda_{M'} \circ \lambda_{M''}(\alpha)\\ &= \lambda_{M''M'}(\alpha)\ .\end{aligned}$$

M'' is developed by successively left multiplying each $\left(\begin{smallmatrix}0 & -1\\ -1 & q_i\end{smallmatrix}\right)$ in turn. The starting matrix is $\left(\begin{smallmatrix}B_{-2} & -A_{-2}\\ -B_{-1} & A_{-1}\end{smallmatrix}\right) = \left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)$ which is the identity matrix. Thus we have

$$\begin{aligned}M''M' &= \begin{pmatrix}B_{i-2} & -A_{i-2}\\ -B_{i-1} & A_{i-1}\end{pmatrix}\begin{pmatrix}a & b\\ c & d\end{pmatrix}\\ &= \begin{pmatrix}B_{i-2} & -A_{i-2}\\ -B_{i-1} & A_{i-1}\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}a & b\\ c & d\end{pmatrix}\\ &= \begin{pmatrix}B_{i-2} & -A_{i-2}\\ -B_{i-1} & A_{i-1}\end{pmatrix}\begin{pmatrix}B_{-2} & -A_{-2}\\ -B_{-1} & A_{-1}\end{pmatrix}\begin{pmatrix}a & b\\ c & d\end{pmatrix}\ .\end{aligned}$$

Hence, if in Algorithm 2.2 we set $B_{-2} \leftarrow a$, $B_{-1} \leftarrow -c$, $A_{-2} \leftarrow -b$, and $A_{-1} \leftarrow d$, then $M''M'$ is the resulting $\begin{pmatrix}B_{i-2} & -A_{i-2}\\ -B_{i-1} & A_{i-1}\end{pmatrix}$ from which the relative generator is calculated giving $(\Psi)(Q'',P'') = (Q,P)$. If only the $\{B_i\}$ sequence is required, then the values a and c from $(Q,P) = \left(a - c\left(\frac{P'-\sqrt{D}}{Q'}\right)\right)(Q',P')$ are obtained from the first column of the matrix M'; therefore, all that is required is to set $B_{-2} \leftarrow a$ and $B_{-1} \leftarrow -c$ in which case the relative generator returned satisfies $(Q'',P'') = (\Psi)(Q,P)$. □

2.3 Binary Quadratic Forms

Ideals of quadratic fields and binary quadratic forms are closely related objects. Most of the algorithms presented in this paper were originally presented in the language of binary quadratic forms. An in-depth discussion and proof of the equivalence of ideals and classes of forms is readily available in works such as [2] or [3]. Following is the relationship we used to convert the algorithms for forms into algorithms for ideals.

Definition 2.12. A *binary quadratic form* is a function $f(x, y) = ax^2+bxy+cy^2$ with $a, b, c, x, y \in \mathbb{Z}$. It is written compactly as (a, b, c) and its *discriminant* is defined to be $\Delta = b^2 - 4ac$. Note that $\Delta \equiv 0, 1 \pmod 4$.

References such as [2] and [3] show that a form (a, b, c) can be put in correspondence with the $\mathbb{Z}$-module

$$a\mathbb{Z} + \left(\frac{-b + \sqrt{\Delta}}{2}\right)\mathbb{Z} \ .$$

This is not a one-to-one mapping as many distinct forms correspond to a single $\mathbb{Z}$-module. To convert algorithms for working with forms to those for ideals, we use the $\mathbb{Z}$-module from the above correspondence and set it equal to an arbitrary $\mathbb{Z}$-module that uses our representation of ideals:

$$a\mathbb{Z} + \frac{-b + \sqrt{\Delta}}{2}\mathbb{Z} = \frac{Q}{\sigma}\mathbb{Z} + \frac{P + \sqrt{D}}{\sigma}\mathbb{Z} \ .$$

Hence, one correspondence that will work is

$$a = \frac{Q}{\sigma} \text{ and } \frac{-b + \sqrt{\Delta}}{2} = \frac{P + \sqrt{D}}{\sigma} \ .$$

Equating the rational parts of the right hand side equation,

$$b = -\frac{2P}{\sigma} \ .$$

We do not have (nor do we require) a one-to-one mapping between ideals and forms but this does give us a correspondence so that we may translate the algorithms. In fact, for the algorithms, we may further simplify the correspondence to $a = Q/\sigma$ and $b = 2P/\sigma$. A few more calculations confirm that $c = R/\sigma$.

2.4 Gaussian Reduction

Gauss's reduction method has also been adapted to ideals of real quadratic fields. We do not present the details because the method is considerably slower than the methods presented. The interested reader may refer to [13] for the converted algorithm and timing comparisons.

3 Modern Reduction Algorithms

3.1 Rickert

In [9], Lehmer describes a method of speeding up the Euclidean Algorithm by working with single-precision numbers in some calculations rather than multi-precision numbers. If we are given numbers a and b (without loss of generality,

$a > 0$) and want to find q and r such that $b = aq + r,\ 0 \leq r < a$, we generally set $q = \lfloor b/a \rfloor$ and $r = b - aq$. Lehmer noticed that usually all that is required to compute q are the leading digits of a and b — the tails often have no effect on the result. Therefore, we should approximate a and b by taking only the leading (word size) bits of each, say $\hat{a}$ and $\hat{b}$, respectively.[1] Now, set $q = \left\lfloor \hat{b}/\hat{a} \right\rfloor$ and $r = b - aq$.

This is a simplification of the original algorithm. In his paper, Lehmer generates new approximations based on the initial values for $\hat{a}$ and $\hat{b}$ until he knows the q values they yield are no longer correct. Then a multi-precision catch-up step is performed and new approximations are taken. We present the full algorithm as Algorithm 3.1 since we use it extensively, and the extended GCD version is not readily available.

Rickert used this idea in [11] to speed up the reduction algorithm for binary quadratic forms of negative discriminant. Let k be the maximum length of the numbers a, b and c minus the size of a computer word. Then set $\hat{a} = \left\lfloor a/2^k \right\rfloor$, $\hat{b} = \left\lfloor b/2^k \right\rfloor$, $\hat{c} = \left\lfloor c/2^k \right\rfloor$. Reduce the form $(\hat{a}, \hat{b}, \hat{c})$ using standard Gaussian reduction and multiply the unimodular matrices used in that reduction to give a single transformation matrix which is then applied to the full form (a, b, c). Continue this procedure until (a, b, c) is reduced. The advantage is that most of the operations and reduction calculations are performed on single precision numbers with comparatively few operations needing to be computed on the full multi-precision form.

We apply this technique to an ideal (Q_i, P_i) by taking single precision approximations $\widehat{Q}_i$, $\widehat{P}_i$ and $\widehat{R}_i$ of Q_i, P_i and R_i; then, let $\widehat{D}_i = \widehat{P}_i^2 - \widehat{Q}_i\widehat{R}_i$ and reduce $(\widehat{Q}_i, \widehat{P}_i)$ in $\mathbb{Q}\left(\sqrt{\widehat{D}_i}\right)$.

Conceptually, if $\widehat{D}_i > 0$ and Lagrange reduction (Algorithm 2.2) is used to reduce $(\widehat{Q}_i, \widehat{P}_i)$, this procedure embeds the partial quotients $[\tilde{q}_0, \tilde{q}_1, \ldots, \tilde{q}_j]$ so that

$$\frac{P_i + \sqrt{D}}{Q_i} = \left[\tilde{q}_0, \tilde{q}_1, \ldots, \tilde{q}_j, \frac{P_{i+1} + \sqrt{D}}{Q_{i+1}}\right] .$$

Using the technique discussed in the previous section, we easily determine the formulas for the multiprecision catchup step that enable us to move from (Q_i, P_i) to (Q_{i+1}, P_{i+1}). This single catchup step saves us from computing j multiprecision iterations of Algorithm 2.2.

Even if $D_i > 0$, $\widehat{D}_i$ need not be positive if a naive approach to computing $\widehat{D}_i$ is used. For a simple example, consider $Q_i = 28$, $P_i = 23$, $R_i = 17$ and $D_i = 53$. If $k = 2$ then $\widehat{Q}_i = \left\lfloor 28/2^2 \right\rfloor = 7$, $\widehat{P}_i = \left\lfloor 23/2^2 \right\rfloor = 5$, $\widehat{R}_i = \left\lfloor 17/2^2 \right\rfloor = 4$ and $\widehat{D}_i = 5^2 - (7)(4) = -3$. Either the algorithm used in the single-precision reductions must take this into account or a different computation for $\widehat{D}_i$ must be used.

[1] This assumes they were the same length to start with. If not, the shorter number is padded in front with zeros.

