# SQUARE FORM FACTORING WITH SIEVES

by

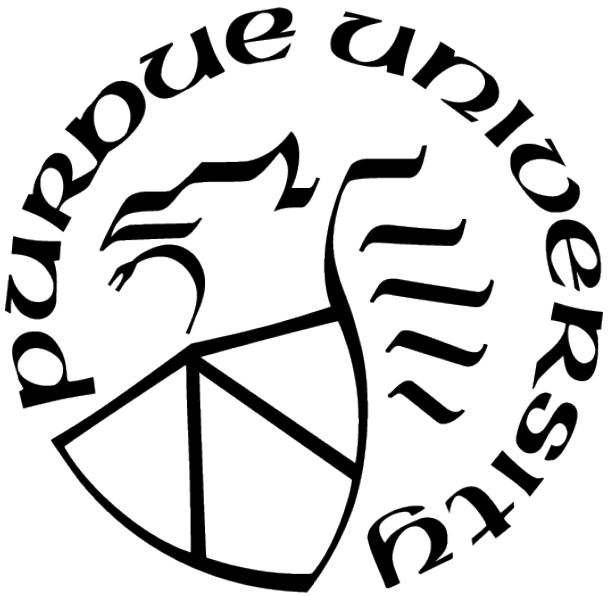
**Clinton Bradford**

**A Thesis**

*Submitted to the Faculty of Purdue University*

*In Partial Fulfillment of the Requirements for the degree of*

**Doctor of Philosophy**




Department of Mathematics

West Lafayette, Indiana

May 2021

# THE PURDUE UNIVERSITY GRADUATE SCHOOL
# STATEMENT OF COMMITTEE APPROVAL

**Dr. Samuel S. Wagstaff, Chair**

Department of Mathematics

**Dr. William Heinzer**

Department of Mathematics

**Dr. Tong Liu**

Department of Mathematics

**Dr. David McReynolds**

Department of Mathematics

**Approved by:**

Dr. Plamen Stefanov

# ACKNOWLEDGMENTS

I would not have reached this point without the assistance of many people, most of all my wife, without whom I would be completely lost in this world. For this thesis in particular, the patience, advice, and support of my advisor was essential, and the broader community in the department and field.

# TABLE OF CONTENTS

# ABSTRACT

Square Form Factoring is an $O(N^{1/4})$ factoring algorithm developed by D. Shanks using certain properties of quadratic forms. Central to the original algorithm is an iterative search for a square form. We propose a new subexponential-time algorithm called SQUFOF2, based on ideas of D. Shanks and R. de Vogelaire, which replaces the iterative search with a sieve, similar to the Quadratic Sieve.

# 1. INTRODUCTION

## 1.1 History of Square Forms Factoring

Quadratic equations have been used for factoring numbers since early in their history, and are present in many factoring algorithms including several of the fastest in use, such as the quadratic sieve proposed by Pomerance in [1]. One algorithm, called SQUFOF or Square Forms Factoring, was invented by Daniel Shanks at some point around 1980, though never published in his lifetime. SQUFOF was described in an unpublished manuscript [2] found in his office after his death. Nearly 30 years later, Gower and Wagstaff [3] published a description of the algorithm and a completed version of the heuristic argument presented in the manuscript. This algorithm is exponential, taking $O(N^{1/4})$ time to factor an integer $N$. However, discussions between R. de Vogelaire and Shanks noted at the end of section 3 of the manuscript lead to a new algorithm, which we call SQUFOF2, which factors $N$ in expected subexponential time $O(\exp(1.02\sqrt{\log N \log\log N}))$, similar to modern factorization methods.

As the names suggest, SQUFOF and SQUFOF2 rely heavily on the theory of quadratic forms in their operation. Both algorithms work in the space of indefinite forms, that is, forms of positive discriminant. Both algorithms use the idea of a square form, that is, a quadratic equation $s^2x^2 + bxy + cy^2$ of discriminant $\Delta = N$ or $\Delta = 4N$, to locate an "ambiguous" quadratic form with a coefficient dividing the discriminant. The main difference between the algorithms is in how the square form is found. SQUFOF finds the square form by searching a sequence of forms iteratively. SQUFOF2 uses polynomial sieving techniques akin to those in the Quadratic Sieve, together with the arithmetic of quadratic forms, to construct a square form in significantly less time on average.

There are more algorithms which construct these "ambiguous" forms. In 1992, Lenstra and Pomerance in proposed a factoring algorithm (algorithm 10.4 in [4]) using the arithmetic of "positive definite" quadratic forms, that is, forms of negative discriminant, together with matrix techniques. We will briefly discuss similarities and differences between this and SQUFOF2 in section 2.3.2.

# 2. BACKGROUND

## 2.1 Square Forms

The algorithm SQUFOF2 relies, as the original SQUFOF does, on the theory of *primitive integral binary quadratic forms of discriminant* $\Delta$, here referred to as *quadratic forms*, or simply *forms*. These are given as triples $f = (a, b, c)$, representing the function $f(x, y) = ax^2 + bxy + cy^2$, with discriminant $b^2 - 4ac = \Delta$. $f$ is primitive if $\gcd(a, b, c) = 1$, and integral if $a, b, c \in \mathbb{Z}$. We will be restricting our discussion to *indefinite quadratic forms*, that is, quadratic forms with discriminant $\Delta = b^2 - 4ac$ positive. It is worth noting that there are a few common, slightly different, notations for quadratic forms in the literature and it is worth being careful in comparing sources.

There are, for a given discriminant, infinitely many quadratic forms, with arbitrarily large coefficients. However, we will be able to focus mainly on a finite number of forms, called *reduced forms*, the finite set of forms satisfying the inequalities:

$$\left|\sqrt{\Delta} - 2|a|\right| < b < \sqrt{\Delta}$$

For small discriminant, it is a simple task to enumerate the reduced forms. We will use, for demonstration, the forms with $\Delta = 40$. Written in decreasing order of $0 < b < \sqrt{\Delta} \approx 6.325$, we have 8 which satisfy the bounds:

| | | |
|---|---|---|
| $(1, 6, -1)$ | $(-1, 6, 1)$ | $(3, 4, -2)$ |
| $(-3, 4, 2)$ | $(2, 4, -3)$ | $(-2, 4, 3)$ |
| $(3, 2, -3)$ | $(-3, 2, 3)$ | |

For a given $\Delta$ we will write the set of forms $\mathbf{F}$ and the set of reduced forms $\mathbf{R}$. We next will need to explore how to convert, in a useful way, from a general form of discriminant $\Delta$ to a reduced form.

### 2.1.1 Equivalence of Forms

One concept central to quadratic forms is an action by $SL_2(\mathbb{Z})$ on the quadratic forms of discriminant $\Delta$. Suppose we have a matrix

$$M = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$

Then we can transform $(x, y)$ using this $M$:

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \alpha x + \beta y \\ \gamma x + \delta y \end{pmatrix}$$

Substituting into $f$ gives us a quadratic form:

$$\begin{aligned} f'(x, y) &= f(\alpha x + \beta y, \gamma x + \delta y) \\ &= a(\alpha x + \beta y)^2 + b(\alpha x + \beta y)(\gamma x + \delta y) + c(\gamma x + \delta y)^2 \\ &= f(\alpha, \gamma)x^2 + (2(a\alpha\beta + c\gamma\delta) + b(\alpha\delta + \beta\gamma))xy + f(\beta, \delta)y^2 \end{aligned}$$

It can be convenient, especially when considering the action of a matrix on a form, to conceptualize the form as a symmetric matrix:

$$\begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix} \in GL_2(\mathbb{Q})$$

of determinant $ac - b^2/4 = -\Delta/4$, evaluated at $x, y$ by:

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = ax^2 + bxy + cy^2$$

Applying the definition of the action by $M$ above, we see that $Mf$ is given by:

$$\begin{pmatrix} x & y \end{pmatrix} M^T \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix} M \begin{pmatrix} x \\ y \end{pmatrix} = a'x^2 + b'xy + c'y^2$$

From this it is clear that the determinant of this matrix is $-\Delta/4$, and so the form $Mf$ has discriminant $\Delta$. This operation, then, defines an action by $SL_2(\mathbb{Z})$ on quadratic forms of discriminant $\Delta$. We say $f$ is *equivalent* to $g$ if they are in the same $SL_2(\mathbb{Z})$ orbit, that is, if there is some $M \in SL_2(\mathbb{Z})$ with $Mf = g$.

$SL_2(\mathbb{Z})$ is generated by the matrices $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ and $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$. We will also consider the action by the set of powers of this second generator, which we will call $\Gamma = \left\{ \begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix} \middle| \lambda \in \mathbb{Z} \right\}$. The action of these generators will prove useful to us when dealing with a few properties of quadratic forms, and so we will write them out:

$$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} c & -b/2 \\ -b/2 & a \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} (a, b, c) = (c, -b, a)$$

$$\begin{pmatrix} 1 & 0 \\ \lambda & 1 \end{pmatrix} \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix} \begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a & a\lambda + \frac{b}{2} \\ a\lambda + \frac{b}{2} & a\lambda^2 + b\lambda + c \end{pmatrix}$$

$$\begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix} (a, b, c) = (a, b + 2a\lambda, a\lambda^2 + b\lambda + c)$$

### 2.1.2 Reduction Operators

We define an operator $\rho : \mathbf{F} \to \mathbf{F}$, applied to a form $(a, b, c)$, by the action of $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ mapping to the form $(c, -b, a)$, followed by the action of $\begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix}$ mapping to the form $(c, -b+2c\lambda, c\lambda^2 - b\lambda + a)$ with $\lambda \in \mathbb{Z}$ chosen to get a reduced form quickly; When $|c| > \sqrt{\Delta}$, we choose $\lambda$ such that the new middle coefficient is as small as possible, that is, $-|c| < -b + 2c\lambda < |c|$. Note that this choice is unique. When $|c| < \sqrt{\Delta}$, we choose $\lambda$ such that

$\sqrt{\Delta} - 2|c| < -b + 2c\lambda < \sqrt{\Delta}$, attempting to reach a reduced form. Note that $\rho(f)$ is not necessarily reduced, even in the second case.

By Proposition 5.6.6 of Cohen [5], repeated applications of $\rho$ will yield a reduced form from a form $(a, b, c)$ in less than $\left\lceil \frac{\log |c|}{\sqrt{\Delta}} \right\rceil + 2$ steps. We denote $k$ repeated applications of $\rho$ by $\rho^k$. On reduced forms, we can also let $k$ be negative by noting that, with the operation $\tau : (a, b, c) \mapsto (c, b, a)$, $\rho(\tau(\rho(\tau(f)))) = f$, that is, $\tau\rho\tau$ is the inverse of $\rho$ and so $\rho^{-k} = \tau\rho^k\tau$. Write $\rho_0 : \mathbf{F} \to \mathbf{R}$ to be the operator taking a form $f$ to the form $\rho^k(f)$ for the smallest nonnegative integer $k$ such that $\rho^k(f)$ is reduced.

