Shorter Compact Representations in Real Quadratic Fields

Alan K. Silvester[1], Michael J. Jacobson, Jr.[2,⋆], and Hugh C. Williams[1]

[1] Department of Mathematics and Statistics, University of Calgary
2500 University Drive NW, Calgary, Alberta, Canada T2N 1N4
aksilves@ucalgary.ca, williams@math.ucalgary.ca
[2] Department of Computer Science, University of Calgary
2500 University Drive NW, Calgary, Alberta, Canada T2N 1N4
jacobs@cpsc.ucalgary.ca

Abstract. Compact representations are explicit representations of algebraic numbers with size polynomial in the logarithm of their height. These representations enable much easier manipulations with larger algebraic numbers than would be possible using a standard representation and are necessary, for example, in short certificates for the unit group and ideal class group. In this paper, we present two improvements that can be used together to reduce significantly the sizes of compact representations in real quadratic fields. We provide analytic and numerical evidence demonstrating the performance of our methods, and suggesting that further improvements using obvious extensions are likely not possible.

Keywords: compact representation, real quadratic field, fundamental unit, infrastructure.

1 Introduction

Let α (> 1) be an algebraic integer in the quadratic order $\mathcal{O}$ of discriminant Δ (> 0). If we put $\alpha = (x + y\sqrt{\Delta})/2$, where the coefficients x and y are rational integers, it is often the case that even when the absolute norm of α, $|N(\alpha)|$, is small, the values of x and y can be very large. Consider the case where $|N(\alpha)| = 1$. In this case α is a unit of $\mathcal{O}$ and therefore a power of the fundamental unit η_Δ of $\mathcal{O}$, but we know (see, for example, Chapter 9 of [16]) that the coefficients in η_Δ can be very large, so much so, that even if Δ is only moderately large it is difficult to impossible to write them down, using conventional decimal representation. For example, when we attempt to solve the famous Cattle Problem of Archimedes we encounter an order of discriminant $\Delta = 410286423278424$ and the coefficients in η_Δ contain about 103,200 decimal digits each. Furthermore, if $\Delta = 99067609099585387015627160788 6$, a number of 30 decimal digits, then each of the coefficients in η_Δ contains more than 2×10^{15} decimal digits,

⋆ The second and third authors are supported in part by NSERC of Canada.

M. Fischlin and S. Katzenbeisser (Eds.): Buchmann Festschrift, LNCS 8260, pp. 50–72, 2013.

see [16, pp. 62,285]. As the average paperback novel contains about one million symbols, this means that it would take over two billion such volumes to record only one of the coefficients. Thus, it is necessary to find a much more compact representation for α other than simply recording the decimal representation of each of the coefficients.

Although a technique for doing this was anticipated in work of Lagarias [17,18], it was Cohen in [4, pp 274, 280-282], who first described in print a method that could be applied to this problem. Somewhat before Cohen's idea had appeared, I (Hugh Williams) had been approached by my graduate student, Gilbert Fung, with the question of how the units in a cubic field with negative discriminant could be represented without recourse to the voluminous decimal representation. This discussion led, in 1991, to our writing a paper [9] on this topic, which we submitted to Mathematics of Computation. Unfortunately, the editor, Dan Shanks, did not seem to know what to do with the paper, and it languished in his care for many months, without being accepted or rejected. At this point, I must confess that I had rather lost interest in this idea, but fortunately I was invited to present a paper at a meeting on Computational Algebra and Number Theory to be held at the University of Sydney in November of 1992. During a previous visit to Saarbrücken, I had explained my ideas to Johannes Buchmann, and he suggested that we work jointly on an account of the technique for application to real quadratic fields. I wrote up a preliminary version of the paper and sent it to Johannes for comments and revisions. This was the paper that I presented at the Sydney meeting. Johannes enlisted the aid of one of his graduate students, Christoph Thiel, and they produced a completely revised version of the paper, with Christoph being added as a third author. Ultimately, this revised paper [2] appeared in the proceedings of the Sydney meeting.

This paper essentially elaborated upon the idea of Cohen, but both extended and formalized it. An improved, but much briefer version appears in Buchmann and Vollmer [3, pp.251-256]. In [16, Chapter 12], we presented another variant which allows us to avoid trying to approximate logarithms and produces somewhat better results than those in [3]. The basic idea of all these techniques is to represent the algebraic number α in terms of a power product which satisfies a number of conditions. In doing this one can drastically reduce the number of digits needed to record the coefficients in α. Furthermore, it can be shown how arithmetic operations can be performed on such representations, leading to more efficient calculations than those required for the standard decimal representation. It must be emphasized that in order to produce a compact representation of α, we usually need an approximate value, within about 1, say, of $\log \alpha$.