Algorithm 3.1 Lehmer's Extended GCD (with Jebelean's condition [7])

Input: Non-negative integers C_{-2} and C_{-1} with $C_{-2} \geq C_{-1}$
Output: A_{j-1}, B_{j-1}, G such that $A_{j-1}C_{-1} + B_{j-1}C_{-2} = G = \gcd(C_{-2}, C_{-1})$
Note: The variables A_j and B_j correspond in absolute value to the standard sequence from the continued fraction expansion of C_{-2}/C_{-1}.

[Initialization]
$A_{-2} \leftarrow 0,\ A_{-1} \leftarrow 1,\ B_{-2} \leftarrow 1,\ B_{-1} \leftarrow 0$
$i \leftarrow 0,\ j \leftarrow 0$

[Main loop]
while $C_{j-1} > 0$ **do**
  $k \leftarrow \max(\lfloor \log_2 C_{j-2} \rfloor + 1 - (\text{word size}), 0)$
  $r_{-2} \leftarrow \lfloor C_{j-2}/2^k \rfloor,\ r_{-1} \leftarrow \lfloor C_{j-1}/2^k \rfloor$
  $a_{-2} \leftarrow 0,\ a_{-1} \leftarrow 1,\ b_{-2} \leftarrow 1,\ b_{-1} \leftarrow 0$

  [Euclidean step]
  while $r_{i-1} > 0$ **do** {Exit is usually at Jebelean's condition}
    $q_i \leftarrow \lfloor r_{i-2}/r_{i-1} \rfloor,\ r_i \leftarrow r_{i-2} - q_i r_{i-1}$
    $a_i \leftarrow a_{i-2} - q_i a_{i-1},\ b_i \leftarrow b_{i-2} - q_i b_{i-1}$

    [Jebelean's condition]
    if i is even **then**
      if $(r_i < -a_i)$ **or** $(r_{i-1} - r_i < b_i - b_{i-1})$ **then break**
    else
      if $(r_i < -b_i)$ **or** $(r_{i-1} - r_i < a_i - a_{i-1})$ **then break**
    end if
    $i \leftarrow i + 1$
  end while

  [Multi-precision step]
  if $i = 0$ **then** {No correct single-precision steps}
    $Q \leftarrow \lfloor C_{j-2}/C_{j-1} \rfloor,\ C_j \leftarrow C_{j-2} - QC_{j-1}$
    $A_j \leftarrow A_{j-2} - QA_{j-1},\ B_j \leftarrow B_{j-2} - QB_{j-1}$
  else
    $C_{j+i-1} \leftarrow a_{i-2}C_{j-1} + b_{i-2}C_{j-2},\ C_{j+i} \leftarrow a_{i-1}C_{j-1} + b_{i-1}C_{j-2}$
    $A_{j+i-1} \leftarrow a_{i-2}A_{j-1} + b_{i-2}A_{j-2},\ A_{j+i} \leftarrow a_{i-1}A_{j-1} + b_{i-1}A_{j-2}$
    $B_{j+i-1} \leftarrow a_{i-2}B_{j-1} + b_{i-2}B_{j-2},\ B_{j+i} \leftarrow a_{i-1}B_{j-1} + b_{i-1}B_{j-2}$
  end if
  $j \leftarrow j + i + 1,\ i \leftarrow -1$
end while

return $(A_{j-1}, B_{j-1}, C_{j-2})$

We can ensure $\widehat{D}_i$ is positive by setting

$$\widehat{Q}_i = (Q_i/|Q_i|) \left\lfloor |Q_i|/2^k \right\rfloor \ ,$$
$$\widehat{P}_i = (P_i/|P_i|) \left\lceil |P_i|/2^k \right\rceil \ ,$$
$$\widehat{R}_i = (R_i/|R_i|) \left\lfloor |R_i|/2^k \right\rfloor \ ;$$

then,

$$\widehat{P}_i^2 = \left\lceil \frac{|P_i|}{2^k} \right\rceil^2 \geq \left(\frac{P_i}{2^k}\right)^2 > \left(\frac{Q_i}{2^k}\right)\left(\frac{R_i}{2^k}\right) \geq \left(\frac{Q_i}{|Q_i|}\right)\left\lfloor \frac{|Q_i|}{2^k} \right\rfloor \left(\frac{R_i}{|R_i|}\right)\left\lfloor \frac{|R_i|}{2^k} \right\rfloor = \widehat{Q}_i\widehat{R}_i \ .$$

Here we used the fact that $P^2 - QR = D > 0 \ \Rightarrow \ P^2 > QR$. Since $\widehat{P}_i^2 > \widehat{Q}_i\widehat{R}_i$ we also have $\widehat{P}_i^2 - \widehat{Q}_i\widehat{R}_i = \widehat{D}_i > 0$. Another approach would be to take the floor in each case and if $\widehat{D}_i < 0$, simply set $\widehat{Q}_i \leftarrow -\widehat{Q}_i$ and recalculate $\widehat{D}_i$ which will now be positive.

When we compute the approximations $\widehat{Q}_i$, $\widehat{P}_i$ and $\widehat{R}_i$, if one of them is 0, or if $(\widehat{Q}_i, \widehat{P}_i)$ is a reduced ideal, a full multi-precision reduction step is taken on (Q_i, P_i) instead, and then the procedure is continued with approximations. We continue this process until a fully reduced ideal $\mathfrak{a}_r$ is obtained.

The implementation of Rickert's algorithm is given as Algorithm 3.2. We could have computed R_{i+2} as $(P_{i+2}^2 - D)/Q_{i+2}$ instead of the way it is presented; however, this would require a multi-precision squaring and a multi-precision division which are both costly, particularly the division. We instead trade this for three multiplications of a multi-precision number by a single-precision number.

The algorithm uses Lagrange reduction (Algorithm 2.2) for the single-precision reductions. There is a possibility that the single-precision parameters returned by Algorithm 2.2 will produce an overflow when they are later multiplied in Algorithm 3.2. For this reason, it is important to test that they remain less than (word size)/2 in length in Algorithm 2.2 and exit early if necessary.

3.2 Schönhage

The divide and conquer technique was used by Schönhage [15] to give an asymptotically fast method of reducing forms. We present the algorithm here in terms of ideals and show how to compute the relative generator.

To use divide and conquer, the problem of reducing an ideal must be somehow split into smaller subproblems. The idea Schönhage had was to find an ideal that is *minimal above* some value s.

Definition 3.1. (Q, P) is *minimal above* above $s > 0$ if

1. $Q, P, R \geq s$ and
2. (a) $Q - 2P + R < s$ or
   (b) $P - Q < s$ and $P - R < s$.

A similar proof to the one Schönhage provides in [15, Sec. 4] guarantees that an ideal minimal above 1 is at most one step from being reduced. Before looking into the details of this method, we present an overview of the algorithm. To satisfy the first condition of Definition 3.1 for any value of s, we need Q, P and R positive to start with; this is accomplished with Algorithm 3.4. We then find an ideal minimal above 1 recursively. Lastly, we do the final reduction step and terminate. The main components are combined as Algorithm 3.3. The algorithm returns a reduced ideal and relative generator.

Algorithm 3.2 Rickert–Style Reduction Algorithm

Input: (Q_0, P_0), D
Output: (Q_i, P_i), Ψ_i such that $(Q_i, P_i) = (\Psi_i)(Q_0, P_0)$

[Initialization]
$B_{-2} \leftarrow 1$, $B_{-1} \leftarrow 0$, $i \leftarrow 0$
$R_0 \leftarrow \dfrac{P_0^2 - D}{Q_0}$

[Reduction]
while $|Q_i| > \sqrt{D}$ **do**
  $k \leftarrow \max(\max(\lceil \log_2 Q_i \rceil, \lceil \log_2 P_i \rceil, \lceil \log_2 R_i \rceil) - (\text{word size}), 0)$
  $\widehat{Q}_i \leftarrow (Q_i/|Q_i|) \left\lfloor |Q_i|/2^k \right\rfloor$, $\widehat{P}_i \leftarrow (P_i/|P_i|) \left\lceil |P_i|/2^k \right\rceil$,
  $\widehat{R}_i \leftarrow (R_i/|R_i|) \left\lfloor |R_i|/2^k \right\rfloor$
  $\widehat{D}_i \leftarrow \widehat{P}_i^2 - \widehat{Q}_i \widehat{R}_i$
  if $\widehat{Q}_i$, $\widehat{P}_i$ or $\widehat{R}_i$ is 0 **or** $|\widehat{Q}_i| < \sqrt{\widehat{D}_i}$ **then**
    [Full multi-precision reduction]
    Compute q_i, r_i so that $P_i + \left\lfloor \sqrt{D} \right\rfloor = q_i Q_i + r_i \quad (0 \le r_i < |Q_i|)$
    $P_{i+1} \leftarrow \left\lfloor \sqrt{D} \right\rfloor - r_i$, $Q_{i+1} \leftarrow q_i(P_i - P_{i+1}) - R_i$, $R_{i+1} \leftarrow -Q_i$
    $B_{i-1} \leftarrow -B_{i-1}$, $B_i \leftarrow q_i B_{i-1} - B_{i-2}$, $i \leftarrow i + 1$
  else
    [Single-precision reduction]
    $a, a', b, b' \leftarrow$ Last two A_i and B_i from Algorithm 2.2 $((\widehat{Q}_i, \widehat{P}_i), \sqrt{\widehat{D}_i})$
    $\varepsilon \leftarrow ab' - ba'$ (Compute efficiently using Equation (2.10))
    $Q_{i+2} \leftarrow \varepsilon(a^2 Q_i - 2abP_i + b^2 R_i)$
    $P_{i+2} \leftarrow \varepsilon(-aa' Q_i + (ab' + a'b)P_i - bb' R_i)$
    $R_{i+2} \leftarrow \varepsilon(a'^2 Q_i - 2a'b' P_i + b'^2 R_i)$
    $B_i \leftarrow -a' B_{i-1} + b' B_{i-2}$
    $B_{i+1} \leftarrow aB_{i-1} - bB_{i-2}$
    $i \leftarrow i + 2$
  end if
end while