One notable fact is that, if $f$ is a reduced form, $\rho(f)$ is a reduced form. Notably, in all cases with positive nonsquare $\Delta$, $\rho(f)$ is a different reduced form than $f$. One easy way to see this fact is that, for a reduced form, as $b^2 - 4ac = \Delta$ with $b < \sqrt{\Delta}$, $-4ac > 0$ and thus $a, c$ must be of opposite sign. Thus we can partition the forms into cycles under $\rho$.

As an example of reduction, consider the reduction operator applied to $(-2, 4, 3)$. First, we apply the matrix $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ to obtain the form $(3, -4, -2)$. Next, we pick an integer $\lambda$ such that, under the action by $\begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix}$, the new middle coefficient $-4 + 2 \cdot 3 \cdot \lambda$ is in the interval $(\sqrt{\Delta} - 2|c|, \sqrt{\Delta})$, that is, the closest possible positive integer to $\sqrt{\Delta} \approx 6.32$ without exceeding it. In this example, $\lambda = 1$, giving the form $(3, 2, -3)$.

Calculating the reduction for all reduced forms of discriminant 40, we can partition $R$ into cycles under reduction:

| | |
|---|---|
| $(1, 6, -1)$ | $(-2, 4, 3)$ |
| $(-1, 6, 1)$ | $(3, 2, -3)$ |
| | $(-3, 4, 2)$ |
| | $(2, 4, -3)$ |
| | $(-3, 2, 3)$ |
| | $(3, 4, -2)$ |

One easy way to partially verify these cycles, given the list of points, is to use the fact that the $c$ coefficient for each form will be the $a$ coefficient for the next. Note, however, there are multiple forms with $a$ coefficient 3, so this does not fully determine the order. Note

that these cycles are not necessarily of the same length for a given discriminant $\Delta$, as in this example we have one cycle of length 2 and another of length 6.

### 2.1.3 Representation of Integers

For a given form $f = (a, b, c)$, $f$ *represents* an integer $k$ if there exist some $x, y$ such that $F(x, y) = ax^2 + bxy + cy^2 = k$. $f$ *primitively represents* $k$ if there exist an $x, y$ pair which are relatively prime at which $f$ represents $k$. When $f$ primitively represents $k$, we can construct integers $s, t$ such that $(k, s, t)$ is a form equivalent to $f$, as follows. As $\gcd(x, y) = 1$, there exist integers $w, z$ such that $xw - yz = 1$. Then we construct the matrix

$$\begin{pmatrix} x & z \\ y & w \end{pmatrix} \in SL_2(\mathbb{Z})$$

Consider its action on a form $f$. This produces a form $(f(x, y), b(xw + zy) + 2(axz + cyw), f(z, w)) = (k, s, t)$. This construction was known to Gauss, and discussed in [6] at the start of section 4.1.

### 2.1.4 Ambiguous Forms

An *ambiguous form* is defined to be a form $(a, b, c)$ which satisfies $a|b$. For the purpose of integer factorization, it is notable that an ambiguous form must satisfy $a|b^2 - 4ac = \Delta$. If the form is reduced, then we know that $a < \sqrt{\Delta}$, and so $\gcd(a, \Delta)$, when $a$ has an odd prime factor, will be a proper divisor of an odd $N$ when $\Delta = 4N$ or $\Delta = N$. Gower and Wagstaff establish in [3] that, for fundamental discriminant $\Delta$, at least half of reduced ambiguous forms will lead to a proper divisor. For this reason, both SQUFOF and SQUFOF2 work by finding such an ambiguous form.

Proposition 3.8 of [6] notes the structure of ambiguous forms in cycles of quadratic forms. In any cycle which contains an ambiguous form, it always contains precisely two ambiguous forms. Ambiguous forms occur as the second of a pair of adjacent forms $(a', b, a)$ and $(a, b, a')$, immediately following each other. Such a form can be recognized easily, when traversing the

cycles by reduction, by comparing the computed middle coefficients. When they match, then we have reached this *symmetry point* and thus an ambiguous form.

### 2.1.5 Composition of Forms

We cannot directly multiply two quadratic equations and expect to get a quadratic equation of the same degree. However, something quite similar to multiplication proves useful; undergoing a change of variables allows for an operation based on multiplication, on the forms. *Form composition* is a binary operation $F \times F \to F$, defined mechanically. We will follow the algorithm given in theorem 4.10 of Buell [6]. Suppose we have forms of the same discriminant $f_1 = (a_1, b_1, c_1), f_2 = (a_2, b_2, c_2)$, computing a form $f_3 = (a_3, b_3, c_3)$ which in a sense equals $(a_1, b_1, c_1) \cdot (a_2, b_2, c_2)$. We calculate some intermediate values; $\beta = (b_1 + b_2)/2$, $n = \gcd(a_1, a_2, \beta)$, and $t, u, v$ such that $a_1 t + a_2 u + \beta v = n$. This choice of $t, u, v$ unfortunately matters for the resulting form, offsetting by a multiple of $2a_3$ added or subtracted from $b_3$. However, in all theoretical applications, we will be considering the operation on the space of forms modulo $\Gamma$, where the differences by multiples of $2a_3$ in $b_3$ disappears. For our practical purposes, we will compute this greatest common divisor as $\gcd(\gcd(a_1, a_2), \beta)$, choosing the smallest nonnegative coefficient for the first term in each extended gcd.

These values are used to compute a new middle coefficient

$$B = \frac{a_1 b_2 t + a_2 b_1 u + v(b_1 b_2 + \Delta)/2}{n}$$

The referenced theorem gives the substitution:

$$\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 & \frac{b_2 - B}{2a_2} & \frac{b_1 - B}{2a_1} & \frac{b_1 b_2 + \Delta - B(b_1 + b_2)}{4a_1 a_2} \\ 0 & a_1 & a_2 & \frac{b_1 + b_2}{2} \end{pmatrix} \begin{pmatrix} x_1 x_2 \\ x_1 y_2 \\ y_1 x_2 \\ y_1 y_2 \end{pmatrix}$$

Under which the direct multiplication holds:

$$(a_1 x_1^2 + b_1 x_1 y_1 + c_1 y_1^2) \cdot (a_2 x_2^2 + b_2 x_2 y_2 + c_2 y_2^2) = a_1 a_2 x_3^2 + B x_3 y_3 + C y_3^2$$

with $C$ computed according to the discriminant formula. Written in the notation of forms,

$$(a_1, b_1, c_1) \cdot (a_2, b_2, c_2) = \left(a_1 a_2, B, \frac{B^2 - \Delta}{4 a_1 a_2}\right)$$

We will only be performing two specific compositions which are both squarings of forms, and as such simpler cases. First, we will be composing the reduced form $F_1 = (1, b, c)$, known as the *principal form*, with itself. Note that if $a_1 = 1$, then $B = b_2$, and the form returned is $(a_2, b_2, c_2)$; that is, any form with leading coefficient 1 is an identity under this definition of composition. However, we will want to take the product of representations of integers by the form, and this definition of composition allows us to do just that. When composing the identity form with itself, substituting appropriately, we have several cancellations resulting in a simplified matrix for the new coordinates:

$$\begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -c \\ 0 & 1 & 1 & b \end{pmatrix} \begin{pmatrix} x_1 x_2 \\ x_1 y_2 \\ y_1 x_2 \\ y_1 y_2 \end{pmatrix} = \begin{pmatrix} x_1 x_2 - c y_1 y_2 \\ x_1 y_2 + y_1 x_2 + b y_1 y_2 \end{pmatrix}$$

This allows us to construct, given a representation of $z_1$ and $z_2$, a representation of $z_1 z_2$. Note, however, that this newly constructed representation is not necessarily a primitive representation, even if $z_1$ and $z_2$ are represented primitively. This will not be an obstacle for our algorithm, however, as we find the primitive representation of $z_3 / \gcd(x_3, y_3)^2$ by $(x_3 / \gcd(x_3, y_3), y_3 / \gcd(x_3, y_3))$ as useful.

The other composition we will need is the squaring of a form $f = (a, b, c)$ other than the identity. We will not need to track representations for this, and the operation is almost as simple. With $n = \gcd(a, b)$, and $t, v$ such that $at + bv = n$, we have

$$B = b - 2acv/n$$

Giving a square operation

$$(a, b, c)^2 = (a^2/n^2, b - 2acv/n, C)$$

A natural question is whether this operation makes the space of forms into a group. It has an identity, and a concept of inverses; for a form $(a, b, c)$, the form $(a, -b, c)$ has in the composition $n = \gcd(a, a, 0) = a$, and thus leading coefficient $a^2/a^2 = 1$. The next question is whether it is associative. This can be seen a variety of ways, but here we will show by direct example. First, we evaluate the composition of $(-2, 4, 3)$ and $(3, 2, -3)$. We compute $\beta = 3$, $n = \gcd(-2, 3, 3) = 1$, and are given a choice for the $t, u, v$; we will choose $t = 1, u = 1, v = 0$ for simplicity. Then $B = -2 \cdot 2 + 3 \cdot 4 = 8$, $A = -6$, and computing $C = (8^2 - \Delta)/(4 \cdot -6) = -1$, for a composed form

$$(-2, 4, 3) \cdot (3, 2, -3) = (-6, 8, -1)$$

We can perform the same sorts of computations to get:

$$[(-2, 4, 3) \cdot (3, 2, -3)] \cdot (-3, 4, -2) = (-6, 8, -1) \cdot (-3, 4, -2) = (2, 8, 3)$$
$$(-2, 4, 3) \cdot [(3, 2, -3) \cdot (-3, 4, -2)] = (-2, 4, 3) \cdot (-1, 8, -6) = (2, 4, -3)$$

The operation is thus not associative, so the space of forms is not a group under this operation. However, by theorem 4.16 of Buell [6], this operator acts as an abelian group operation on the class group $\mathbf{F}/SL_2(\mathbb{Z})$, taking the sets of equivalent forms to each other. This operation has as an identity the class containing the principal form, called the *principal class*. One particularly useful fact is that classes of order 2, called *ambiguous classes*, all contain two reduced ambiguous forms.

In fact, as shown in [7], this operation forms a group on the infinite set $\mathbf{F}/\Gamma$, from which the useful concept of infrastructure distance will be extracted in section 2.2.4.

Even without explicitly forming a group on $\mathbf{R}$, equation 5.1 in [7] describes a relationship

$$F \cdot (G \cdot H) = \rho^n((F \cdot G) \cdot H)$$

With $|n|$ small. In a sense, then, this operation behaves similarly to a group operation on the space of reduced forms.