The purpose of this paper is to provide an adjustment to our previous definition of a compact representation, which allows us to compute it in fewer iterations than those required for the earlier definition. We also provide an analysis which suggests that these new compact representations are quite likely as small as we can expect to achieve with these methods. These analytical results are backed up by various numerical computations.

2 Background on Quadratic Fields

For more details about quadratic fields, the reader is referred to [16], upon which the following material is based.

Let $D \in \mathbb{Z}$ be an integer, not a perfect square, and greater than 1. The elements of the real quadratic field $\mathbb{K} = \mathbb{Q}(\sqrt{D})$ have the form $\alpha = (a + b\sqrt{D})/c$ for integers a, b, and c. The *conjugate* $\overline{\alpha}$ of a $\alpha \in \mathbb{K}$ is given by $\overline{\alpha} = (a - b\sqrt{D})/c$. The quadratic integers of $\mathbb{K}$ have the form $\alpha = a + b\omega$ where

$$r = \begin{cases} 1 \text{ if } D \not\equiv 1 \pmod 4 \,, \\ 2 \text{ otherwise} \end{cases} \quad \text{and} \quad \omega = \frac{r - 1 + \sqrt{D}}{r} \,, \tag{1}$$

The *height* of a quadratic integer measures its size.

Definition 1. *The* height *of a quadratic integer* α *is* $H(\alpha) = \max\{|\alpha|, |\overline{\alpha}|\}$.

Recalling that $|N(\alpha)| = |\alpha\overline{\alpha}| \geq 1$, we see that $H(\alpha) \geq 1$ and so an element's height cannot be arbitrarily small.

The set of all quadratic integers of $\mathbb{K}$ is called the *maximal order*, denoted $\mathcal{O}_{\mathbb{K}}$. The *field discriminant* $\Delta_{\mathcal{O}_{\mathbb{K}}}$ is the discriminant of the order $\mathcal{O}_{\mathbb{K}}$, and can be explicitly determined as $\Delta_{\mathcal{O}_{\mathbb{K}}} = 4D/r^2$. Suborders $\mathcal{O}_\Delta$ of $\mathcal{O}_{\mathbb{K}}$ have discriminant $\Delta = f^2\Delta_{\mathcal{O}_{\mathbb{K}}}$, $f > 1$.

The smallest unit of $\mathcal{O}_\Delta$ greater than 1 is called the *fundamental unit* and is denoted η_Δ. If $\eta \in \mathcal{O}_\Delta^*$ is a unit then $\eta = \pm\eta_\Delta^n$ for some integer n, i.e., the unit group of a quadratic order $\mathcal{O}_\Delta$ is given by $\mathcal{O}_\Delta^* = \langle -1, \eta_\Delta \rangle$. As the size of η_Δ grows exponentially as Δ increases, we often work with a more manageable quantity called the *regulator*, denoted $\mathcal{R} = \log \eta_\Delta$.

2.1 Ideals

The non-zero ideals of $\mathcal{O}_\Delta$ can be represented as

$$\mathfrak{a} = S\left[\frac{Q}{r}, \frac{P + \sqrt{D}}{r}\right] \,, \tag{2}$$

where $S, Q, P \in \mathbb{Z}$, $r \in \{1, 2\}$, $r \mid Q$, and $rQ \mid D - P^2$. We will refer to an ideal as "$S[Q, P]$" where it is understood that S, Q, and P satisfy the conditions listed here. Given two ideals in $S[Q, P]$ representation, well-known formulas originally due to Gauss can compute their product in $S[Q, P]$ representation [16, Ch.5].

A *principal* ideal $\mathfrak{a}$ is an $\mathcal{O}_\Delta$-ideal which can be written as $\mathfrak{a} = (\theta)$ for some $\theta \in \mathcal{O}_\Delta$, in other words it has only a single generator. Two ideals are said to be *equivalent* if there exist non-zero $\alpha, \beta \in \mathcal{O}_\Delta$ such that $(\alpha)\mathfrak{a} = (\beta)\mathfrak{b}$ and we denote this by $\mathfrak{a} \sim \mathfrak{b}$. We remark that we will frequently abuse this notation by writing $\mathfrak{a} = (\gamma)\mathfrak{b}$, where it is understood that $(\gamma) = (\beta/\alpha)$ is a fractional $\mathcal{O}_\Delta$-ideal, i.e., there exists a non-zero $\alpha \in \mathcal{O}_\Delta$ such that $\alpha(\gamma) \subseteq \mathcal{O}_\Delta$.