[Compute relative generator]
$\Psi_i^{-1} \leftarrow B_{i-2} + B_{i-1}\left(\dfrac{P_i - \sqrt{D}}{Q_i}\right)$
return (Q_i, P_i), Ψ_i

Make Q, P and R Positive

The first step in making Q, P and R positive is to ensure $Q+R > 0$. If $Q+R < 0$ then negate Q, P and R. This negation is justified by considering the equations

Algorithm 3.3 Schönhage Reduction

Input: $\mathfrak{a} = (Q, P)$, D
Output: $\mathfrak{a}' = (Q', P')$, Ψ such that $\mathfrak{a}' = (\Psi)\mathfrak{a}$ and $\mathfrak{a}'$ is reduced

[Initialization]
$R \leftarrow \dfrac{P^2 - D}{Q}$

[Make Positive]
$Q', P', R', a_1, c_1, \text{sign} \leftarrow$ Algorithm 3.4 (Q, P, R)

[Reduce Ideal Minimal Above 1]
$Q', P', R', \begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix} \leftarrow$ Algorithm 3.5 $(Q', P', R', -1)$
$a_3 \leftarrow a_1 a_2 + b_2 c_1$, $c_3 \leftarrow a_1 c_2 + c_1 d_2$
if sign $= -1$ **then**
  $Q' \leftarrow -Q'$, $P' \leftarrow -P'$, $R' \leftarrow -R'$
end if

[Final Lagrange Reduction Steps]
$(Q', P'), \Psi \leftarrow$ Algorithm 2.2 $((Q', P'), D)$ where $B_{-2} \leftarrow a_3$ and $B_{-1} \leftarrow -c_3$.

return $\mathfrak{a}'$, Ψ

in Proposition 2.9. If

$$\begin{aligned} Q_j &= \varepsilon(d^2 Q_i + 2cdP_i + c^2 R_i), \\ P_j &= \varepsilon(bdQ_i + (ad + bc)P_i + acR_i), \end{aligned}$$

then

$$\begin{aligned} -Q_j &= \varepsilon(d^2(-Q_i) + 2cd(-P_i) + c^2(-R_i)), \\ -P_j &= \varepsilon(bd(-Q_i) + (ad + bc)(-P_i) + ac(-R_i)) \ . \end{aligned}$$

Hence, the same steps that reduce $(-Q, -P)$ will also reduce (Q, P). In Algorithm 3.4 a sign value is returned which indicates if the parameters must be negated before terminating Algorithm 3.3.

Second, we want to ensure that $P > 0$. If $P < 0$, we use the values of the matrix $M = \left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right)$ in the above equations to see that $(Q, P) \sim (R, -P)$.

At this point we know that $Q + R > 0$ which guarantees that at most one of Q or R is negative. If $Q < 0$, then using the values in $\left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$ we set $Q \leftarrow Q + 2P + R$, $P \leftarrow P + R$, and leave R unchanged. Since $Q + R, P$, and R were positive, we know the new values for Q, P, and R are positive. A similar argument works with the values in $\left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$ if $R < 0$. The full algorithm to make the ideal parameters positive is given as Algorithm 3.4.

Algorithm 3.4 Make Positive

Input: Q, P, R where (Q, P) is an ideal.

Output: $Q', P', R', a, c, \text{sign}$, where $\left(a + c\left(\frac{P'-\sqrt{D}}{Q'}\right)\right)(Q', P') = (Q, P)$ and Q', P', R' are all positive.

[Initialization]
$Q' \leftarrow Q$, $P' \leftarrow P$, $R' \leftarrow R$
$a \leftarrow 1$, $c \leftarrow 0$ {Denotes a and c of Definition 2.10}

[Ensure $Q' + R' \geq 0$ and $P' \geq 0$]
if $(Q' + R' < 0)$ **then**
  $Q' \leftarrow -Q'$, $P' \leftarrow -P'$, $R' \leftarrow -R'$
  sign $\leftarrow -1$
else
  sign $\leftarrow 1$
end if
if $(P' < 0)$ **then**
  $Q' \leftarrow R'$, $P' \leftarrow -P'$, $R' \leftarrow Q'$
  $a \leftarrow 0$, $c \leftarrow -1$
end if

[Ensure Q', P', R' are all positive]
if $(Q' < 0)$ **then**
  $Q' \leftarrow Q' + 2P' + R'$, $P' \leftarrow P' + R'$, $R' \leftarrow R'$
  $a \leftarrow a$, $c \leftarrow a + c$
else if $(R' < 0)$ **then**
  $Q' \leftarrow Q'$, $P' \leftarrow Q' + P'$, $R' \leftarrow Q' + 2P' + R'$
  $a \leftarrow a + c$, $c \leftarrow c$
end if

return $Q', P', R', a, c, \text{sign}$

Monotone Reduction

The main problem we want to solve is to find an ideal minimal above $s = 1$.[2] To do this, we use the divide and conquer MR algorithm (Algorithm 3.5). There is a value for s that will only require several reduction steps to reduce the ideal minimal above s. We recursively call MR with larger values of s until this case is found. The reduction steps in each of these subproblems are accumulated by calculating the product of the matrices that effect the reductions in each recursive call. Often, only the leading bits of the ideal parameters are used in the calculations. The product is not only used to update the tail bits of ideal parameters but it also provides the information required to compute the relative

[2] In the implementations we take as input the parameter m instead of s; s and m are related by the equation $s = 2^m$.

generator.

Giving a bit more detail, suppose the MR algorithm is called with the parameters Q, P, R, s. If Q, P, R are all less than $4s$ then we do a few simple steps, which we will explain later, until (Q, P) is minimal above s; otherwise, we recurse by increasing the threshold by $\max(Q, 2P, R))/2$. That is, we choose a new value $s' = (s + \max(Q, 2P, R))/2$ and execute $Q', P', R', M_1 \leftarrow \mathrm{MR}(Q, P, R, s')$. This gives (Q', P') minimal above s'. Now, a few reduction steps are taken (we keep track of the relative generator in the matrix M_2) until Q', P' and R' are all less than s'. Then MR is called again with the original s but the new parameters corresponding to the partially reduced ideal. Thus, we execute $Q', P', R', M_3 \leftarrow \mathrm{MR}(Q', P', R', s)$. The values needed to compute the relative generator are obtained by multiplying $M_3 M_2 M_1$.

In the actual algorithm we often work with only the leading bits of Q, P and R to speed up calculations in a style similar to Lehmer's GCD algorithm. The full algorithm for monotone reduction is annotated and given as Algorithm 3.5. In the following section we explain how steps 5 and 9 use Algorithm 3.6 to reduce the ideal; these are the only steps in which the ideal is being reduced.