### 2.1.6 Square Forms and Square Roots

A *square form* is defined to be a form $(a, b, c)$ where $a = s^2$ is a square. For squarefree discriminant, the form must satisfy $\gcd(s, b) = 1$. These forms are notable in that they have a simple to construct *square root form* under composition; the form $(s, b, sa)$. This can be verified by composing it with itself; if $\gcd(s, b) = 1$, under the squaring operation, we get $(s^2, b-2as^2v, C)$ which is equivalent to $(s^2, b, c)$ by the matrix $\begin{pmatrix} 1 & av \\ 0 & 1 \end{pmatrix}$. When $\gcd(s, b) \neq 1$, these values will be off by the square of the common factor.

Note that if we take the square root of a element in the principal cycle, then the square root form will be in an *ambiguous class*, and thus the cycle contains a reduced ambiguous form. This is the mechanism by which both SQUFOF and SQUFOF2 construct their ambiguous forms.

## 2.2 Quadratic Order Correspondence

The language of quadratic forms is closely related with several other formulations. For the purposes of this paper, we will require facts about the "infrastructure distance" relying on correspondence between a quadratic form of discriminant $\Delta$ and certain subsets of $\mathbb{Q}[\sqrt{\Delta}]$, discussed in [7]. Here is where we find the need for $\Delta$ to be a fundamental discriminant in the complexity proof. As noted in chapter 5, this requirement does not hold in practice, and this correspondence is not used in the actual running of the algorithm.

### 2.2.1 Fundamental Discriminants

Recall the ring of integers for a quadratic extension by a squarefree integer $d$:

$$O_{\mathbb{Q}[\sqrt{d}]} = \begin{cases} \mathbb{Z}\left[\frac{1+\sqrt{d}}{2}\right] & d \equiv 1 \mod 4 \\ \mathbb{Z}[\sqrt{d}] & d \equiv 2, 3 \mod 4 \end{cases}$$

The discriminant of a quadratic extension is defined, for any integral basis $e_1, e_2$, and the nontrivial automorphism $\sigma : a + b\sqrt{d} \mapsto a - b\sqrt{d}$, as

$$\Delta = (e_1\sigma(e_2) - \sigma(e_1)e_2)^2$$

For $\mathbb{Q}[\sqrt{d}]$, this gives $\Delta = d$ in the first case and $\Delta = 4d$ in the second. If $\Delta$ is a possible discriminant of a quadratic extension, we say $\Delta$ is a *Fundamental Discriminant*; explicitly, $\Delta$ is a fundamental discriminant if either:

- $\Delta \equiv 1 \mod 4$, $\Delta$ a squarefree integer
- $\Delta = 4k$, $k \equiv 2, 3 \mod 4$, $k$ a squarefree integer

### 2.2.2 Quadratic Orders

A subset $A \subset O_{\mathbb{Q}[\sqrt{\Delta}]}$ is called an *Order*, or, *Quadratic Order*, if $A$ is a ring with identity 1 and field of fractions $O_{\mathbb{Q}[\sqrt{\Delta}]}$. Note that an order $A$ is finite index as an additive group in $O_{\mathbb{Q}[\sqrt{\Delta}]}$, with index $f \in \mathbb{Z}_{>0}$, and generators 1 and $f\frac{1+\sqrt{\Delta}}{2}$ if $\Delta \equiv 1 \mod 4$ or $f\sqrt{\Delta/4}$ otherwise. The concept of discriminant extends to orders, as well, with the discriminant of $A$ defined as

$$(1 \cdot \sigma(fe_2) - \sigma(1)fe_2)^2 = (\sigma(e_2) - e_2)^2 f^2 = f^2\Delta$$

Note that $f$ can be any positive integer. Therefore any number $\equiv 0, 1 \mod 4$ can be a discriminant of an order in a quadratic extension by its squarefree part.

### 2.2.3 Invertible Ideals

Suppose we have an invertible order $A$ of discriminant $\Delta$ in a quadratic field $K$. The product of two subsets $M, M'$ of $K$, written $M \cdot M'$, is the space of finite sums of elements $m \cdot m'$ with $m \in M, m' \in M'$. An *invertible $A$-ideal* is a subset $M \subset K$ with $A \cdot M = M$, and a subset $M' \subset K$ with $M \cdot M' = A$. As $K$ is commutative, note that $M'$ is also an invertible $A$-ideal. Note that $A$ is an invertible $A$-ideal, as $1 \in A$, and $A$ is its own inverse; in fact, $A$ is the identity under this product. The product of two invertible $A$-ideals is an invertible

$A$-ideal with inverse the product of the inverses, and so the space of invertible $A$-ideals forms a group under this set product. Denote this group $\mathbf{I}$. Equation 2.3 in [7] explicitly describes the invertible $A$-ideals for orders in quadratic fields as:

$$M = \left(\mathbb{Z} + \frac{b + \sqrt{\Delta}}{2a}\mathbb{Z}\right) \cdot \alpha$$

With $a, b \in \mathbb{Z}$, $a \neq 0$, $\alpha \in K^*$. Note that these choices are not unique for a given invertible ideal; $\alpha$ can be any element in a $\mathbb{Z}$-basis of $M$. For a given $\alpha$, we are given a choice of sign for $a$; for convention, choose $a$ of the same sign as $N(\alpha)$. Then $b$ is determined up to addition by a multiple of $2a$. Further, these $a, b$ satisfy $\Delta - b^2 \equiv 0 \mod 4a$, that is, there is a $c$ such that $(a, b, c)$ is a quadratic form of discriminant $\Delta$.

The freedom of choice of $b$ to be off by factors of $2a$ aligns with the action by $\Gamma$ on forms. In section 8 of [7], Lenstra establishes a surjection from choices of generator $\alpha$, represented by a coset of $\mathbb{Q}^*_{>0}$ in $K^*$, together with invertible ideals, to forms mod $\Gamma$:

$$I \otimes (K^*/\mathbb{Q}^*_{>0}) \twoheadrightarrow \mathbf{F}/\Gamma$$

Which is a group homomorphism under the operations of invertible ideal multiplication with kernel $\{(A\gamma, \gamma\mathbb{Q}^*_{>0}) | N(\gamma) > 0\} \simeq K^*_{>0}$.

For positive discriminant, this becomes an isomorphism

$$I/\mathbb{Q}^*_{>0} \otimes \{\pm 1\} \longleftrightarrow \mathbf{F}/\Gamma$$

Lenstra establishes this group structure coincides with composition, forming a group which maps into the ideal class group $\mathbf{F}/SL_2(\mathbb{Z}) \simeq \mathbf{I}/\{A\alpha | \alpha \in K^*\}$.

### 2.2.4 Infrastructure Distance

It is out of this finer structure that Lenstra defines his infrastructure distance, which is a slight modification of ideas presented by Shanks in [8]. The infrastructure distance is initially defined within the kernel of this map, $\{(A, \gamma\mathbb{Q}^*_{>0})K^*_{>0} | \gamma \in K^*\}$. Lenstra notes that two elements $(A, \gamma_1\mathbb{Q}^*_{>0})K^*_{>0}$ and $(A, \gamma_2\mathbb{Q}^*_{>0})K^*_{>0}$ of the kernel are equal if and only if

$\gamma_1 = \zeta\gamma_2$, for some $\zeta \in A^*$ with $N(\zeta) = 1$. Let $\nu \in A^*$ be the smallest such unit with $\nu > 1$; the *regulator* is defined as $R = \log \nu$. Lenstra defines the infrastructure distance as

$$d : \{(A, \gamma\mathbb{Q}^*_{>0})K^*_{>0} | \gamma \in K^*\} \to \mathbb{R}/R\mathbb{Z}$$
$$d((A, \gamma\mathbb{Q}^*_{>0})K^*_{N>0}) = \left(\frac{1}{2}\log\left|\frac{\gamma}{\sigma(\gamma)}\right| \mod R\right)$$

Notably, this distance, together with the sign of $a$, uniquely determines the form up to the action of $\Gamma$:

$$d((a,b,c)) = d((a',b',c')) \iff |a| = |a'| \text{ and } b \equiv b' \mod 2a$$

This distance, as a unary operator, is defined only on the principal cycle. To apply it within other cycles, Lenstra defines the binary operator:

$$d(f,g) = d(gf^{-1})$$

Note that this is only defined for $f, g$ in the same cycle, as that is precisely when $gf^{-1}$ will be in the principal cycle.

In section 11 of [7], Lenstra notes that the reduced forms in each cycle, traversed by the reduction operator from a starting form $f$, occur in increasing order of distance $d(f, g)$ until cycling back to the first form, with $d(f, f) = R \equiv 0 \mod R$. This fact allows us to confidently search for a reduced form in a cycle, within a certain distance of a starting form, by iterating the reduction operator. This is a key operation in both the original SQUFOF and in SQUFOF2.

We will need, for estimating the time complexity of SQUFOF2, an estimate for the number of reduction steps between the reduction of a computed inverse square root form and the ambiguous forms in its cycle, where these forms are given as quadratic forms $(a, b, c)$. Lenstra gives an explicit formula for the distance when reducing forms:

**Theorem 2.2.1.** *For an indefinite integral binary quadratic form of discriminant* $\Delta$

$$F(X,Y) = aX^2 + bXY + cY^2$$

*The infrastructure distance between* $F$ *and* $\rho(F)$ *is given by*

$$d(F,\rho(F)) = \frac{1}{2}\log\left|\frac{b+\sqrt{\Delta}}{b-\sqrt{\Delta}}\right|$$

For a proof, see section 11 of [7]. This explicit formula for the infrastructure distance leads to an interesting corollary, particularly in the context of the original SQUFOF:

**Corollary 2.2.2.** *For all reduced forms* $F = (a',b',c') \in \mathbf{R}$*, and principal form* $F_1 = (1,b,c)$*, the distance* $d(F,\rho(F)) \leq d(F_1,\rho(F_1))$*, with equality if and only if* $b' = b$*.*

*Proof.* By the discriminant formula, $\Delta = b'^2 - 4a'c'$, and so $b' \equiv \Delta \mod 2$. The principal form is constructed as a reduced form with $a = 1$, and thus $|\sqrt{\Delta} - 2| < b < \sqrt{\Delta}$, and so $b$ is uniquely determined. For any reduced $F$, $b'$ has the same parity requirement and upper bound, and so we have $b' \leq b$, and $b' - \sqrt{\Delta} \leq b - \sqrt{\Delta} < 0$, giving

$$d(F,\rho(F)) = \frac{1}{2}\log\left|\frac{b'+\sqrt{\Delta}}{b'-\sqrt{\Delta}}\right| \leq \frac{1}{2}\log\left|\frac{b'+\sqrt{\Delta}}{b-\sqrt{\Delta}}\right| \leq \frac{1}{2}\log\left|\frac{b+\sqrt{\Delta}}{b-\sqrt{\Delta}}\right| = d(F_1,\rho(F_1)) \qquad \square$$

Heuristically, Lenstra estimates the infrastructure distance between two adjacent forms on a cycle at on average $\pi^2/(12\log 2) \approx 1.18$, and so the infrastructure distance adjusted by this factor will be a good estimate for the number of steps along the reduction cycle.