Our algorithms for compact representations rely on arithmetic with principal ideals that are reduced. An $\mathcal{O}_\Delta$-ideal $\mathfrak{a}$ is *primitive* if it cannot be written as

an integer multiple of another ideal $\mathfrak{b}$, i.e., if $\mathfrak{a} \neq (m)\mathfrak{b}$ for any $m \in \mathbb{Z}$, where $|m| > 1$. Using the notation of (2), we say an ideal $\mathfrak{a}$ is primitive if $S = 1$, denoted as $\mathfrak{a} = [Q, P]$. The *norm* $N(\mathfrak{a})$ of an $\mathcal{O}_\Delta$-ideal $\mathfrak{a}$ is the index $|\mathcal{O}_\Delta/\mathfrak{a}|$ and when the ideal $\mathfrak{a}$ is written in the form of (2), we have

$$N(\mathfrak{a}) = S^2 Q/r \ . \tag{3}$$

Finally, an $\mathcal{O}_\Delta$-ideal $\mathfrak{a}$ is *reduced* if it is primitive and there does not exist $\alpha \in \mathfrak{a}$, $\alpha \neq 0$, such that both $|\alpha| < N(\mathfrak{a})$ and $|\overline{\alpha}| < N(\mathfrak{a})$. A useful property of reduced $\mathcal{O}_\Delta$-ideals, when written in the form of (2), is that $0 < P < \sqrt{D}$ and $0 < Q < 2\sqrt{D}$ [16, Cor. 5.8.1, p. 101]. That is, if $\mathfrak{a}$ is a reduced ideal, then $N(\mathfrak{a}) < \sqrt{\Delta}$.

A primitive ideal $\mathfrak{a}$ given by $[Q, P]$ can be reduced by expanding the continued fraction of $\alpha = (P + \sqrt{D})/Q$ as described in [16, Ch.5]. If we start the reduction procedure with the reduced ideal $\mathfrak{a} = \mathfrak{a}_1 = \mathcal{O}_\Delta$, we obtain a sequence of reduced principal ideals $\mathfrak{a}_{i+1} = (\theta_i)\mathfrak{a}_1$. Since the coefficients of a reduced ideal are bounded, there are only finitely many, and consequently this sequence must be periodic. Hence, we can find some minimal $p > 0$ such that $\mathfrak{a}_{p+1} = \mathfrak{a}_1$. These ideals can be arranged into a cycle $\mathcal{C} = \{\mathfrak{a}_1, \mathfrak{a}_2, \ldots, \mathfrak{a}_p\}$ with

$$1 = \theta_1 < \theta_2 < \theta_3 < \cdots < \theta_p < \cdots \ .$$

called the *cycle of reduced principal ideals*. A well-known fact—derived, for instance, from [16, (5.33), p. 113]—is that if the fundamental unit $\eta_\Delta = \theta_{p+1}$. It can be shown that $p \approx \mathrm{O}(\mathcal{R}) = \mathrm{O}(\Delta^{1/2+\epsilon})$.

2.2 Infrastructure

The *infrastructure*, discovered by Daniel Shanks [20], refers to the group-like structure existing within each equivalence class of ideals in $\mathcal{O}_\Delta$. For our purposes, we focus on the principal class, in particular the set of reduced principal ideals $\mathcal{C}$. The arithmetic properties of this set are key to algorithms for computing compact representations.

Let $\mathfrak{a}_1 = [1, \omega]$ be the first ideal in $\mathcal{C}$. The *distance* of $\mathfrak{a}_i = (\theta_i)$ is defined as $\delta_i = \delta(\mathfrak{a}_i) = \log_2 \theta_i \pmod{\mathcal{R}}$. Let $\mathfrak{a}_i$ and $\mathfrak{a}_j$ be two reduced principal ideals in $\mathcal{C}$. Since they are principal, their product $\mathfrak{a} = \mathfrak{a}_i\mathfrak{a}_j = (\theta_i\theta_j)$ will also be principal. However, $\mathfrak{a}$ may no longer be reduced, but is equivalent to some reduced principal ideal $\mathfrak{a}_l \in \mathcal{C}$. Thus,

$$\mathfrak{a}_l = \left(\frac{\theta'_k \theta_i \theta_j}{m}\right)$$

$$\delta_l = \delta(\mathfrak{a}_l) = \log_2\left(\frac{\theta'_k \theta_i \theta_j}{m}\right) \equiv \delta_i + \delta_j + \log_2 \frac{\theta'_k}{m} \pmod{\mathcal{R}} \ . \tag{4}$$

We denote by $\mathfrak{a}_i \star \mathfrak{a}_j$ the computation of the reduced ideal equivalent to the product ideal $\mathfrak{a}_i\mathfrak{a}_j$ and refer to this process as a *giant step*. The key observation is that $\mathcal{C}$ is almost a group under this operation — only associativity fails,

because instead of having $\delta_l = \delta_i + \delta_j$ in (4), we are stuck with the additional error term $\log_2(\theta'_k/m)$ and so δ_l is only close to $\delta_i + \delta_j$. However, this error term can be bounded in absolute value, say by μ. This bound depends on the particular reduction algorithm selected, but for the method described above, it can be shown [16, p. 175] that $\mu < \mathrm{O}(\log \Delta)$ which is quite small compared to $\delta_i, \delta_j \approx \mathrm{O}(\mathcal{R})$.