Algorithm 3.5 Monotone Reduction — MR

Input: Q, P, R, m where (Q, P) is an ideal.
Output: Q', P', R', M where M is a matrix tracking the relative generator and (Q', P') is minimal above 2^m

[Initialization]
$Q' \leftarrow Q$, $P' \leftarrow P$, $R' \leftarrow R$

[1. If sufficiently small in size, skip to end]
if $\min(Q', 2P', R') < 2^{m+2}$ **then**
  $M \leftarrow I_2$
else

  [2. Calculate size of operands and split if necessary]
  $n \leftarrow \max(\lfloor \log_2 Q' \rfloor, \lfloor \log_2 2P' \rfloor, \lfloor \log_2 R' \rfloor) - m + 1$
  if $m \leq n$ **then**
    $m' \leftarrow m$, $p \leftarrow 0$
  else
    $m' \leftarrow n$, $p \leftarrow m - n + 1$
    Split Q', P', R' as $Q' = Q_0 + 2^p Q_1 \quad (0 \leq Q_0 < 2^p)$
    $P' = P_0 + 2^p P_1 \quad (0 \leq P_0 < 2^p)$
    $R' = R_0 + 2^p R_1 \quad (0 \leq R_0 < 2^p)$
    $Q' \leftarrow Q_1$, $P' \leftarrow P_1$, $R' \leftarrow R_1$
  end if

  [3. Calculate half-way point]
  $h \leftarrow m' + \lfloor n/2 \rfloor$
  if $\min(Q', 2P', R') < 2^h$ **then**

$M \leftarrow I_2$
else
[4. Recursive call — reduce (Q, P) minimal above 2^h]
$Q', P', R', M \leftarrow \mathrm{MR}(Q', P', R', h)$
end if

[5. Reduce until $\max(Q', 2P', R') < 2^h$]
while $\max(Q', 2P', R') \geq 2^h$ **and** (Q', P') is **not** minimal above $2^{m'}$ **do**
$Q', P', R', M \leftarrow$ Algorithm 3.6 (Q', P', R', M, m')
end while
if (Q', P') is not minimal above $2^{m'}$ **then**

[6-7. Recursive call — reduce (Q', P') minimal above m']
$Q', P', R', M' \leftarrow \mathrm{MR}(Q', P', R', m')$
$M \leftarrow M'M$

[8. Update tails]
if $p > 0$ **then**
$F \leftarrow dP_0 + cR_0$, $G \leftarrow dQ_0 + cP_0$
$Q' \leftarrow 2^p Q' + cF + dG$
$P' \leftarrow 2^p P' + aF + bG$
$R' \leftarrow 2^p R' + b^2 Q_0 + 2abP_0 + a^2 R_0$
end if
end if
end if

[9. Reduce above m]
while (Q', P') is not minimal above 2^m **do**
$Q', P', R', M \leftarrow$ Algorithm 3.6 (Q', P', R', M, m')
end while

return Q', P', R', M

Simple Steps

Reduction in the MR algorithm is accomplished via the use of two basic steps. These reduction steps are designed so that the ideal parameters Q, P, and R remain positive. The first of these is called a *low step*. Here, Q remains unchanged and the reduction is

$$\begin{aligned} Q' &\leftarrow Q \ , \\ P' &\leftarrow P - Q \ , \\ R' &\leftarrow Q - 2P + R \ . \end{aligned} \tag{3.1}$$

The second is called a *high step* and it is symmetric, leaving R unchanged with the reduction

$$\begin{aligned} Q' &\leftarrow Q - 2P + R \ , \\ P' &\leftarrow P - R \ , \\ R' &\leftarrow R \ . \end{aligned}$$

The matrices that effect these reductions are,

$$L = \underset{\text{low step}}{\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}} \text{ and } H = \underset{\text{high step}}{\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}} \ .$$

The restriction that all of the ideal parameters remain positive provides a unique reduction path using the above steps. Observe that once it is not possible to take a low step or high step without one of Q, P or R becoming negative, (Q, P) is minimal above 1 and hence almost reduced. To prove these claims, observe that at each stage, an entry in the subsequent reduction is $Q - 2P + R$. If this is positive then

$$\begin{aligned} &(Q - P) + (R - P) > 0 \\ \Rightarrow\ &(P - Q) + (P - R) < 0 \ . \end{aligned}$$

This implies that at most one of $P - Q$ and $P - R$ can be positive. Choosing the positive course, we have a unique reduction path. The divide and conquer technique requires a generalization of this idea. Instead of simply requiring that Q, P, and R be positive, we require that they be greater than 2^m for some $m \in \mathbb{Z}$. As we will soon show, this criterion also provides a unique reduction path and we observe from Definition 3.1 that once it is not possible to take a low step or high step without one of Q, P or R becoming less than 2^m, (Q, P) is minimal above 2^m. Given $Q, P, R > 2^m$,

$$\begin{aligned} &Q - 2P + R > 2^m \\ \Rightarrow\ &(P - Q) + (P - R) < -2^m \\ \Rightarrow\ &\text{At most one of } P - Q \text{ and } P - R \text{ is greater than } 2^m \ . \end{aligned}$$

Since a low step requires the calculation of $P - Q$ and a high step requires $P - R$, only one of these steps satisfies the condition that all parameters be greater than 2^m.

We may make these steps more efficient by combining consecutive low steps as $\left(\begin{smallmatrix} 1 & -q \\ 0 & 1 \end{smallmatrix}\right)$. This gives the formulas

$$\begin{aligned} Q' &\leftarrow Q \ , \\ P' &\leftarrow P - qQ \ , \\ R' &\leftarrow q^2 Q - 2qP + R \ . \end{aligned}$$

We may find the optimal value of q so that all coefficients are greater than 2^m by seeing that $D = P'^2 - QR' \;\Rightarrow\; P'^2 \geq D + 2^m Q$, if $R' \geq 2^m$. Now we have two conditions on P':

$$1)\ P' = \underbrace{P - qQ \geq \sqrt{D + 2^m Q}}_{P'^2 \geq D + 2^m Q} \;\Rightarrow\; q \leq \frac{P - \sqrt{D + 2^m Q}}{Q}$$

$$2)\ P' = \underbrace{P - qQ \geq 2^m}_{\textit{minimal above}\ \text{constraint}} \;\Rightarrow\; q \leq \frac{P - 2^m}{Q}$$

Hence, the optimal value of q is

$$q = \left\lfloor \frac{P - t}{Q} \right\rfloor, \qquad t = \max(\sqrt{D + 2^m Q},\ 2^m)\ .$$

Consecutive high steps may be combined with q found symmetrically as

$$q = \left\lfloor \frac{P - t}{R} \right\rfloor, \qquad t = \max(\sqrt{D + 2^m R},\ 2^m)\ .$$

Given Q, P and R, one combined set of low steps or one combined set of high steps is referred to as a simple step. Algorithm 3.6 performs the simple step operation.

3.3 Schnorr-Seysen

In November 1982, Seysen wrote an unpublished manuscript entitled "An Improved Composition Algorithm." An updated paper under the same title, coauthored with Schnorr, was produced in August 1983. Although never published, this paper contains the essential idea of the algorithms described in Section 3.4 and modern variations of Shanks' NUCOMP algorithm (for example, [6]).

The original algorithm was cast in the language of forms but we present it here in terms of ideals with minor corrections. Also, the notation and arguments of the proof have been changed to the standard notation of continued fractions. Note that the style of the proof has not changed — the authors were indeed using techniques equivalent to continued fraction theory although their notation would suggest that perhaps they were unaware of it.

Recall from Section 2 that multiplication of two reduced ideals (Q_a, P_a) and (Q_b, P_b) of the same discriminant yielding $(S)(Q_0, P_0) = (Q_a, P_a)(Q_b, P_b)$ requires the calculation of

$$\begin{aligned}
&\sigma S = \gcd(Q_a, Q_b, P_a + P_b) = XQ_a + YQ_b + Z(P_a + P_b)\ , \qquad (3.2)\\
&U = Y(P_b - P_a) + ZR_b \pmod{Q_a/S}\ ,\\
&Q_0 = \frac{Q_a Q_b}{\sigma S^2}\ ,\\
&P_0 = \frac{Q_b U}{\sigma S} + P_b\ .
\end{aligned}$$

Algorithm 3.6 Simple Step Above 2^m

Input: Q, P, R, M, m where (Q, P) is an ideal and $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$
Output: Q', P', R', M' where (Q', P') has been reduced one simple step

$D \leftarrow P^2 - QR$ {Store in a lookup table indexed by Q, P, R}
if $(P > Q)$ **then**
  [Low step]
  $t \leftarrow \max(\lceil \sqrt{D + 2^m Q} \rceil, 2^m)$
  $q \leftarrow \left\lfloor \dfrac{P - t}{Q} \right\rfloor$
  $Q' \leftarrow Q$, $P' \leftarrow P - qQ$, $R' \leftarrow R - q(P + P')$
  $M' \leftarrow \begin{pmatrix} a - qc & b - qd \\ c & d \end{pmatrix}$
else
  [High step]
  $t \leftarrow \max(\lceil \sqrt{D + 2^m R} \rceil, 2^m)$
  $q \leftarrow \left\lfloor \dfrac{P - t}{R} \right\rfloor$
  $R' \leftarrow R$, $P' \leftarrow P - qR$, $Q' \leftarrow Q - q(P + P')$
  $M' \leftarrow \begin{pmatrix} a & b \\ c - aq & d - bq \end{pmatrix}$
end if
return Q', P', R', M'