In the algorithm, we construct the distance from the identity form $F_1$ to a square form $F$. For that, we will need the distance of a matrix action:

**Theorem 2.2.3.** *For an indefinite integral binary quadratic form of discriminant* $\Delta$

$$F(X,Y) = aX^2 + bXY + cY^2$$

*and matrix*

$$S = \begin{pmatrix} x & z \\ y & w \end{pmatrix} \in SL_2(\mathbb{Z})$$

*The infrastructure distance between $F$ and $S \cdot F$ is*

$$d(F, S \cdot F) = \frac{1}{2} \log \left| \frac{2ax + y(b + \sqrt{\Delta})}{2ax + y(b - \sqrt{\Delta})} \right| \mod R$$

*Where $R$ is the regulator of $\mathbb{Q}[\sqrt{\Delta}]$.*

*Proof.* Write $F'(X, Y) = S \cdot F = a'X^2 + b'XY + c'Y^2$. To $F, F'$ we can write representatives for the invertible ideals corresponding to them:

$$M = \left( \mathbb{Z} + \frac{b + \sqrt{\Delta}}{2a} \mathbb{Z} \right) \qquad M' = \left( \mathbb{Z} + \frac{b' + \sqrt{\Delta}}{2a'} \mathbb{Z} \right)$$

We can use the substitution $F(xX + zY, yX + wY) = F'(X, Y)$ to verify that $\gamma = (2ax + y(b - \sqrt{\Delta}))/(2a') \in \mathbb{Q}[\sqrt{\Delta}]$ satisfies $M' = \gamma M$. Letting $X$ and $Y$ stand in for arbitrary integers:

$$\begin{aligned}
\gamma M &= \frac{\left(2ax + y(b - \sqrt{\Delta})\right)}{2a'} \cdot M \\
&= \frac{\left(2ax + y(b - \sqrt{\Delta})\right) \left((xX + zY) + \frac{b + \sqrt{\Delta}}{2a}(yX + wY)\right)}{2a'} \\
&= \frac{2 \underbrace{(ax^2 + bxy + cy^2)}_{=F(x,y)=a'} X + \left( \underbrace{2(axz + cwy) + b(wx + yz)}_{=b'} + \underbrace{(xw - yz)}_{=1} \sqrt{\Delta} \right) Y}{2a'} \\
&= \left( X + \frac{b' + \sqrt{\Delta}}{2a'} Y \right) = M'
\end{aligned}$$

This allows calculation of the infrastructure distance using the definition:

$$\begin{aligned}
d(F, S \cdot F) &= d((M, 1), (\gamma M, 1)) = d((M, 1), (M, \sigma(\gamma))) \\
&= d((M, \sigma(\gamma)) \cdot (M^{-1}, 1)) = d((A, \sigma(\gamma))) \\
&= \frac{1}{2} \log \left| \frac{\sigma(\gamma)}{\gamma} \right| = \frac{1}{2} \log \left| \frac{2ax + y(b + \sqrt{\Delta})}{2ax + y(b - \sqrt{\Delta})} \right| \mod R
\end{aligned}$$

□

**Corollary 2.2.4.** *For an indefinite integral binary quadratic form of discriminant* $\Delta$

$$F(X,Y) = aX^2 + bXY + cY^2$$

*and primitive representation* $F(x,y) = v$*, the infrastructure distance between* $F$ *and the form* $G$ *with leading coefficient* $v$*, constructed as in section 2.1.3, is*

$$d(F,G) = \frac{1}{2}\log\left|\frac{2ax + y(b+\sqrt{\Delta})}{2ax + y(b-\sqrt{\Delta})}\right| \mod R$$

*Where* $R$ *is the regulator of* $\mathbb{Q}[\sqrt{\Delta}]$*.*

*Proof.* The matrix for this transformation is given as $\begin{pmatrix} x & z \\ y & w \end{pmatrix}$, for some $w, z$ with $xw - yz = 1$, which exist as $F(x,y)$ is a primitive representation of $v$. The distance follows from theorem 2.2.3. □

The form produced by the matrix action is not on the reduced cycle, so we will also need a bound on the distance to the nearby reduced form $\rho_0(g)$. A suitable bound is discussed in section 12 of [7], which states that the reduction of a form is one of the forms closest in infrastructure distance above or below with the same $a$ sign, or the form with opposite $a$ sign between them. This means the reduction adds at most two steps along the cycle over what would be expected from infrastructure distance before reduction.

## 2.3 Other Quadratic Form Factoring Algorithms

There have been a few factoring algorithms explicitly relying on the construction of ambiguous forms. Two relevant algorithms are the original SQUFOF, and and algorithm due to Lenstra and Pomerance in [4].

### 2.3.1 SQUFOF

We now have everything we need in order to discuss the original SQUFOF algorithm, which in its simplest form consists of two searches by reduction. For an in depth analysis of SQUFOF, see [3].

To factor a squarefree composite positive integer $N$, SQUFOF begins by initializing the principal form $F_1$ of fundamental discriminant $\Delta = 4N$ if $N \equiv 2, 3 \mod 4$, and $\Delta = N$ if $N \equiv 1 \mod 4$. SQUFOF then traverses the principal cycle of forms from $F_1$ by reduction until it finds a square form $F_n = (u, v, w^2)$. Such a form occurs after $O(N^{1/4})$ reduction steps.

SQUFOF then considers the ambiguous cycle containing the inverse square root form $(-w, v, -uw)$. Traversing this cycle finds an ambiguous form, and thus a divisor of $N$, in roughly half as many steps as traversed in the principal cycle. In the simplest form, subject to some reasonable assumptions, this factor is nontrivial for at least $1/2$ of the square forms. This can be improved; through careful caching of intermediate results in the initial search, SQUFOF can select only square forms which lead to nontrivial factors by avoiding square forms which have ambiguous form on the principal cycle.

One notable advantage of SQUFOF is that, in its simplest form, it only needs to store the integer to be factored, and up to two reduced quadratic forms with three coefficients of size $< 2\sqrt{N}$. This space is negligible compared to many other factoring algorithms, however its time complexity is $O(N^{1/4})$, the estimated distance to the first square form leading to a factor, makes it less useful in practice.

To illustrate SQUFOF, we will give a brief example. Consider factoring the integer $N = 4187$. We first construct the identity form $F_1$. $\sqrt{N} \approx 64.71$, and so $b = 2\lfloor 64.71 \rfloor = 128$. $c$ is

computed according to the discriminant formula, $c = (128^2 - 4187 \cdot 4)/4 = 64^2 - 4187 = -91$. We begin walking the principal cycle by reduction until we find a square form:

| | $a$ | $b$ | $c$ |
|---|---|---|---|
| $F_1$ | 1 | 128 | $-91$ |
| $F_2$ | $-91$ | 54 | 38 |
| $F_3$ | 38 | 98 | $-47$ |
| $F_4$ | $-47$ | 90 | 46 |
| $F_5$ | 46 | 94 | $-43$ |
| $F_6$ | $-43$ | 78 | 62 |
| $F_7$ | 62 | 46 | $-59$ |
| $F_8$ | $-59$ | 72 | 49 |

After 7 reduction steps, we find the form $F_8 = (-59, 72, 7^2)$. We convert this to the inverse square root form $(-7, 72, -7 \cdot -59) = (-7, 72, 413)$. This form is not reduced, and so we reduce it in two reduction steps to the form $(-7, 128, 13)$. We can refer to this form as $G_0$, and walk this ambiguous cycle of reduced forms from $G_0$ until we reach the symmetry point:

| | $a$ | $b$ | $c$ |
|---|---|---|---|
| $F^{-1}$ | $-7$ | 72 | 413 |
| $G_0$ | $-7$ | 128 | 13 |
| $G_1$ | 13 | 106 | $-106$ |
| $G_2$ | $-106$ | 106 | 13 |

After roughly half as many steps as the square form in the principal cycle, we find an ambiguous form $(-106, 106, 13)$. Taking $\gcd(106, N) = 53$, we factor $N = 53 \cdot 79$.

### 2.3.2 Lenstra and Pomerance's Algorithm

Similar to both SQUFOF and SQUFOF2, Lenstra and Pomerance's algorithm in [4] constructs forms of discriminant a multiple of $N$, and constructs a form with a coefficient

dividing the discriminant. However, the structure of positive definite forms is quite different. SQUFOF and SQUFOF2 both rely on cycles of equivalent reduced forms. The definite reduced forms, however, are unique up to equivalence, and are isomorphic to ideal class group under composition.

Lenstra and Pomerance's algorithm constructs and solves a matrix of factorizations over a factor base, as does SQUFOF2. However, where SQUFOF2 is factoring values of a particular quadratic form at points in a region, Lenstra and Pomerance factor randomly chosen reduced forms themselves, expressed as a product of generators of the class group. Both algorithms solve these systems over $GF_2$, and obtain a square root of a form, and thus an ambiguous class. For Lenstra and Pomerance, an ambiguous class is represented by an ambiguous form, and they can upon construction obtain their divisor of $\Delta$ and check if it is a nontrivial divisor of $N$. However, SQUFOF2 operates in the space of indefinite forms, and has the additional step of traversing the ambiguous cycle in search of a reduced ambiguous form. This search, however, is reliably quite short.

Both algorithms have the same heuristic complexity, and each have their own advantages. SQUFOF2, notably, can completely avoid explicit computations in the class group, nearly all of the arithmetic of quadratic forms in general, and can use the efficient polynomial sieving methods for sieving values.

# 3. SQUFOF2

The main work in the original SQUFOF is in the iterative search for the first square form in the principal sequence, found at roughly $O(N^{1/4})$ reduction steps. This makes SQUFOF an $O(N^{1/4})$ algorithm.

However, Shanks wrote in [2] that:

> Gauss [9] proved that *any* form $F$ in the principal genus has a square root $f$ such that $f \cdot f$ (under composition and reduction) $= F$. His constructive proof gives a remarkable algorithm for computing $f$.
>
> The $w^2$ in $(u, v, w^2)$ is the *value* of a form with $x = 0$ and $y = 1$. If, for any $F_n$, its value is a square for certain values of $x$ and $y$, then one can easily construct an equivalent form (probably not reduced) that is a square form.
>
> This variation on SQUFOF was suggested by R. de Vogelaire when I first spoke on SQUFOF in [10]. He calls it the "fat" SQUFOF. One tries small pairs $(x, y)$ in $F_n$ to see if it has a small square value.