In the rest of this paper, we will refer to one application of the continued fraction algorithm to the ideal $\mathfrak{a}_i$ as a *(forward) baby step*, denoted $\mathfrak{a}_{i+1} = \rho(\mathfrak{a}_i)$. Although we will not derive formulas here, given a reduced principal ideal $\mathfrak{a}_i \in \mathcal{C}$, we can also compute the *backward baby step* $\mathfrak{a}_{i-1} = \rho^{-1}(\mathfrak{a}_i)$ [16, §3.4, p. 64].

2.3 Approximating Distances

While performing computations in the infrastructure, we need to keep track of distances while maintaining accurate approximations in the face of round-off and truncation errors. The method of (f, p) representations [14], adapted from ideas of Hühnlein and Paulus [12], was devised to provide provable bounds on the round-off and truncation errors accumulated during computations. This idea was later refined by the authors of [15] and we will use the method of w-near (f, p) representations, as described in [16, Ch. 11, p. 265].

Let $p \in \mathbb{N}$ and $f \in \mathbb{R}$ be such that $1 \leq f < 2^p$. If $\mathfrak{a}$ is a primitive $\mathcal{O}_\Delta$-ideal, then an (f, p) *representation* of $\mathfrak{a}$ is a triple $(\mathfrak{b}, d, k)$ where $\mathfrak{b} \sim \mathfrak{a}$, $d \in \mathbb{N}$ with $2^p < d \leq 2^{p+1}$, and $k \in \mathbb{Z}$. In addition, there exists $\theta \in \mathbb{K}$ such that $\mathfrak{b} = (\theta)\mathfrak{a}$ with

$$\left| \frac{\theta}{2^{k-p} d} - 1 \right| < \frac{f}{2^p} .$$

In essence, an (f, p) representation stores both an approximation to the *relative generator* θ and an approximation of its distance, both with precision p. The parameter f is a measure of the approximation error, though it is rarely if ever explicitly computed. If $\mathfrak{b}$ is a reduced $\mathcal{O}_\Delta$-ideal, then $(\mathfrak{b}, d, k)$ is a *reduced* (f, p) *representation* of $\mathfrak{a}$.

A *w-near* (f, p) representation is a a reduced (f, p) representation $(\mathfrak{b}, d, k)$ of an $\mathcal{O}_\Delta$-ideal $\mathfrak{a}$ with the following two additional conditions:

1. $k < w$ for some $w \in \mathbb{Z}^+$ and
2. if $\rho(\mathfrak{b}) = (\psi)\mathfrak{b}$ then there exist integers d' and k' such that $k' \geq w$, $2^p < d' \leq 2^{p+1}$ and
$$\left| \frac{\psi\theta}{2^{k'-p} d'} - 1 \right| < \frac{f}{2^p} .$$

Such representations have the useful property that $\theta \approx 2^w$ and $k \approx w$. Since this property will be used repeatedly in later material, particularly with respect to compact representations, we will state it more formally.

Lemma 1 ([16, Lem. 11.3, p. 270]). *Let $(\mathfrak{b}, d, k)$ be a w-near (f, p) representation of some $\mathcal{O}_\Delta$-ideal $\mathfrak{a}$ with $p > 4$ and $f < 2^{p-4}$. If θ and ψ are defined as above, then*

$$\frac{15N(\mathfrak{b})}{16\sqrt{\Delta}} < \frac{15}{16\psi} < \frac{\theta}{2^w} < \frac{17}{16} \quad \textit{and} \quad -\log_2 \frac{34\psi}{15} < k - w < 0\,.$$

2.4 Algorithms

We define here the basic algorithms we require for performing various computations with (f, p) representations. The majority of these algorithms will not be explicitly presented, rather references to the appropriate sections of [16] will be given.