By Equation (2.2) and [17, Thm 4.3], reducing (Q_0, P_0) involves finding e and g such that $|Q_r| = |g^2 Q_0 + 2egP_0 + g^2 R_0| < \sqrt{D}$. Using Equations (3.2) we may derive

$$\begin{aligned} Q_r &= e^2 Q_0 + 2egP_0 + g^2 R_0 \\ &= \frac{e^2 Q_a Q_b}{S^2} + 2\frac{egQ_b U}{S} + \frac{g^2 Q_b U^2}{Q_a} + 2egP_b + 2\frac{g^2 S P_b U}{Q_a} + \frac{g^2 S^2 P_b^2}{Q_a Q_b} - \frac{g^2 S^2 D}{Q_a Q_b} \\ &= \frac{S}{Q_a}\left(\frac{Q_b}{S}\left(\frac{e^2 Q_a^2}{S^2} + 2\frac{egQ_a U}{S} + g^2 U^2\right) + 2gP_b\left(\frac{eQ_a}{S} + gU\right) + g^2 S\frac{P_b^2 - D}{Q_b}\right) \\ &= \frac{S}{Q_a}\left(\frac{Q_b}{S}\left(e\frac{Q_a}{S} + gU\right)^2 + 2gP_b\left(e\frac{Q_a}{S} + gU\right) + g^2 S R_b\right) \quad . \end{aligned} \tag{3.3}$$

The authors recognized that $\left(e\frac{Q_a}{S} + gU\right)$ is the dominant term in this equation and so decreasing its magnitude would decrease the size of Q_r. The extended Euclidean algorithm, on input of Q_a/S and U, outputs numbers e and g such that $\gcd(Q_a/S, U) = e(Q_a/S) + gU$. This is exactly the dominant term in Equation (3.3) and the GCD is the smallest possible integer value (in absolute value) it can take. Developing the simple continued fraction of $U/(Q_a/S)$ is

essentially the same procedure and also computes the numbers e and g.

To decrease the size of Q_r, we show that we do not need to develop the entire continued fraction expansion but only a portion of it. Let $[q_0, q_1, \ldots, q_m]$ be the partial quotients of the simple continued fraction expansion of the rational number $\frac{U}{Q_a/S}$, define A_i and B_i as in Equation (2.9), and

$$C_{-2} = U, \quad C_{-1} = Q_a/S, \quad C_i = C_{i-2} - q_i C_{i-1} = (-1)^{i-1}(A_i C_{-1} - B_i C_{-2}) \ . \tag{3.4}$$

By Equation (2.10), $A_i B_{i-1} - A_{i-1} B_i = (-1)^{i-1}$; therefore, by Proposition 2.9 (with $a \leftarrow B_{i-1}$, $b \leftarrow -A_{i-1}$, $c \leftarrow -B_i$, $d \leftarrow A_i$), if we let $\varepsilon = (-1)^{i-1}$, then (Q_0, P_0) is equivalent to the ideal

$$(\varepsilon(A_i^2 Q_0 - 2A_i B_i P_0 + B_i^2 R_0), \varepsilon(-A_{i-1} A_i Q_0 + (A_i B_{i-1} + A_{i-1} B_i) P_0 - B_{i-1} B_i R_0)) \ . \tag{3.5}$$

If we find r such that

$$C_r \le \sqrt{\frac{Q_a \sqrt{D}}{Q_b}} < C_{r-1} \ , \tag{3.6}$$

and combine this with the fact that the remainders $\{C_r\}$ are strictly decreasing, we obtain the following relations

$$\begin{aligned}
& B_r C_{r-1} + B_{r-1} C_r = C_{-1} \qquad \text{(continued fraction theory)} \\
\Rightarrow\ & B_r C_r < B_r C_{r-1} \le B_r C_{r-1} + B_{r-1} C_r = \frac{Q_a}{S} \\
\Rightarrow\ & B_r \le \frac{Q_a/S}{C_{r-1}} < \frac{Q_a/S}{\sqrt{\frac{Q_a\sqrt{D}}{Q_b}}} \qquad \text{(by (3.6))} \\
\Rightarrow\ & B_r^2 \le \frac{Q_a Q_b}{S^2 \sqrt{D}} \ .
\end{aligned} \tag{3.7}$$

By Equation (3.4) we have $|C_r| = |A_r C_{-1} - B_r C_{-2}| = |A_r \frac{Q_a}{S} - B_r U|$, and substituting these equations into Equation (3.3) we have

$$\begin{aligned}
|Q_r| &\le \left| \frac{S}{Q_a} \left(\frac{Q_b}{S} C_r^2 + 2P_b B_r C_r + B_r^2 S R_b \right) \right| \\
&\le \left| \frac{S}{Q_a} \left(\frac{Q_b}{S} \frac{Q_a \sqrt{D}}{Q_b} + 2P_b \frac{Q_a}{S} + \frac{Q_a Q_b}{S^2 \sqrt{D}} S R_b \right) \right| \qquad \text{(by (3.6 – 3.7))} \\
&\le \left| \sqrt{D} + 2P_b + \frac{Q_b R_b}{\sqrt{D}} \right| \\
&< 4\sqrt{D} \qquad \text{(since } (Q_b, P_b) \text{ is reduced)} \ .
\end{aligned}$$

By [17, Cor. 4.2.1], (Q_r, P_r) is within a few steps of being reduced. In their implementation, Schnorr and Seysen compute (Q_0, P_0) and then reduce it using Equation (3.5). The saving of this method over reducing (Q_0, P_0) classically is that the operations are simpler since we are taking the continued fraction of a rational number instead of a quadratic irrational. Also, the calculation of the values Q_i, P_i, and R_i at each stage is traded for the calculation of A_i and B_i and so we are working with smaller numbers. This is essentially the idea that Atkin [1] developed in his improvement to Shanks' algorithm except that he also found a way to compute (Q_r, P_r) directly (without first needing to compute (Q_0, P_0)) with intermediate operands not usually exceeding $\sqrt{D}$.

3.4 New Ideal Reduction Algorithm

Recall from Section 2.2 that the Lagrange algorithm to reduce an ideal (Q_0, P_0) of positive discriminant is achieved through the continued fraction expansion of $(P_0 + \sqrt{D})/Q_0$. The sequence of ideals (Q_i, P_i) is generated by

$$\frac{P_0 + \sqrt{D}}{Q_0} = \left[q_0, q_1, \ldots, q_i, \frac{P_{i+1} + \sqrt{D}}{Q_{i+1}} \right] \ ,$$

where the values q_i, Q_{i+1}, P_{i+1} are obtained from Algorithm 2.1.

Observe that after multiplication of two reduced ideals, the values of P_0 and Q_0 in the resulting ideal (Q_0, P_0) typically have magnitude D; hence, adding $\sqrt{D}$ to P_0 does not usually affect the value of the quotient obtained. That is, $P_0/Q_0 \approx (P_0 + \sqrt{D})/Q_0$. This means we may eliminate $\sqrt{D}$ from the computations for some time. In particular, we can compute $q_i = \lfloor P_i/Q_i \rfloor$ rather than $q_i = \lfloor (P_i + \sqrt{D})/Q_i \rfloor$. The computational savings are significant since we compute only the simple continued fraction expansion of the rational number P_0/Q_0 instead of the irrational number $(P_0 + \sqrt{D})/Q_0$. The question we ask then is, "At what point do we need to include $\sqrt{D}$ again?" Surprisingly, we do not need to include it at all.

Suppose the partial quotients of the continued fraction expansion of P_0/Q_0 are given by $[\tilde{q}_0, \tilde{q}_1, \ldots, \tilde{q}_m]$. We embed a portion of this sequence into the continued fraction expansion of $(P_0 + \sqrt{D})/Q_0$:

$$\frac{P_0 + \sqrt{D}}{Q_0} = \left[\tilde{q}_0, \tilde{q}_1, \ldots, \tilde{q}_r, \frac{P_{r+1} + \sqrt{D}}{Q_{r+1}} \right] \qquad (r \leq m)$$

where the Q_{r+1} and P_{r+1} values could be computed at each step along the way, or by formulae presented later starting in Equation (3.8). Note that this was also the case in the Schnorr-Seysen algorithm except that we embedded the partial quotients of $\dfrac{U}{Q_a/S}$ instead.

Although the quotients may differ from the "correct" expansion, each new ideal is of course still equivalent to the previous one. What we will show is that this process reduces the size of Q_i such that $|Q_{r+1}| < \sqrt{D}$ for some r, and further that this guarantees (Q_{r+1}, P_{r+1}) is reduced.

In the following discussion, we assume that Q_0 is positive. This poses no problem since $(Q_0, P_0) = (-Q_0, P_0)$.