## 3.1 Algorithm Description

This method from Gauss provides an efficient way to transform a square value into a square form, suggesting a perhaps faster path to a solution. In SQUFOF2, we will construct a rather large square value at the principal form $F_1$ of discriminant $4N$ if $N \equiv 2, 3 \mod 4$ and $N$ if $N \equiv 1 \mod 4$. A square value of any form in the principal cycle would work, but using the principal form itself simplifies many of the computations; under composition of forms, $F_1 \circ F_1 = F_1$, considerably simplifying the arithmetic we will have to perform and shortening the distance between the inverse square root and ambiguous form. In addition, having the first coefficient 1 simplifies a few of the equations as well.

We will construct this square value out of $B$-*smooth* values at points $(x, y)$, that is, values which factor into primes no larger than $B$, for a bound $B$ tuned to the problem as described in section 4.3.8. We will also require $\gcd(x, y) = 1$. We will also define a *factor base*, that is, a set of primes together with $-1$, over which the $B$-smooth values factor. In this case,

we will use all primes less than $B$ which may occur as factors of $F_1$, which is in general not all the primes less than $B$; note that a prime $p$ can divide $F_1(x,y) = x^2 + bxy + cy^2$, with $\gcd(x,y) = 1$, only if the discriminant $D$ is a quadratic residue mod $p$ as by the quadratic formula, any solution $(x,y)$ must satisfy $2x \equiv y(-b \pm r) \mod p$ where $r^2 \equiv D \mod p$. We take as our factor base $-1$ together with all primes $p \leq B$ for which $1 = \left(\frac{D}{p}\right)$. Note that in either case $\left(\frac{D}{p}\right) = \left(\frac{N}{p}\right)$.

This same quadratic formula allows us to sieve for smooth values efficiently. We fix some bound $S$ on the largest $x, y$-values sieved, tuned to the problem as with $B$, and at each $y$ value, starting from 1, we perform a sieve for smooth values. To perform this sieve, initialize an array of zeroes of length $2S+1$ representing the $x$-values in $[-S, S]$. For each odd prime in the factor base, $p|f(x,y) \iff 2x \equiv y(-b \pm r) \mod p$ where $r^2 \equiv D \mod p$. For a given $N$ and $y$, the right hand side of this equation is fixed, and gives two $x$ values at regular intervals of length $p$. This makes it relatively simple to, for each $p$, at each possible $x$ value, add $\log(p)$ to the sieve vector at that position.

At the end of the sieving process, then, for all squarefree $B$-smooth values $F_1(x,y)$ with $\gcd(x,y) = 1$, the $x$th entry in the array will be:

$$\sum_{p<B, p|F_1(x,y)} \log(p) = \log\left(\prod_{p<B, p|F_1(x,y)} p\right) = \log(|F_1(x,y)|)$$

In practice, these computations can be done using floating point arithmetic, with a small tolerance, allowing us to quickly check which values are likely to be $B$-smooth. We also filter out any values with $\gcd(x,y) > 1$. We can then verify the sieved values by trial division over our factor base, storing these factorizations for use in the next step. This trial division catches repeated prime powers missed by the simple version of the sieve.

The algorithm identifies collections of these smooth values which have a square product. To do this, for each smooth value $F_1(x,y)$ we construct a vector over $GF(2)$, of length the size of the factor base, with a 1 in the $i$th position whenever the $i$th prime divides $F_1(x,y)$ to an odd power. In this way, a vector of all 0s represents a square $B$-smooth value. We can add these vectors, with sums of vectors representing the products of the corresponding values, and be guaranteed at least one solution $vA = 0$ so long as we have more "rows" than

"columns", that is, at least as many smooth values as there are primes in our factor base. SQUFOF2 can fail to find a proper factor from a square value, with estimated probability less than 1/2 as discussed in 2.1.4, so we may need a small number of different square values. We construct a matrix of these rows, sieving until we have a few more rows than columns, and compute the left null space of the matrix using techniques of linear algebra.

A solution to this system corresponds to a list of $(x, y)$ pairs with product $\prod F_1(x, y)$ a square. We can reduce this list of $(x, y)$-values to a single point on $F_1$ with square value using composition of forms. Such a constructed pair can be fairly large compared to the original bounds, and is likely not reduced. From this square value, by the technique of Gauss mentioned above, we convert to a corresponding inverse square root form. This form will also have large coefficients, and the next step is to apply the reduction operator until we get a reduced form. As we will see in 4.3.7, a square form with large coefficients will reduce to a form very close on the cycle to an ambiguous form, and thus a factor of $N$. If this factor is not proper, we simply select another solution to the linear system and try again until successful. As each attempt has estimated probability of finding a factor at least 1/2, we will typically only need to do this process a small number of times before successfully factoring $N$.

## 3.2 Examples

We will demonstrate the algorithm with two examples.

### 3.2.1 A Small Example of SQUFOF2

Let us first consider factoring the integer $N = 4819$, with a smoothness bound of 20, and a bound on the sieve of 20. First we compute the identity form $F_1$ of discriminant $\Delta = 4 \cdot 4819 = 19276$. $\sqrt{N} \approx 69.42$, so $b = 2\lfloor 69.42 \rfloor = 138$. By the discriminant formula, $c = \dfrac{b^2 - \Delta}{4} = -58$, and $F_1 = (1, 138, -58)$. Next, we build our factor base of primes with $\left(\frac{N}{p}\right) = 1$, together with $-1$, getting the set $\{-1, 2, 3, 5, 11, 13, 17\}$. Sieving, we get the values:

$$F_1(-2, 1) = -330 = -1 \cdot 2 \cdot 3 \cdot 5 \cdot 11$$
$$F_1(4, 1) = 510 = 2 \cdot 3 \cdot 5 \cdot 17$$
$$F_1(1, 2) = 45 = 3^2 \cdot 5$$
$$F_1(-1, 3) = -935 = -1 \cdot 5 \cdot 11 \cdot 17$$

The product of these values is the square $(2 \cdot 3 \cdot 5^2 \cdot 11 \cdot 17)^2$. We compose $(-2, 1)$ and $(4, 1)$, using the formula described in section 2.1.5:

$$\begin{pmatrix} -2 \cdot 4 - (-58) \cdot 1 \cdot 1 \\ -2 \cdot 1 + 4 \cdot 1 + 138 \cdot 1 \cdot 1 \end{pmatrix} = \begin{pmatrix} 50 \\ 140 \end{pmatrix}$$

Note that the representation of the product

$$F_1(50, 140) = -168300 = -1 \cdot 2^2 \cdot 3^2 \cdot 5^2 \cdot 11 \cdot 17$$

is no longer primitive, and we can divide the pair by $\gcd(50, 140) = 10$, dividing the value by $10^2$:

$$F_1(5, 14) = -1683 = -1 \cdot 3^2 \cdot 11 \cdot 17$$

We can compose the other smooth values similarly:

$$F_1(5,14) \cdot F_1(1,2) = F_1(1629,3888) = 9^2 F_1(181,432)$$
$$F_1(181,432) \cdot F_1(-1,3) = F_1(74987,178959) = 187^2 F_1(401,957)$$
$$f(401,957) = 25 = 5^2$$

giving a final square value 25 primitively represented by $(401, 957)$. We calculate the extended greatest common divisor to get coefficients $284 \cdot 401 - 119 \cdot 957 = 1$, and form the matrix

$$\begin{pmatrix} 401 & 119 \\ 957 & 284 \end{pmatrix} \in SL_2(\mathbb{Z})$$

whose action takes $F_1$ to the square form $(25, -124, -39)$, which we can convert to the inverse square form $(195, -124, -5)$. We apply the reduction operator to get the reduced form $(-5, 134, 66)$. Another application of the reduction operator takes us to $(66, 130, -9)$, and then $(-9, 122, 122)$, and finally $(122, 122, -9)$, where we find the symmetry point. Taking $\gcd(122, 4819) = 61$, we factor $4819 = 61 \cdot 79$.

### 3.2.2 A Larger Example of SQUFOF2

For our second example of SQUFOF2, we will factor $N = 72224443$ with the parameters (described in chapter 4) $\alpha = 0.7$, $\beta = 0.8$, giving a smoothness bound of 158 and sieve bound of 327.

Computing the identity form $F_1$, we have $\sqrt{N} \approx 8498.5$, so $b = 2\lfloor 8498.5 \rfloor = 16996$. By the discriminant formula, $c = -8439$. There are 24 primes in the factor base, the numbers $-1, 2, 3, 7, 11, ..., 113, 137, 149, 151, 157$.

We begin the sieve at $y = 1$, and find 40 smooth values. We can perform elimination on the matrix, already finding the solution

$$\{(-198, 1), (-179, 1), (-93, 1), (-84, 1)\}$$

We begin composing, dividing by any nontrivial common divisor:

$$\begin{aligned}
F_1(-198,1)\cdot F_1(-179,1) &= F_1(43881,16619)\\
F_1(43881,16619)\cdot F_1(-93,1) &= F_1(136166808,280954838) = 8362^2\cdot F_1(16284,33599)\\
F_1(16284,33599)\cdot F_1(-84,1) &= F_1(282174105,568242572) = 8357^2\cdot F_1(33765,67996)\\
F_1(33765,67996) &= 4655196441 = 68229^2
\end{aligned}$$

Now we compute the inverse square root form $(-1791814782933, -699296020, -68229)$. This form reduces, in just 4 steps, to the form $(8439, 16996, -1)$. This form is ambiguous, with the next form in the cycle $(-1, 16996, 8439)$. Unfortunately, $\gcd(-1, N) = 1$, so this solution to the linear system led to a trivial factor.

When we find a trivial factor, we backtrack to the linear algebra and find another solution to the linear system. Composing the smooth values at

$$\{(-315,1),(-292,1),(-222,1),(-179,1),(-118,1),(-39,1),(-21,1),(-8,1)\}$$

Gives a square value of $228355535538459^2$, and an inverse square root form

$$\begin{aligned}
(-153281639804923455737174027255794344354639 9,&\\
37418022901199079712249857344,&\\
-228355535538459)&
\end{aligned}$$

Which reduces in 10 reduction steps to $(-5439, 15158, 2718)$. After 9 reduction steps along the ambiguous cycle, we reach the symmetry point $(-15362, 15362, 861)$, with $\gcd(-16362, N) = 7681$ and we have factored $N = 7681 \cdot 9403$.

# 4. SQUFOF2 COMPLEXITY PROOF

We now will provide a heuristic proof of the algorithm's running time, subject to a few reasonable assumptions.