1. Given $\mu, \nu \in \mathcal{O}_\Delta$, Algorithm IMULT [16, Alg. 12.2, p. 286] computes $\lambda = \mu\nu$.
2. Given an (f, p) representation $(\mathfrak{b}, d, k)$ of $\mathfrak{a}$, we can determine a w-near representation $(\mathfrak{c}, g, h)$ of $\mathfrak{a}$ along with the corresponding relative generator using EWNEAR [16, Alg. 12.1, pp. 286 and 457] provided that $k < w$. In order to implement our improved algorithms, we require a version that also works in the case $k > w$. A modified version of EWNEAR that takes care of this is presented in Appendix A.
3. If $(\mathfrak{a}[x], d_x, k_x)$ and $(\mathfrak{a}[y], d_y, k_y)$ are respectively, x- and y-near (f', p) and (f'', p) representations of $\mathfrak{a} = (1)$, we can employ EADDXY [16, Alg. 12.3, p. 286] to produce an $x + y$-near (f, p) representation $(\mathfrak{a}[x + y], d, k)$ of $\mathfrak{a}$ where $f = 13/4 + f' + f'' + f'f''/2^p$, as well as a relative generator $\lambda \in \mathcal{O}$ such that
$$\mathfrak{a}[x + y] = \left(\frac{\lambda}{N(\mathfrak{a}[x])N(\mathfrak{a}[y])}\right)\mathfrak{a}[x]\mathfrak{a}[y]\,.$$
4. If $(\mathfrak{a}[x], d', k')$ is an x-near (f', p) representation of the $\mathcal{O}_\Delta$-ideal $\mathfrak{a} = (1)$, algorithm ETRIPLEX (described in Appendix B) computes a $3x$-near (f, p) representation $(\mathfrak{a}[3x], d, k)$ of $\mathfrak{a}$ with $f = 13/4 + 3f' + 3f'^2/2^p + f'^3/2^{2p}$ and
$$\lambda = \frac{a + b\sqrt{D}}{r} \text{ such that } \mathfrak{a}[3x] = \left(\frac{\lambda\theta^3}{N(\mathfrak{a}[x])^3}\right)\mathfrak{a},$$
where $\mathfrak{a}[x] = (\theta)\mathfrak{a}$.

3 Compact Representations

In this section, we describe how to compute a compact representation. Our presentation follows that of [16]; for a more detailed description, see [16, §§12.2–3, pp. 290–304].

The algorithm AX [16, Alg. 11.6, pp. 279–80] computes a reduced principal ideal $\mathfrak{a}$ at distance approximately x from $\mathfrak{a}_1 = (1)$. At the heart of AX is a square-and-multiply routine that uses the binary expansion of x to make a series of giant steps in the infrastructure. For each bit in the binary expansion, we compute the giant step $\mathfrak{a}_j \star \mathfrak{a}_j$—the squaring step—which results in an ideal with roughly

double the distance from where we started. If the current bit is 1, then we also adjust the resulting ideal via ρ to correct the distance—the multiplying step.

At each stage of AX, suppose we were to keep track of the relative generator that appears. For the giant steps we would have μ_j such that $\mathfrak{a}'_{j+1} = (\mu_j/N(\mathfrak{a}_j)^2)\mathfrak{a}_j^2$ from EADDXY, and for the adjustment steps we would have ν_j such that $\mathfrak{a}_{j+1} = (\nu_j)\mathfrak{a}'_{j+1}$ from EWNEAR. Then $\mathfrak{a}_{j+1} = (\lambda_j/L_{j+1}^2)\mathfrak{a}_j^2$ with $\lambda_j = \mu_j\nu_j$ (computed with IMULT) and $L_{j+1} = N(\mathfrak{a}_j) < \sqrt{\Delta}$ Note that the ν_j values will be small compared to the μ_j.

At the end of AX, we will not only have an ideal $\mathfrak{a}_n = \mathfrak{a}[x] = (\theta)$ at distance approximately x, but also a list of quadratic integers $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, and a list of ideal norms $\{L_1, L_2, \ldots, L_n\}$. At this point it should be clear that if we combine the relative generators λ_j and ideal norms L_j by an appropriate combination of multiplications, divisions, and exponentiations, we will get the generator θ. This leads to the definition of a compact representation.

Definition 2. *For any θ such that $(\theta) = \mathfrak{a}[x] \in \mathcal{O}_\Delta$, a* compact representation *of θ is*

$$\theta = \prod_{i=0}^{l} \left(\frac{\lambda_i}{L_i^2}\right)^{2^{l-i}}$$

where the following properties are satisfied:

1. *$l = \mathrm{O}(\log\log\theta)$ for large θ.*
2. *$\lambda_i \in \mathcal{O}_\Delta$ and L_i is an integer $(0 \le i \le l)$.*
3. *$0 < L_i \le \Delta^{1/2}$ and $H(\lambda_i) = \mathrm{O}(\Delta)$ $(0 \le i \le l)$.*
4. *$\pi_j \in \mathcal{O}_\Delta$, $L_{j+1} = |N(\pi_j)|$,*

$$\pi_j = \prod_{i=0}^{j} \left(\frac{\lambda_i}{L_i^2}\right)^{2^{j-i}} ,$$

π_j generates a reduced ideal $\mathfrak{b}_j$, where $\mathfrak{b}_0 = \mathfrak{a}[1]$, and

$$L_{i+1}^2\mathfrak{b}_{i+1} = \lambda_{i+1}\mathfrak{b}_i^2 \quad (0 \le i \le l-1) .$$

We remark that this definition is slightly different from that given in [16]. Notice that upon substituting $d_i := L_{i+1}$, $\lambda := d_l$, $L_0 = N\big((1)\big) = 1$ and shifting the denominators, we get the same presentation as in [16, (12.8), p. 290].