Theorem 3.2. *Given an ideal (Q_0, P_0) of positive discriminant, let $[\tilde{q}_0, \tilde{q}_1, \ldots, \tilde{q}_m]$ be the partial quotients of the simple continued fraction expansion of P_0/Q_0. Embedding these quotients into the continued fraction expansion*

$$\frac{P_0 + \sqrt{D}}{Q_0} = \left[\tilde{q}_0, \tilde{q}_1, \ldots, \tilde{q}_r, \frac{P_{r+1} + \sqrt{D}}{Q_{r+1}}\right]$$

yields a reduced ideal (Q_{r+1}, P_{r+1}) for some $r \leq m$.

Proof. This is similar to Lagrange reduction except that the quotients $\tilde{q}_i$ as defined above are used instead. Let

$$\begin{aligned} A_{-2} &= 0, \qquad A_{-1} = 1, \qquad A_i = \tilde{q}_i A_{i-1} + A_{i-2} \ , \\ B_{-2} &= 1, \qquad B_{-1} = 0, \qquad B_i = \tilde{q}_i B_{i-1} + B_{i-2} \ . \end{aligned}$$

Then by Equations (2.6), (2.8) and (2.11) we know

$$\begin{aligned} G_i &= Q_0 A_i - P_0 B_i \ , \\ Q_{i+1} &= (-1)^{i+1}\left(\frac{G_i^2 - DB_i^2}{Q_0}\right) \ , \\ P_{i+1} &= (-1)^i\left(\frac{G_i G_{i-1} - DB_i B_{i-1}}{Q_0}\right) = \frac{G_i - Q_{i+1}B_{i-1}}{B_i} \ , \end{aligned} \tag{3.8}$$

and so $|Q_{r+1}| < \max\left(\dfrac{G_r^2}{Q_0}, \dfrac{DB_r^2}{Q_0}\right)$. We want to show that each of these components is less than $\sqrt{D}$ for a suitable choice of r.

From continued fraction theory we know that A_i/B_i approximates P_0/Q_0. In particular, we have the bound (see [12, Cor. 12.3])

$$\begin{aligned} &\left|\frac{A_i}{B_i} - \frac{P_0}{Q_0}\right| < \frac{1}{B_i B_{i+1}} \\ \Rightarrow &\left|\frac{Q_0 A_i - P_0 B_i}{Q_0 B_i}\right| < \frac{1}{B_i B_{i+1}} \\ \Rightarrow &|G_i| < \frac{Q_0}{B_{i+1}} \\ \Rightarrow &G_i^2 < \frac{Q_0^2}{B_{i+1}^2} \ . \end{aligned} \tag{3.9}$$

Since Q_0 is positive, each $\tilde{q}_i > 0$ for $i \geq 1$ ($\tilde{q}_0$ might be 0 or negative) and the sequence $\{B_i\}$ is strictly increasing for $i \geq -1$; therefore, at some point we obtain r such that

$$B_r < \frac{\sqrt{Q_0}}{\sqrt[4]{D}} < B_{r+1} \ . \tag{3.10}$$

This bound, along with Equation (3.9) gives

$$\frac{G_r^2}{Q_0} < \frac{Q_0}{B_{r+1}^2} < \frac{Q_0}{\left(\frac{\sqrt{Q_0}}{\sqrt[4]{D}}\right)^2} = \sqrt{D}$$

and

$$\frac{DB_r^2}{Q_0} < \frac{D\left(\frac{\sqrt{Q_0}}{\sqrt[4]{D}}\right)^2}{Q_0} = \sqrt{D} \ .$$

Therefore $|Q_{r+1}| < \sqrt{D}$ and so (Q_{r+1}, P_{r+1}) is reduced by [17, Thm 4.3]. □

The proof and algorithm are readily adapted to the case of ideals of negative discriminant. In that case we do not have the concept of embedding the quotients into a continued fraction expansion but the proof does not depend on this.

The reduction algorithm is given as Algorithm 3.7. We are fortunate that the calculation of G_i does not require any extra steps in this algorithm. Given C_{-2} and C_{-1}, by Equation (3.4) the remainders in the continued fraction expansion of C_{-2}/C_{-1} are defined to be

$$C_i = C_{i-2} - \tilde{q}_i C_{i-1} \ .$$

In Algorithm 3.7, we set $C_{-2} = P_0$ and $C_{-1} = Q_0$ and so Equation (3.4) tells us that $C_i = (-1)^{i-1}(A_i C_{-1} - B_i C_{-2})$; hence, $G_i = (-1)^{i-1} C_i$. This eliminates the necessity of calculating the $\{A_i\}$.

From the inequality given in Equation (3.10) we observe that the $\{B_i\}$ are bounded in magnitude by $\sqrt[4]{|D|}$. If the continued fraction expansion of P_0/Q_0 is implemented using Lehmer's extended GCD algorithm, most of the calculations in Algorithm 3.7 will involve relatively small numbers.

Unlike all of the other algorithms presented in this paper, the knowledge of R_0 does not give any advantage in the computations. This assertion is supported both by our tests and an analysis of the number of operations required. Equation (2.5) gives the following formulas which utilize R_0. The formula is given for the computation of A_i since A_i and A_{i-1} are also required. Note that the $\{A_i\}$ may instead be computed as a sequence in Algorithm 3.7 using the same formula as the sequence of $\{B_i\}$ but with initial values $A_{-2} = 0$ and $A_{-1} = 1$. In our tests, it was more efficient to compute A_{i-1} and A_i at then end using the formula below rather than computing the entire sequence for radicand bit sizes less than 16384 bits.

Algorithm 3.7 Efficient Ideal Reduction

Input: $\mathfrak{a}_0 = (Q_0, P_0)$, D, where $Q_0,\ P_0 > 0$
Output: Ideal $\mathfrak{a}_{i+1}$ and Ψ_{i+1} such that $\mathfrak{a}_{i+1} = (\Psi_{i+1})\mathfrak{a}_0$ (If $D > 0$, $\mathfrak{a}_{i+1}$ is reduced; if $D < 0$, $\mathfrak{a}_{i+1}$ is within 3 steps of being reduced.)

[Initialization]
$i \leftarrow -2$
$B_{-2} \leftarrow 1,\ B_{-1} \leftarrow 0$
$C_{-2} \leftarrow P_0,\ C_{-1} \leftarrow Q_0$

[Continued fraction expansion of P_0/Q_0]
while $B_{i+1} < \left\lfloor \sqrt{\lfloor\, Q_0/\lfloor\sqrt{|D|}\rfloor\,\rfloor} \right\rfloor$ **do**
  $i \leftarrow i + 1$
  $q_{i+1} \leftarrow \lfloor C_{i-1}/C_i \rfloor,\ C_{i+1} \leftarrow C_{i-1} - q_{i+1}C_i$
  $B_{i+1} \leftarrow q_{i+1}B_i + B_{i-1}$
end while

[Compute (Q_{i+1}, P_{i+1}) and Ψ_{i+1}]
$$Q_{i+1} \leftarrow (-1)^{i+1}\left(\frac{C_i^2 - DB_i^2}{Q_0}\right)$$
$$P_{i+1} \leftarrow \frac{(-1)^{i+1}C_i - Q_{i+1}B_{i-1}}{B_i}$$
$$\Psi_{i+1}^{-1} \leftarrow B_{i-1} + B_i\left(\frac{P_{i+1} - \sqrt{D}}{Q_{i+1}}\right)$$
return (Q_{i+1}, P_{i+1}), Ψ_{i+1}

$$\begin{aligned} A_i &= \frac{G_i + P_0B_i}{Q_0} \\ F_i &= A_iP_0 - B_iR_0 \\ Q_{i+1} &= (-1)^{i+1}(A_iG_i - B_iF_i) \\ P_{i+1} &= (-1)^{i+1}(-A_{i-1}G_i + B_{i-1}F_i) \\ R_{i+1} &= (-1)^{i+1}(-A_{i-1}G_{i-1} + B_{i-1}F_{i-1}) \end{aligned}$$

We present Table 3.4 which summarizes the calculations required to compute Q_i, P_i, and R_i. In each case we give a count of each type of operation, both when R_0 is utilized and when it is not. The operations counts come from the formulas given in Equations (2.5) – (2.8).