We will present this proof in a few sections. First we will establish the complexities of the various operations involved in the algorithm, and a convention to keep the equations simple. Next we will establish some reasonable assumptions needed for the proof to work. Then we will give an analysis of each broad step to establish some common bounds in terms of $L(N)^{1+o(1)}$, the parameter $\alpha$ which determines the size of the factor base, the parameter $\beta$ which determines the size of the sieve, and an *elimination exponent* $r \in (2, 3]$ representing the complexity of the chosen algorithm for solving a $m \times m$ linear system over $GF(2)$ in $O(m^r)$ time. We will then solve for theoretical optimal parameter choices in terms of $r$ for very large $N$, and finally arrive at the overall time and space complexity of the algorithm given existing elimination methods.

## 4.1 Preliminaries

We will make extensive use of the "unusual convention" $L = L(N)^{1+o(1)}$ from Pomerance [1] , where $L(N) = \exp(\sqrt{\log N \log\log N})$, to greatly simplify the presentation and analysis. This convention allows us to write some useful statements which at first glance seem absurd:

$$
\begin{aligned}
kL &= e^{\sqrt{\log(N)\log\log(N)}(1+o(1))+\log(k)} = L \\
\log(N)^k L &= e^{k\log\log(N)} e^{\sqrt{\log(N)\log\log(N)}(1+o(1))} \\
&= e^{\sqrt{\log(N)\log\log(N)}\left(1+o(1)+k\sqrt{\frac{\log\log(N)}{\log(N)}}\right)} \\
&= e^{\sqrt{\log(N)\log\log(N)}(1+o(1))} = L \\
\log\log(N)^k L &= e^{\sqrt{\log(N)\log\log(N)}\left(1+o(1)+k\sqrt{\frac{\log\log\log(N)}{\log(N)}}\right)} = L \\
L\log(L) &= L\sqrt{\log(N)\log\log(N)} = L \\
\pi(L) &\approx \frac{L}{\log(L)} = L
\end{aligned}
$$

As a result, we are able write and compare the estimates in terms of only powers of $L$.

We will need to perform a few standard operations on integers. Addition or subtraction of two $k$-digit integers is $O(k)$.

We will need to multiply pairs of $k$-digit integers, the time complexity of which is denoted $M(k)$. While traditional methods take $O(k^2)$ time, there are techniques such as Schönhage-Strassen, Discussed in section 9.5 of Crandall and Pomerance [11], which perform multiplication in $M(k) = O(k \log k \log\log k)$. Notably, if $k = L^\alpha$, this takes $M(L^\alpha) = L^\alpha \log L^\alpha \log\log L^\alpha = L^\alpha$ time, by the same convention for $L$. During the composition and reduction in the algorithm, integers of such size may be constructed, and these fast techniques will help keep the computation time bounded by the difficulty of the sieve.

We will need to divide and take square roots, and take remainders from division; the same chapter of [11] provides algorithms for these in $O(M(k))$ time. Similarly, the chapter provides an algorithm due to Stehlè and Zimmermann for taking the greatest common divisor of two $k$-digit numbers in $O(M(k)\log(k))$ time, which as noted for integers of size $L^\alpha$ is $O(L^\alpha)$ time.

We will, in analyzing the factor base, need to compute quadratic residues mod $p$ for primes less than the bound $L^\alpha$. For this analysis the naive method of testing each $1 < r < p/2$, taking time bounded by $L^\alpha M(\log(L^\alpha))) = L^\alpha$, suffices. See 5 for some discussion of an algorithm to use in practice.

We have discussed two parameters, $\alpha, \beta \in (.1, 1)$, used as exponents of $L$ in constructing the bound on the factor base $B = L^\alpha$ and on the size of the $x, y$ ranges sieved $L^\beta$. For the course of this discussion, we will be using those parameters to estimate each step, and selecting a theoretical optimal value at the end.

### 4.2 Assumptions

As noted above, the proof is heuristic, and relies on a few assumptions. First, we will require a lower bound on the proportion of primes in the factor base. One can construct $N$, via the Chinese Remainder Theorem, which are not quadratic residues mod $p$ for any $p < L(N)^\alpha$ for a small $\alpha$, giving a factor base consisting of only $-1$ and 2, making the sieve very difficult. Such examples are rare, as the expected portion of $p < L^\alpha$ for which $\left(\frac{N}{p}\right) = 1$

is 1/2, and so the expected number of primes $p$ in the factor base is $\pi(L^\alpha)/2$. To enable the proof, we assume for $\alpha$ not too small that we have enough of the primes in the factor base:

**Hypothesis 1.** *There is a constant $n_1$ such that if $N > n_1$, then for any $\alpha \in (0.1, 1)$ the number of primes $p < L^\alpha$ for which $p \nmid N$ and $(N/p) = +1$ is at least $\pi(L^\alpha)/3$.*

To analyze the sieve, we will need some idea of how many of the $x, y$ values we sieve will be smooth. For this, we rely on another assumption:

**Hypothesis 2.** *There is a constant $n_2$ such that if $N > n_2$, then for any $\alpha \in (0.1, 1)$ and any $\beta \in (0.1, 1)$ the values $|F_1(x, y)|$ with $-L^\beta < x < L^\beta$, $0 < y < L^\beta$ and $\gcd(x, y) = 1$ have the same probability of being $L^\alpha$-smooth as all integers in $(1, \max |F_1(x, y)|)$.*

Note that, as we are discarding points with $\gcd(x, y) \neq 1$, as they provide the same factorization mod 2 as the earlier value $F_1(x/\gcd(x, y), y/\gcd(x, y)) = F_1(x, y)/\gcd(x, y)^2$ which will all ready be recorded, we include that in the assumption. This is reasonable, as when $\gcd(x, y) = 1$, there is no reason to expect $|F_1(x, y)|$ to be different in factorization from general numbers of this size. However, when $\gcd(x, y) > 1$, we have noted that $F_1$ will be guaranteed to have square divisors, and thus likely more prime factors than a typical number of the same size.

We have noted that, for all $x, y$ with $\gcd(x, y) = 1$, then no prime with $\left(\frac{N}{p}\right) = -1$ can divide $F_1(x, y)$. However, this is less of a concern, as we have also noted that for a given $y$, primes with $\left(\frac{N}{p}\right) = 1$ give two solutions $x$ to the equation $F_1(x, y) \equiv 0 \mod p$, and these two cases occur in roughly equal proportion, by assumption 1. The case $p|N$ corresponds to only one solution, $2x \equiv -b \mod p$, though for the algorithm this would be sufficient to return a factor of $N$.

We will need the proportion of integers with $\gcd(x, y) = 1$, as well, for which we will use the following lemma:

**Lemma 1.** *Let $m$ be a large integer. Let $G(m)$ be the number of pairs $(x, y)$ of integers with $1 \leq x \leq m$, $1 \leq y \leq m$ and $\gcd(x, y) = 1$. Then $G(m) = (6/\pi^2)m^2 + O(m \log m)$ as $m \to \infty$.*

*Proof.* Note that there is only one pair with $x = y$ and $\gcd(x, y) = 1$, the pair $(1, 1)$. Next we will count the pairs with $x < y$; there will be the same number of pairs with $y < x$.

For a given $y \in [2, m]$, the number of $x$-values with $x \leq y, \gcd(x, y) = 1$ is given by Euler's totient function $\phi(y)$, so the total number of pairs is $\sum_{y=2}^{m} \phi(y) = (3/\pi^2)m^2 + O(m \log m)$ by Theorem 330 of [12] or Theorem 3.7 of [13].

Accounting for pairs with $y < x$, we double the sum to obtain $\frac{6}{\pi^2}m^2 + O(m \log m)$. □

The infrastructure distance to the computed square form, given by theorem 2.2.3, may be quite large when the denominator $2ax + y(b - \sqrt{\Delta})$ is close to 0. This is unlikely in practice, as $x, y$ are not chosen for their closeness to solutions to $F_1$ such as the denominator; in fact, $F_1(x, y) = s^2$ is in the algorithm typically much larger than 0. To formalize this intuition in lieu of a more rigorous justification, we assume:

**Hypothesis 3.** *For pairs $(x, y)$ with square value $F_1(x, y)$ computed as in SQUFOFII, at least $\frac{1}{2}$ of the pairs satisfy $|2ax + y(b - \sqrt{\Delta})| > 1$.*

We will need one last assumption. We discussed in section 2.1.4 the set of reduced ambiguous forms, and how each leads to a factor of $D$, with at least half leading to a proper factor. We will assume, for our purposes, that each of these is equally likely to be the one found in the execution of SQUFOF.

**Hypothesis 4.** *Each of the reduced ambiguous forms of the fundamental discriminant $\Delta$ has an equal chance of being the one at the symmetry point in the execution of SQUFOF2.*

This is similar to 4.19 in [3], and reasonable, as the square roots of forms on the principal cycle can appear in any ambiguous cycle.

### 4.3 Overview of the Algorithm Complexity

We will now give an analysis for a given squarefree odd composite $N$ with no prime factors less than $L(N)^\alpha = \exp\left(\alpha\sqrt{\log(N)\log\log(N)}\right)$.

### 4.3.1 Construction of the Form

We begin the algorithm by constructing the identity form of discriminant $\Delta = 4N$ if $N \equiv 3 \mod 4$, and $\Delta = N$ if $N \equiv 1 \mod 4$. This form $F_1 = (1, b, c)$ is constructed slightly differently in each case. In the case $N \equiv 3 \mod 4$, then $\Delta = 4N$, and the form has $b$ chosen to be the largest *even* number less than $\sqrt{\Delta}$, that is, $b = 2\left\lfloor\sqrt{N}\right\rfloor$, with $c$ computed according to the formula for the discriminant, $c = (D - b^2)/4 = N - \left\lfloor\sqrt{N}\right\rfloor^2$.

Alternately, the case $N \equiv 1 \mod 4$, we have $\Delta = N$, we choose $b$ as the first *odd* number less than $\sqrt{D} = \sqrt{N}$, that is, $2\left\lfloor\frac{\sqrt{N}-1}{2}\right\rfloor + 1$. The proof for the two cases differ only slightly, but as the expressions are slightly simpler for the case $N \equiv 3 \mod 4$, we will demonstrate on that case. Interestingly, as we will note in chapter 5, in practice it suffices to use this discriminant for all odd non-square composite $N$.

These coefficients are of size less than $2\sqrt{N}$, and so these initial computations take $O(M(\log(N)))$ time to take the integer square root of N, square, and multiply.

### 4.3.2 Initialization of the Factor Base

The next step is preparing the factor base. It consists of $-1$, together with the primes $p < L^\alpha$ such that $\left(\frac{N}{p}\right) = 1$, starting with 2, as discussed in 3. It will be useful to store, alongside each odd prime in the factor base, a quadratic residue for the prime for use in the sieve.

As discussed above, each step takes $L^\alpha$ time, takes $\pi(L^\alpha)L^\alpha = L^{2\alpha}$ time. Storing these takes $L^\alpha$ space.