Returning to the Cattle Problem, a compact representation of $\eta_{4\,10286\,42332\,78424}$ requires only 1,212 bits, whereas writing out its coefficients explicitly would require 206,400 bits. In general, in order to write down θ using standard decimal representation, we require $O(\log_2\theta)$ bits. However, using a compact representation, we require only $\mathrm{O}((\log_2\log_2\theta)\log_2\Delta)$ bits to express θ.

4 Reducing the Size of the Terms

The overall size of a compact representation is determined by two factors: the size of the individual terms and the total number of terms. In this section, we describe a method to reduce the size of the terms.

Consider the sequence of s_i values computed as AX executes, corresponding to the intermediate results produced by applying a square-and-multiply process according to the binary representation of x. Let $x = \sum_{i=0}^{l} 2^{l-i} b_i$ be such a representation and set $s_0 = b_0$ $(= 1)$. As we progress through AX computing giant steps, ideally we wish to compute

$$\mathfrak{a}[s_{i+1}]' = \mathfrak{a}[s_i]^2 \ .$$

However because of the way giant steps in the infrastructure work, when we compute $\mathfrak{a}[s_i]^2$ we actually "fall short" of this ideal, computing instead

$$\mathfrak{a}[s_{i+1}]' = (\mu_i)\mathfrak{a}[s_i]^2$$

for a correction factor μ_i corresponding to the error term in (4). We then take the ideal $\mathfrak{a}[s_{i+1}]'$ and, depending on the value of b_i, either set $\mathfrak{a}[s_{i+1}] = \mathfrak{a}[s_{i+1}]'$ or compute a baby-step $\mathfrak{a}[s_{i+1}] = \rho(\mathfrak{a}[s_{i+1}]') = (\nu_i)\mathfrak{a}[s_{i+1}]'$ so that

$$\mathfrak{a}[s_{i+1}] = (\lambda_i)\mathfrak{a}[s_i]^2 \ .$$

As we also mentioned in Section 3, the μ_i values constitute the bulk of the λ_i terms that we wish to store as a compact representation. With some careful reasoning [16, pp. 445–6], one can show that when using EADDXY as in [16, Alg. 12.3, p. 286] we have

$$\mathrm{O}(\Delta^{1/4}) < \mu_i < \mathrm{O}(\Delta^{3/4}) \ . \tag{5}$$

In other words, while the relative generator μ_i is bounded and cannot become too large, it also cannot become very small.

In the following, we will describe a method to adjust the s_i values to exploit the short-fall we experience and so reduce the upper bound in (5). If we increase the s_i values at each step, we will compute ideals $\mathfrak{a}[s_{i+1}]'$ further along the infrastructure than we want. As before, we still experience a short-fall, but will be closer to our goal of $\mathfrak{a}[s_{i+1}]$ than before. By using a larger and backwards EWNEAR step, we use the relative generator ν_i to cancel out, in a sense, a substantial portion of μ_i.

Let $h \in \mathbb{Z}^+$ and let n be the largest integer such that $x \geq (2^n - 1)h$. Set $y = x + (2^n - 1)h$ and compute the binary representation of $y = \sum_{i=0}^{l} 2^{l-i} b_i$. We iterate the while-loop over $0 \leq i < l - n$ as usual $(s_{i+1} = 2s_i + b_i)$, and use $s_{i+1} = 2s_i + b_i - h$ for $l - n \leq i < l$. This yields

$$s_i = \begin{cases} \sum_{j=0}^{i} 2^{i-j} b_j & \text{for } 0 \leq i \leq l - n \\ \sum_{j=0}^{i} 2^{i-j} b_j - (2^{n-l+i} - 1)h & \text{for } l - n < i \leq l. \end{cases}$$

Note that $s_l = y - (2^n - 1)h = x$; thus at the end of the algorithm, we have $\mathfrak{a}[s_l] = \mathfrak{a}[x]$ as desired. Furthermore, we clearly have $s_i > 0$ for $0 \leq i \leq l - n$. now, if $s_i \leq 0$ for some i such that $l - n < i < l$, then

$$s_{i+1} = 2s_i + b_i - h \leq 0$$

because $h \geq 1$. By induction, we get $x = s_l \leq 0$, a contradiction. Thus, $s_i > 0$ for all i such that $0 \leq i \leq l$.