The size of the denominator in the divisions is important, so note that to calculate Q_i and P_i without R_0 requires a division by Q_0, which is usually the same size as D when (Q_0, P_0) is the product of two reduced ideals, and a division by B_i which is approximately the same size as $\sqrt[4]{|D|}$. The two divisions

	Calculated	Multiplications	Divisions	Additions
Without R_0	Q_i, P_i	4	2	2
Utilizing R_0	Q_i, P_i	8	2	5
Without R_0	Q_i, P_i, R_i	5	3	3
Utilizing R_0	Q_i, P_i, R_i	12	2	7

Table 3.1: Calculation of Q_i, P_i and R_i

utilizing knowledge of R_0 are both by Q_0. In the case of calculating R_i without knowledge of R_0, the extra division is by Q_i which is approximately the same size as $\sqrt{|D|}$.[3]

4 Timings

We present timings for the algorithms discussed in this paper with the intent of discovering the best algorithm to use for each application. We have not implemented the Schnorr-Seysen algorithm because it is the same as JSW-NUCOMP except that it requires the computation of the unreduced product ideal. Thus in an application that requires both multiplication and reduction, it will clearly be slower than JSW-NUCOMP. As well, it cannot be used strictly for reduction since it requires knowledge of the terms Q_a/S and U used in the multiplication.

In our key exchange tests the relative generator is used to ensure that both parties compute the same key. It is not possible to work with exact representations of the relative generator and so an approximation must be made. We have used the (f, p) ideal representation of [6]. This allows us to work efficiently with an approximation of the relative generator at the precision required to obtain matching keys.

All timings were taken on a computer with the following configuration:

- Platform: IBM x330 Server
- Processor: Dual Pentium IV Xeon 2.8 GHz
- RAM: 1 GB
- Operating System: Red Hat Linux Release 9
- Linux Kernel: 2.4.20-28.9smp
- Compiler: gcc 3.2.2
- Multi-precision Library: GMP 4.1.2

[3]The size of Q_0 is discussed in Section 2 while the sizes of B_i and Q_i come from the proof of Theorem 3.2.

4.1 Reduction Only

To compare the algorithms' performance, prime ideals were generated and then reduced. When the ideal parameters Q and P were more than 8192 bits, a prime ideal whose parameters were 8192 bits was squared as many times as required to obtain the desired size for the parameters.

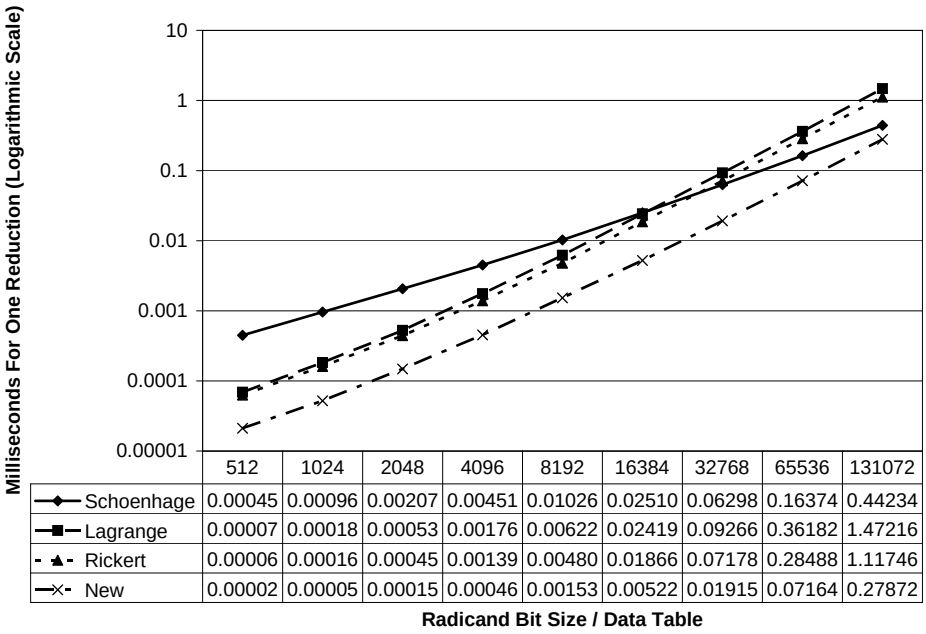
For each radicand size k from 512 bits to 131072 bits (the number of bits was doubled each time), 2500 ideals were reduced as follows. Ten radicands (D) of size k were randomly chosen. For each of these ten D, fifty ideals (prime ideals or a power of a prime ideal as previously discussed) whose parameters Q and P had a bit length of k were randomly chosen and then reduced with each of the four algorithms. Then, fifty prime ideals whose parameters had a bit length of $2k$ were reduced, next fifty prime ideals whose parameters had a bit length of $4k$ were reduced, then fifty prime ideals whose parameters had a bit length of $8k$ were reduced, and finally fifty prime ideals whose parameters had a bit length of $16k$ were reduced.

In total, 36 sets of timings (9 radicand sizes times 4 reduction algorithms) were completed. The following two charts have been chosen to illustrate the results. In Figure 1, the size of the radicand varies from 512 bits to 131072 bits. For each radicand, the size of the ideal parameters match the size of the radicand. The chart is presented using a logarithmic scale and it shows that in every case the new algorithm vastly outperforms the other three algorithms. At all practical sizes it is at least three times faster than each of the other three algorithms. The chart also demonstrates the asymptotic superiority of Schönhage's algorithm.

In Figure 2, the size of the radicand is fixed at 131072 bits. The size of the paramaters of the ideals range from 131072 bits to 2097152 bits. For paramaters larger than 262144 bits, Schönhage's algorithm outperforms our new algorithm. These ideal reduction timings indicate that Schönhage's algorithm would rarely be used in practice. For typical applications such as solving norm equations, computing the ideal class group, and computing the ideal class number, the new reduction algorithm is the optimal choice.

4.2 Reduction Following Multiplication

In most cases, reduction is required as the result of multiplying two reduced ideals. To evaluate the performance of the algorithms a cryptographic key exchange was performed and the time required for each party to exchange a key was recorded. We executed the key exchange protocol using radicands of 795, 1384, 1732, 3460, and 5704 bits with 1000 randomly selected radicands chosen for each bit size. These are the recommended sizes given in [6] to provide at least 80, 112, 128, 192, and 256 bits of security. In all of the key exchanges except JSW-NUCOMP, Shanks' formulas given in Equation (2.1) were used for all ideal multiplications. In the case of squaring an ideal, the formulas were optimized in a straight-forward manner. This eliminates one of the GCD computations and simplifies some of the other equations.



	512	1024	2048	4096	8192	16384	32768	65536	131072
Schoenhage	0.00045	0.00096	0.00207	0.00451	0.01026	0.02510	0.06298	0.16374	0.44234
Lagrange	0.00007	0.00018	0.00053	0.00176	0.00622	0.02419	0.09266	0.36182	1.47216
Rickert	0.00006	0.00016	0.00045	0.00139	0.00480	0.01866	0.07178	0.28488	1.11746
New	0.00002	0.00005	0.00015	0.00046	0.00153	0.00522	0.01915	0.07164	0.27872

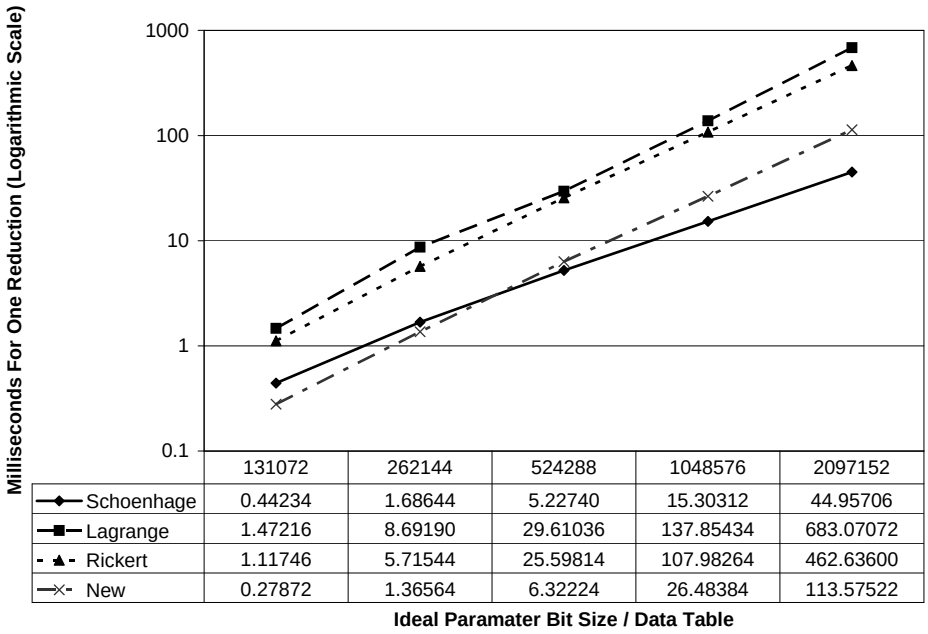
Figure 1: Reduction Comparison — Q same size as D, size of D increases

Figure 3 details the key exchange timings. The timings are for the entire key exchange including the multiplication, reduction and near ideal computations. The keys of both parties were compared to ensure they matched. Additionally, the keys returned by each of the algorithms were compared to ensure that each algorithm indeed generated the same key. We can see that JSW-NUCOMP is the fastest method available for all sizes of radicand when both multiplication and reduction is required.