### 4.3.3 Constructing and Performing the Sieves

We next perform a sieve for each $y$ in $(0, L^\beta)$. Fixing $y$, we initialize a list for $x$-values in the interval $(L^\beta, L^\beta)$, each with value 0. For each prime $p$ in the factor base, we compute the $x$ values at which $p|F_1(x, y)$ by the quadratic formula discussed above, that is, $x \equiv -b/2 \pm r \mod N$ where $r^2 \equiv N \mod p$. At each such position, we know $p|F_1(x, y)$, and so we add $\log(p)$ at the value in the position at that $x$. If the entry at that position is close to

$\log(|F_1(x,y)|)$, then we know that $F_1(x,y)$ is likely $L^\alpha$-smooth. We check the smoothness by trial division over the factor base, which catches the repeated prime powers missed by the algorithm. We can store the factorizations, paired with the points $(x,y)$, for the next step.

This will take $L^\beta$ evaluations of $F_1$ at each $y$, for $L^{2\beta}$ complexity in initialization of the rectangle. It will also take an addition at each of the $L^\alpha$ primes in the factor base to find the roots, as well as $L^\beta$ divisions of complexity $M(\log(L^\alpha))$ which is absorbed into the powers of $L$ by the convention, giving a time complexity of $L^\alpha L^\beta$ for performing the sieve, with an overall complexity of $L^{\max\{\alpha\beta,2\beta\}}$, bounded by $L^{\max\{2\alpha,2\beta\}}$, for the full sieve.

### 4.3.4 Linear Algebra

We can stop the sieve prematurely when we have a few more solutions than the size of the factor base. We convert these factorizations into rows over $GF(2)$ of length the size of the factor base $L^\alpha$. At each position in the row we place a 1 if $p$ divides $F_1(x,y)$ to an odd power, and a 0 if not.

We now have a matrix that is nearly square, with a number of columns equal to the size of the factor base, $L^\alpha$. Our goal is to obtain subsets of the columns where the product of the values is square, that is, elements of the left null space of the matrix over $GF(2)$. This problem can be solved via Gaussian Elimination in $O((L^\alpha)^3) = O(L^{3\alpha})$ time; other elimination methods solve the problem in time $O((L^\alpha)^r)$ for some elimination exponent $r \in (2,3]$. One of the best general elimination methods available at time of writing is the Coppersmith-Winograd method [14] with an exponent of roughly 2.49. For the purpose of this algorithm, we will write the complexity of solving the system as $O(L^{r\alpha})$, and discuss the effect of the elimination exponent in more detail.

### 4.3.5 Assembling the Square Form

Each solution $S = \{(x_0,y_0),\ldots,(x_n,y_n)\}$ represents a list of $(x,y)$ pairs for which the product of values $\prod_{(x,y)\in S} F_1(x,y)$ is a square. The next task is turning this into a single $x,y$ pair with a square value, by the techniques discussed in section 2.1.5. Starting with $(x,y) = (x_0,y_0)$, each composition consists of replacing $(x,y)$ with $(xx_i - cyy_i, xy_i + yx_i + byy_i)$,

and then dividing both new values by their greatest common divisor. As $F_1$ is a reduced form, the coefficients are bounded by $\sqrt{D} = 2\sqrt{N}$. Because the pairs are chosen from within the sieving rectangle, we know $-L^\beta < x_i < L^\beta, 0 < y_i < L^\beta$. and so the updated $x, y$ grow in absolute value by no more than $2\sqrt{N}L^\beta$. The largest possible solution to the linear system is $L^\alpha$ rows, and so the final $(x, y)$ are no larger in absolute value than $L^\beta(2\sqrt{N}L^\beta)^{L^\alpha} = (2\sqrt{N}L^\beta)^{L^\alpha}$.

The arithmetic of numbers of size $\log((2\sqrt{N}L^\beta)^{L^\alpha}) = L^\alpha \log((2\sqrt{N}L^\beta) = L^\alpha$ digits takes $L^\alpha$ time by the techniques discussed in section 4.1. As we compose at most $L^\alpha$ different pairs $(x, y)$, the composition takes time bounded by $L^{2\alpha}$.

### 4.3.6 Constructing the Reduced Inverse Square Root Form

Now that we have a square value $s^2 = F_1(x, y)$ for $F_1$, we will construct a square form, and its inverse square root, using the techniques of Gauss discussed in 2.1.1. To construct the square form, find $w$ and $z$ such that $xw - yz = 1$, via an extended euclidean algorithm in time $L^\alpha$. We then construct the form $(F_1(x, y), b(xw + zy) + 2(xz + cyw), F_1(z, w))$, and its inverse square root form $(s, -b(xw + zy) - 2(xz + cyw), sF_1(z, w))$ in time $L^\alpha$ for the computations on $L^\alpha$-digit integers.

As discussed in section 2.1.2, reducing this inverse square root form takes at most $2 + \left\lceil \frac{\log |sF_1(z,w)|}{\sqrt{\Delta}} \right\rceil$ reduction steps of the modified reduction algorithm. $sF_1(z, w)$ is $L^\alpha$ digits, and as such, operations take $L^\alpha$ time. The other coefficients are also no larger than $L^\alpha$ digits at each step, so each reduction takes at most $L^\alpha$ time, and overall computing the equivalent reduced form takes no more than $L^{2\alpha}$ time.

### 4.3.7 Returning and Checking for Success

In order to estimate the length of the return, we must estimate the infrastructure distance between $F_1$ and the square form $f = (s^2, B, C)$. The distance from the identity to the square form can be computed explicitly using corrolary 2.2.4 as

$$d(F_1, f) = \frac{1}{2} \log \left| \frac{2ax + y(b + \sqrt{\Delta})}{2ax + y(b - \sqrt{\Delta})} \right|$$

By assumption 3, we have a positive proportion of the solutions with denominator at least 1, and so rejecting solutions with smaller denominator, for the remaining solutions this quantity is bounded above by the logarithm of the numerator, $L^{\alpha}$.

This distance is twice the distance between the inverse square form $g$ and reduced ambiguous form $h$, as

$$\begin{aligned} d(g,h) + d(g,h) &= d(hg^{-1}) + d(hg^{-1}) \\ &= d(hhg^{-1}g^{-1}) \\ &= d(g \cdot g, h \cdot h) \\ &= d(f^{-1}, F_1) \\ &= d(F_1, f) \end{aligned}$$

The reduction of the inverse square root form will add at most two reduction steps to the walk to the reduced ambiguous form. The number of steps is proportional to the infrastructure distance, in this case bounded by $L^{\alpha}$.

It remains to check whether the ambiguous form yields a proper factor by taking the greatest common divisor of the middle coefficient and $N$. If this greatest common divisor is 1, then we simply try another solution to the linear system. By hypothesis 4, each solution has at least a $1/2$ chance of returning a nontrivial factor of $N$; sieving for 10 solutions gives a less than 0.1% chance of failure.

### 4.3.8 Selecting Parameters

**Theorem 4.3.1.** *Assuming Hypotheses 1, 2, 3, and 4, the expected time complexity of SQUFOF2 to factor a large square free integer $N$ using an elimination method with an exponent $r$ is $L(N)^{r/\sqrt{4r-4}+o(1)}$. The space complexity is $L(N)^{1/\sqrt{r-1}+o(1)}$.*

*Proof.* As we have just seen, the time complexity is dominated by the sieve construction, taking $L^{2\beta}$, and the solving of the linear system, taking $L^{r\alpha}$ with $r > 2$. Other portions of the algorithm take less time, such as the construction and reduction of the inverse square form from the solution which take at most $L^{2\alpha}$ operations.

The remaining question is how small we can choose $\alpha$ and $\beta$ and still have a reasonable confidence that we can get enough solutions. For a given $\alpha$, we need $L^\alpha$ smooth values in order to guarantee a solution to the linear system. By lemma 1, $\frac{6}{\pi^2}L^{2\beta} = 2\beta$ of the $(x, y)$ pairs have $\gcd(x, y) = 1$. By hypothesis 2, the probability that a given $(x, y)$ pair satisfying $\gcd(x, y) = 1$ is $L^\alpha$-smooth will be the same as for integers less than the maximum value of $|F_1(x, y)| = |x^2 + bxy + cy^2|$ on the sieved region. As $b, c < 2\sqrt{N}$, and $x, y \leq L^\beta$, this maximum value will be bounded by $4\sqrt{N}L^{2\beta} = \sqrt{N}L^{2\beta}$. The number of $B$-smooth values less than some $M$ is given by Dickman's function $\psi(M, B)$. By Dickman's theorem, given as 1.4.9 in [15], the probability of a smooth value can be computed using:

$$\psi(\sqrt{N}L^\beta, L^\alpha) \approx \sqrt{N}L^{2\beta}u^{-u}$$

where

$$u = \frac{\log(\sqrt{N}L^{2\beta})}{\log(L^\alpha)} = \frac{\log N}{2\alpha\sqrt{\log(N)\log\log(N)} + o(1)} + \frac{2\beta}{\alpha} \approx \frac{1}{2\alpha}\sqrt{\frac{\log N}{\log\log N}} = \frac{\log(L)}{2\alpha\log\log(N)}$$

And so

$$u^{-u} = \left(L^{\log(u)/\log(L)}\right)^{-u} = L^{-\log(u)/(2\alpha\log\log N)} = L^{-1/(4\alpha)+o(1)} = L^{-1/(4\alpha)}$$

This probability gives an estimated number of solutions at $L^{2\beta}L^{-1/(4\alpha)}$. Solving this for the desired number of solutions, $L^\alpha$, gives the result

$$\beta = \frac{\alpha}{2} + \frac{1}{8\alpha}$$

Giving the time complexity as

$$L^{\max\{2\beta, 2\alpha\}} = L^{\max\{\alpha+\frac{1}{4\alpha}, r\alpha\}}$$

Minimizing that exponent, we get $\alpha = \frac{1}{2\sqrt{r-1}}$, and time complexity $L^{r\alpha} = L(N)^{\frac{r}{\sqrt{4r-4}}+o(1)}$, matching the complexity of the Quadratic Sieve. Similar to the Quadratic Sieve, $r\alpha =$

$\frac{r}{2\sqrt{r-1}} = 1 + O((r-2)^2)$, and so the algorithm is not sensitive to small differences in the elimination exponent $r \in (2, 3]$.

The space requirement is $L^{2\alpha} = L(N)^{\frac{1}{r-1}+o(1)}$ for the matrix and $L^{\beta} < L^{2\alpha}$ for each row of the sieve. □

For Gaussian Elimination, the exponent $r = 3$, giving $\alpha = 1/(2\sqrt{2}) \approx 0.35$, $\beta = 3/(4\sqrt{2}) \approx 0.53$, and a time complexity approximately $L^{1.06}$. For the Coppersmith-Winograd method, we have $r \approx 2.49$, and $\alpha \approx 0.41$ and $\beta \approx 0.51$ and a time complexity $L^{1.02}$. As $r \to 2$, we have complexity approaching $L$ with parameters approaching $\alpha = 0.5$, $\beta = 0.5$.