All that remains is to determine an appropriate value for h and from that, determine how much the height of λ_i can be reduced.

Recalling (5), we see that $h = \lceil (1/4)\log_2 \Delta \rceil$ is a good choice. In order to determine how much λ_i is reduced, we must compute a revised bound for $H(\lambda_i)$. As the algorithm executes, it finds a series of reduced principal $\mathcal{O}_\Delta$-ideals $\mathfrak{a}[s_i] = \mathfrak{a}_i = (\pi_i)\mathfrak{a}_1$. By Lemma 1, we can conclude that for an s_{i-1}-near (f,p)-representation of $\mathfrak{a}[s_{i-1}]$ and an s_i-near (f,p)-representation of $\mathfrak{a}[s_i]$ where $l - n < i \leq l$,

$$\frac{15L_i}{16\sqrt{\Delta}} 2^{s_{i-1}} < \pi_{i-1} < \frac{17}{16} 2^{s_{i-1}} \quad \text{and} \quad \frac{15L_{i+1}}{16\sqrt{\Delta}} 2^{s_i} < \pi_i < \frac{17}{16} 2^{s_i} . \tag{6}$$

From the definition of a compact representation, we also know that

$$\pi_i = \left(\frac{\lambda_i}{L_i^2}\right) \pi_{i-1}^2 \quad \Longrightarrow \quad \lambda_i = \frac{L_i^2 \pi_i}{\pi_{i-1}^2}, \tag{7}$$

and so combining (6) and (7), we get

$$\lambda_i < L_i^2 \left(\frac{17}{16} 2^{s_i}\right) \left(\frac{16\sqrt{\Delta}}{15L_i} 2^{-s_{i-1}}\right)^2 = \frac{16 \cdot 17}{15^2} 2^{s_i - 2s_{i-1}} \Delta .$$

Since $s_i - 2s_{i-1} = b_i - h$ and $b_i \in \{0, 1\}$ we have

$$0 < \lambda_i < \frac{5}{2} 2^{-\lceil (1/4)\log_2 \Delta \rceil} \Delta \leq \frac{5}{2} \Delta^{-1/4} \Delta = \frac{5}{2} \Delta^{3/4}.$$

We can also show that $|\overline{\lambda_i}| < 5/2\Delta^{3/4}$ by using the reasoning of [16, p.289]. Hence, our modified algorithm reduces the height of λ_i for $l - n < i \leq l$ from $\mathrm{O}(\Delta)$ to $\mathrm{O}(\Delta^{3/4})$. The λ_i values for $0 \leq i \leq l - n$ will have height $\mathrm{O}(\Delta)$, but there are only a small number of these as $l - n < 2 + \log_2 h$. Thus, they will have little impact on the amount of space needed to record θ.

We refer to a compact representation computed using the ideas above as an h-compact representation.

Theorem 1. *Let $\theta \in \mathcal{O}_\Delta$ such that $\mathfrak{a}[x] = (\theta)$ for some $x \in \mathbb{Z}^+$. The number of bits in an h-compact representation of θ is $\mathrm{O}((\log_2 \log_2 \theta) \log_2 \Delta^{3/4})$.*

Proof. From the preceding discussion, we know $H(\lambda_i) < (5/2)\Delta^{3/4}$. As $l = \lceil \log_2 x \rceil$ and $2^x < (16\sqrt{\Delta}/15)\theta$, we also have $l = \mathrm{O}(\log_2 \log_2 \theta)$. Thus, we require

$$\mathrm{O}(l \log_2 \Delta^{3/4}) = \mathrm{O}((\log_2 \log_2 \theta) \log_2 \Delta^{3/4})$$

bits to express θ as an h-compact representation. □

Although our improvement does not change the asymptotic running time, the improvement to the O-constant does yield a significant improvement in practice. Returning to our running example, an h-compact representation of the fundamental unit $\eta_{410286423278424}$ uses only 974 bits, a substantial size reduction of 19.6% as compared to the standard compact representation.

5 Reducing the Number of Terms

In the following, we describe a method to reduce the number of terms in a compact representation, in an effort to further reduce the overall size. Recall that for each step of the algorithm we compute an ideal $(\mathfrak{a}[s_{i+1}])$ at double the distance of the ideal we are currently at $(\mathfrak{a}[s_i])$. In order to store fewer terms, we have to progress further from ideal to ideal, for example, by computing an ideal at triple the distance we are at currently. In other words, instead of computing the binary expansion of x and applying a square-and-multiply routine, we could compute a ternary expansion and use a cube-and-multiply routine, using ETRIPLEX in place of EADDXY. We refer to a compact representation produced in this manner as a 3-compact representation.