5 Conclusion

In this paper we surveyed all of the known reduction algorithms for ideals. With the exception of the Jacobson-Scheidler-Williams NUCOMP, this is the first time the modern algorithms have been presented in this language. It is also the first time that the algorithms of Rickert, Schönhage, and Schnorr-Seysen have been adapted to ideals of positive discriminant. We provided an easy method of calculating the relative generator from any type of reduction algorithm and provided many example implementations using this method.

It is notable that the Schnorr-Seysen algorithm anticipated the new algorithm of Section 3.4 and the improvements to NUCOMP presented in [6]. All three of these algorithms embed the quotients of an approximation to $(P_0 + \sqrt{D})/Q_0$ into the continued fraction expansion of $(P_0 + \sqrt{D})/Q_0$ but the logic



	131072	262144	524288	1048576	2097152
Schoenhage	0.44234	1.68644	5.22740	15.30312	44.95706
Lagrange	1.47216	8.69190	29.61036	137.85434	683.07072
Rickert	1.11746	5.71544	25.59814	107.98264	462.63600
New	0.27872	1.36564	6.32224	26.48384	113.57522

Figure 2: Reduction Comparison — D is 131072 bits, size of Q increases

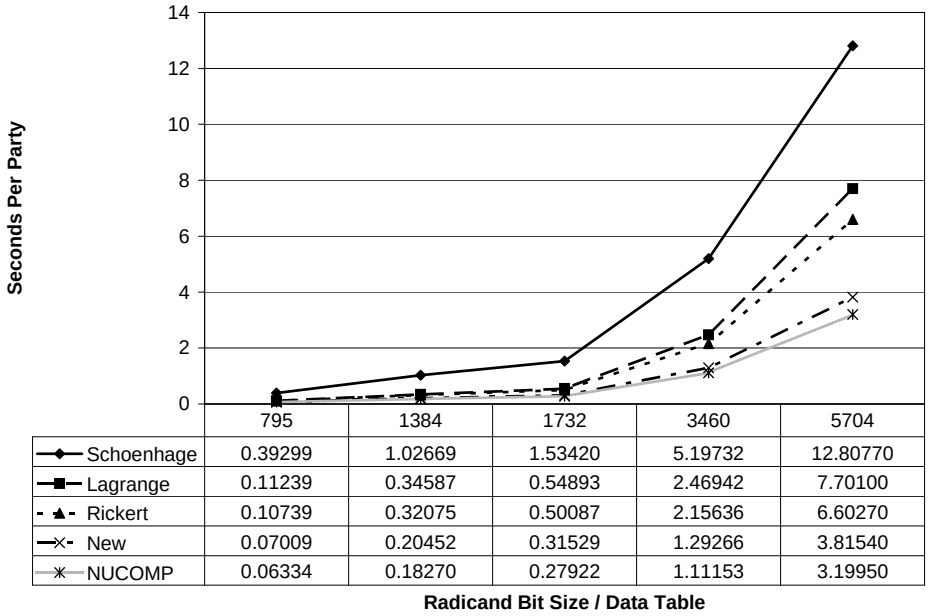
by which they were discovered was radically different. All of the algorithms with the exception of the Schnorr-Seysen algorithm were implemented and compared. The Schnorr-Seysen algorithm was not implemented because the technique is nearly identical to Jacobson-Scheidler-Williams NUCOMP except that efficient formulas for multiplication are not used.

For practical purposes, NUCOMP is the best algorithm to use when multiplication and reduction are both required, and the new algorithm presented here is the best when only reduction is required. Asymptotically, Schönhage's algorithm is the fastest; however, it would only be used for extremely large bit sizes.

All of the algorithms in this paper may also be used for reducing binary quadratic forms. Two binary quadratic forms are not properly equivalent unless the matrix relating the two forms has a determinant equal to 1 (equivalent to $\varepsilon = 1$ in this paper). This must be considered when using these algorithms with binary quadratic forms. For example, Lagrange's method can be used to reduce a form but ε alternates between $+1$ and -1. Any binary quadratic form reduction algorithm would need to ensure that the combined steps of the reduction had $\varepsilon = 1$.

Further work which may done in this area includes:

- Using the new algorithm in place of Lagrange reduction in the single-precision reductions of Rickert's algorithm.



	795	1384	1732	3460	5704
Schoenhage	0.39299	1.02669	1.53420	5.19732	12.80770
Lagrange	0.11239	0.34587	0.54893	2.46942	7.70100
Rickert	0.10739	0.32075	0.50087	2.15636	6.60270
New	0.07009	0.20452	0.31529	1.29266	3.81540
NUCOMP	0.06334	0.18270	0.27922	1.11153	3.19950

Figure 3: Key Exchange Comparison

- Incorporating Lehmer-style single precision calculations into Schönhage's algorithm. For example, the while loops and their simple steps may benefit from this.
- Further practical efficiencies may be derived in Schönhage's algorithm from performing some separate action once the parameters get close to the base case.
- The key exchange may be made more efficient by using better exponentiation techniques such as those described in [10]. For example, windowing trades off some precomputation in exchange for fewer and faster calculations during the main exponentiation loop.
- The algorithms utilizing continued fractions may be able to benefit from Shanks' Baby-Step Giant-Step method to take giant steps toward a near reduced ideal.
- Adapting these techniques to higher degree number fields and function fields.

Undoubtedly, there is an abundance of work to be done in this area of research. Advancements will be immediately applicable to diverse areas such as solving the Pell equation, cryptography, and calculating the fundamental unit or regulator of a quadratic field.

References

[1] A.O.L. Atkin, *Letter to D. Shanks on the programs NUDUPL and NUCOMP, 12 December 1988*, From the Nachlass of D. Shanks, made available by Hugh C. Williams.

[2] D.A. Buell, *Binary Quadratic Forms*, Springer-Verlag, New York, 1989, Classical theory and modern computations. MR 92b:11021

[3] H. Cohen, *A Course In Computational Algebraic Number Theory*, Graduate Texts in Mathematics, vol. 138, Springer-Verlag, Berlin, 1993. MR 94i:11105

[4] L.E. Dickson, *History Of The Theory Of Numbers. Vol. III: Quadratic And Higher Forms.*, With a chapter on the class number by G.H. Cresse, Chelsea Publishing Co., New York, 1966. MR 39 #6807c

[5] J.B. Fraleigh, *A First Course In Abstract Algebra*, sixth ed., Addison-Wesley Publishing Co., Don Mills, Ont., 1998.

[6] M.J. Jacobson, Jr., R. Scheidler, and H.C. Williams, *An improved real quadratic field based key exchange procedure*, To appear in Journal of Cryptology.

[7] T. Jebelean, *A double-digit Lehmer-Euclid algorithm for finding the GCD of long integers*, J. Symbolic Comput. **19** (1995), no. 1-3, 145–157, Design and implementation of symbolic computation systems (Gmunden, 1993). MR 96h:11128

[8] P. Kaplan and K.S. Williams, *The distance between ideals in the orders of a real quadratic field*, Enseign. Math. (2) **36** (1990), no. 3-4, 321–358. MR 92e:11028

[9] D.H. Lehmer, *Euclid's algorithm for large numbers*, The American Mathematical Monthly **45** (1938), no. 4, 227–233.

[10] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook Of Applied Cryptography*, CRC Press Series on Discrete Mathematics and its Applications, CRC Press, Boca Raton, FL, 1997, With a foreword by Ronald L. Rivest. MR 99g:94015

[11] N.W. Rickert, *Efficient reduction of quadratic forms*, Computers And Mathematics (Cambridge, MA, 1989), Springer, New York, 1989, pp. 135–139. MR 90f:11049

[12] K.H. Rosen, *Elementary Number Theory And Its Applications*, fourth ed., Addison-Wesley, Reading, MA, 2000. MR 2000i:11001

[13] R.E. Sawilla, *Fast ideal arithmetic in quadratic fields*, Master's thesis, University of Calgary, Canada, August 2004.

[14] C.P. Schnorr and M. Seysen, *An improved composition algorithm*, August 1983.

[15] A. Schönhage, *Fast reduction and compostion of binary quadratic forms*, International Symposium On Symbolic And Algebraic Computation (ISSAC), ACM Press, 1991, pp. 128–133.

[16] I. Stewart and D. Tall, *Algebraic Number Theory*, second ed., Chapman and Hall Mathematics Series, Chapman & Hall, London, 1987. MR 88d:11001

[17] H.C. Williams and M.C. Wunderlich, *On the parallel generation of the residues for the continued fraction factoring algorithm*, Math. Comp. **48** (1987), no. 177, 405–423. MR 88i:11099