These calculations are for very large $N$, and the convention on $L$ makes selecting practical values for factoring reasonably sized integers based on these calculations difficult. These values are not optimal for $N$ of a size likely to be factored on a modern computer, and in many cases are too small to find a solution. One should experiment with $\alpha$ and $\beta$ values which work for the size of $N$ to be factored. For factoring several example random composite $N$ between 10 and 30 decimal digits, our implementation performed best with $\alpha \approx 0.7$ and $\beta \approx 0.8$.

# 5. IMPLEMENTATION OF THE ALGORITHM

In this chapter we cover several details relevant to implementations of the algorithm, which are not relevant to theorem 4.3.1. The end performance can be better than this theorem may suggest, through some quirks of the algorithm and some optimizations that can be made.

## 5.1 General Implementation Notes

Practical implementations of SQUFOF2 can differ in many ways from the theoretical. Notably, we have found $\Delta$ need not be a fundamental discriminant; the algebraic simplifications noted in section 4.3.1 when $\Delta = 4N$ can be used even in the $N \equiv 1 \mod 4$ case, and we do not worry about whether $N$ is squarefree; on sample $N$, the algorithm performed similarly on non-squarefree $N$ as for $N$ of the same size with the same number of distinct prime factors. The algorithm does require, of course, that $N$ is odd, and is not square. However, as we are taking a square root of $N$ in the implemented algorithm, this is easy to check.

We often find ourselves using the fraction $b/2$. As is common in programs using the arithmetic of quadratic forms, we will be storing the middle coefficient as half of that described in section 2.1, and doubling where necessary instead of dividing. Discriminant calculations will be done in terms of $\Delta/4 = N = (b/2)^2 - ac$. Some literature on quadratic forms defines the forms similarly, but that notation is not as compatible with the odd discriminant cases used here.

In SQUFOF2, the sieve starts with small $x, y$, which are likely to have smaller values of the form, and thus likely to have smaller prime factors. This density means the vectors in SQUFOF2 are in practice more likely than random pairs in the range to produce a smooth value. When they are smooth, they are also more likely to provide small solutions, and so the probability of a small prime in the factor base dividing $F_1(x, y)$ to an odd power is much larger than for a large prime. This makes finding the solutions significantly more likely with fewer than $L^\alpha$ rows. Some elimination algorithms can work on a single row, or batches of rows, allowing the algorithm to sieve for smooth values only until enough linear solutions are found, well before the point at which the solutions are guaranteed.

There are faster algorithms than the ones used in the proof for some of the stages. For our implementation, when computing the quadratic residues, we use Tonelli-Shanks algorithm. For arithmetic and greatest common divisors, our implementations rely on the GMP library's internal functions, which select from the leading algorithms based on thresholds of size on the inputs.

SQUFOF2 will not work on prime numbers, and so it should be applied only to numbers which have failed a suitable pseudoprime test. In practice, it will terminate in failure after having found a chosen number $t$ square values which do not lead to square results; each failure, as estimated in chapter 4 subject to the assumptions, will have probability at most $1/2$, and so a given composite number would fail with probability $2^{-t}$; rejecting a prime input, then, would be significantly more work than factoring a composite number and would not be a proof of primality.

Alternately, SQUFOF2 will fail if it exhausts the rectangle of possible $x, y$ values. This, too, is very unlikely for effective bounds; the algorithm tends to run faster sieving at fewer $y$ values, and thus finding smaller $y$ values than $x$ values by a factor of roughly $\sqrt{N}$. This helps keep the growth smaller in the composition step, by a factor of roughly $\sqrt{N}$ at each composition, saving work there and in the subsequent reduction.

The sieve can be done in the logarithm, with addition or subtraction instead of division or multiplication, and a tolerance. In practice the algorithm is not particularly sensitive to the tolerance, with similar numbers of smooth values found in the same amount of time.

### 5.2 Large Primes Variant

The sieve tolerance can allow for one quite useful benefit. For the purposes of the square form construction, we do not require the integers to be $L^{\alpha}$-smooth, only the end product to be a square value. Similar to the Quadratic Sieve, SQUFOF2 can benefit from a *Large Primes* variant, where values which do not factor completely over the factor base and have a reasonably small remainder are stored in a separate list. When two potential rows have the same remainder, their values can be multiplied as in the algorithm, producing a pair $x, y$ with the square-free part of $F_1(x, y)$ $L^{\alpha}$-smooth, and thus able to be used in the linear algebra

and subsequent construction. This provides similar time advantages to the Quadratic Sieve, with a similarly small additional storage requirement. In addition, dividing each $x, y$ by $\gcd(x, y)$ often reduces the size of the value and pair, reducing the losses from this variant from a larger square value and thus longer reduction step.

### 5.3 Elimination over $GF(2)$

Elimination over $GF(2)$, particularly Gaussian elimination, can be done efficiently on computers using a slightly different representation of the matrix. Row addition can be done as a bitwise XOR, allowing us to run the elimination algorithm with $L^{\alpha}$ integers of bit length $L^{\alpha}$ instead of a $L^{2\alpha}$-entry matrix. In addition, using multiprecision integers, this allows for the size of the matrix to grow dynamically, requiring significantly less space for a typical solution and fewer computations than for the matrix interpretation for a typical solution. Gaussian over $GF(2)$ is also slightly faster than over a general field, as there is no need for the division step when eliminating values from the pivot column.

# 6. FUTURE WORK

## 6.1 Elimination Improvements

Adapting the algorithm to use the more specific study of elimination over $GF(2)$ could improve one of the main bottlenecks of the algorithm, and may result in a minor reduction in the exponent of $L(N)$.

## 6.2 Sparse Solutions using Compressive Sensing

A matrix over $GF(2)$ with more rows than columns permits sparser solutions as the number of independent solutions increases. At the small end, because we can linearly combine solutions over $GF(2)$, we know if we have at least 2 solutions then we have a solution of 0-norm at most 2/3rds the number of columns. One minor but potentially useful insight from this line of thought is that, in the unlikely event of finding a solution that uses a large proportion of the rows, we can save ourselves much work in the reduction by finding at least one more solution. Gains for smaller solutions are less pronounced. For even sparser solutions, we are not aware of a specific study of this over $GF(2)$, but have found some literature in the domain of Compressive Sensing, which deals with the finding of sparse solutions to linear systems where they are guaranteed to exist.

The sieve being one of the dominant portions of the algorithm, and the reductions seeming likely to be a constant factor for reasonably many rows, this will likely not reduce the theoretical complexity. However, it may result in some performance gains in practice.

## 6.3 Multipliers

One useful consequence of Hypothesis 4 is that SQUFOF2, similar to SQUFOF, is equally likely to find any factor of a given squarefree $N$. For $N$ at which the algorithm would be relatively slow, one could factor a multiple of $N$. This modification is analyzed by Gower for the original SQUFOF in section 5 of [3]. A similar analysis may produce similar results for SQUFOF2.

### 6.4 Parallelization of the Algorithm

One thing to note about SQUFOF2 is that much of it parallelizes particularly well. Some of the most expensive portions of the algorithm - the sieve and the composition - are easily partitioned into tasks and distributed. Various parallel elimination algorithms for general matrices exist, including parallelized Gaussian elimination. As is common with parallel algorithms, there are some drawbacks in terms of order of operations, but that is not much of a loss compared to the gains from paralellization.

There are expensive steps in the algorithm which do not have obvious parallel implementations. Most notable is the reduction of the composed form. At time of writing, we are not aware of a published parallel algorithm for this, but the algorithm used here bears a strong similarity euclidean algorithm for the computing of greatest common divisors, and there is a reasonable hope such a parallel algorithm could be adapted from that well-studied problem. As one small benefit, the reduction is not expected to be done more than a small number of times, and likely only once or twice, and the reduction and walk can be done in a separate process to the rest of the program allowing the continued search for other candidates even while solutions are being analyzed.

A full description and analysis of a parallel version of the algorithm may be of interest, especially with a parallel algorithm for form reduction.

# 7. CONCLUSION

SQUFOF2 represents a lineage of factoring algorithms that has been largely forgotten. However, it is no faster than the Quadratic Sieve and, for large integers, slower than the Number Field Sieve. In the modern world of subexponential factoring algorithms, it is more of a curiosity than a useful tool for factoring particularly large numbers.

# REFERENCES


[1] C. Pomerance, "Analysis and comparison of some integer factoring algorithms," in *Computational Methods in Number Theory, Part 1*, H. W. Lenstra Jr. and R. Tijdeman, Eds., ser. Math. Centrum Tract, vol. 154, CWI, Amsterdam, 1982, pp. 89–139.

[2] D. Shanks, "SQUFOF Notes," Manuscript, 30 pages, available at `http://homes.cerias.purdue.edu/~ssw/shanks.pdf`.

[3] J. Gower and S. S. Wagstaff Jr., "Square form factorization," *Math. Comp.*, vol. 77, pp. 551–588, 2008.

[4] H. W. Lenstra and C. Pomerance, "A rigorous time bound for factoring integers," *Journal of the American Mathematical Society*, vol. 5, no. 3, pp. 483–516, 1992.

[5] H. Cohen, *A Course in Computational Algebraic Number Theory*. New York: Springer-Verlag, 1996.

[6] D. A. Buell, *Binary Quadratic Forms, Classical Theory and Modern Computations*. Berlin, New York: Springer-Verlag, 1989.

[7] H. W. Lenstra Jr., "On the calculation of regulators and class numbers of quadratic fields," in *Journées Arithmétiques, 1980*, J. V. Armitage, Ed., ser. Lecture Notes Series, vol. 56, London Math. Soc., 1982, pp. 123–150.

[8] D. Shanks, "The infrastructure of a real quadratic field and its applictions," in *Proceedings of the 1972 Number Theory Conference, Boulder*, 1972, pp. 217–224.

[9] C. F. Gauss, *Disquisitiones Arithmeticae*, English. New Haven: Yale University Press, 1966.

[10] D. Shanks, *Square forms factorization*, Lecture, before 1975.

[11] R. Crandall and C. B. Pomerance, *Prime numbers: a computational perspective*. Springer Science & Business Media, 2006, vol. 182.

[12] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, Fifth. Oxford, England: Clarendon Press, 1979.

[13] T. M. Apostol, *Intoduction to Analytic Number Theory*. New York: Springer-Verlag, 1976.

[14] D. Coppersmith and S. Winograd, "On the asymptotic complexity of matrix multiplication," *SIAM J. Comput.*, vol. 11, pp. 472–492, 1982.

[15] R. Crandall and C. Pomerance, *Prime Numbers: A Computational Perspective*. New York: Springer-Verlag, 2001.