To see that this method computes a correct compact representation, note that it produces a series of reduced principal $\mathcal{O}_\Delta$-ideals $\mathfrak{a}[s_i] = \mathfrak{b}_i = (\pi_i)\mathfrak{a}_1$ $(\mathfrak{a}_1 = (1))$ where

$$\left| \frac{2^p \pi_i}{2^{k_i} d_i} - 1 \right| < \frac{f}{2^p} .$$

Moreover, $\pi_i \in \mathcal{O}_\Delta$, $|N(\pi_i)| = N(\mathfrak{a}[s_i]) = N(\mathfrak{b}_i) = L_{i+1}$ and if p is sufficiently large, we can appeal to a result analogous to Theorem 11.9 of [16, p. 280] to ensure that $f < 2^{p-4}$. If we set $\lambda_i = (m_i + n_i\sqrt{D})/r$, then

$$\pi_{i+1} = \left(\frac{\lambda_{i+1}}{L_{i+1}^3} \right) \pi_i^3 \tag{8}$$

where $\pi_0 = \lambda_0$. If we define $L_0 = 1$, then we get

$$\pi_j = \prod_{i=0}^{j} \left(\frac{\lambda_i}{L_i^3} \right)^{3^{j-i}}$$

for $j = 0, 1, \ldots, l$. When $j = l$, we have $s_l = x$, $\mathfrak{a}[x] = \mathfrak{b}_l = (\pi_l)$, and hence $\mathfrak{a}[x] = (\theta)$ where

$$\theta = \prod_{i=0}^{l} \left(\frac{\lambda_i}{L_i^3} \right)^{3^{j-i}} .$$

One simple improvement on this idea that will slightly reduce the sizes of the terms is to use a signed ternary representation of x. Instead of digits $0, 1, 2$, in the ternary representation of x, we use digits $-1, 0, 1$, thereby reducing the average size of terms obtained when $b_i \neq 0$.

Notice that we can also combine the ideas behind the h-compact and 3-compact representations to reduce both the sizes and number of terms. Working through the details of adding "$-h$" to the 3-compact representation, we find we must let n be the largest integer such that $x \geq ((3^n - 1)/2)h$ and set $y = x + ((3^n - 1)/2)h$. Furthermore, using ETRIPLEX, we compute

$$\mathfrak{a}[s_{i+1}]' = (\mu_i)\mathfrak{a}[s_i]^3 = ((\mu_i')\mathfrak{a}[s_i]) \left((\mu_i'')\mathfrak{a}[s_i]^2 \right)$$

for each iteration of the main while-loop. Thus, we have $\mathrm{O}(\Delta^{1/4}) < \mu_i', \mu_i'' < \mathrm{O}(\Delta^{3/4})$, and since $\mu_i = \mu_i'\mu_i''$, we see $\mathrm{O}(\Delta^{1/2}) < \mu_i < \mathrm{O}(\Delta^{3/2})$. So our choice of h needs to be increased to $h = \lceil (1/2)\log_2 \Delta \rceil$.

Let $y = \sum_{i=0}^{l} 3^{l-i} b_i$. We now derive bounds on the heights of the λ_i defined above. From (8) we have $\lambda_i = (L_i^3 \pi_i)/\pi_{i-1}^3$, which, when combined with (6) gives

$$\lambda_i < L_i^3 \left(\frac{17}{16} 2^{s_i}\right) \left(\frac{16\sqrt{\Delta}}{15 L_i} 2^{-s_{i-1}}\right)^3 < \frac{16^2 \cdot 17}{15^3} 2^{s_i - 3s_{i-1}} \Delta^{3/2} .$$

Since $s_i - 3s_i = b_i - h$ and $b_i \in \{0, \pm 1\}$ we have

$$\lambda_i < \frac{16^2 \cdot 17 \cdot 2}{15^3} \cdot 2^{-h} \Delta^{3/2} < \frac{11}{4} \Delta .$$

Now, since $\overline{\lambda_i} = (L_i^3 \overline{\pi}_i)/\overline{\pi}_{i-1}^3$ and $|\pi_i \overline{\pi}_i| = L_{i+1}$, we find

$$|\overline{\lambda_i}| = \left| \frac{L_i^3 \overline{\pi}_i}{\overline{\pi}_{i-1}^3} \right| = \frac{L_i^3 (L_{i+1}/\pi_i)}{(L_i/\pi_{i-1})^3} = \frac{L_i^3 L_{i+1} \pi_{i-1}^3}{L_i^3 \pi_i} = \frac{L_{i+1} \pi_{i-1}^3}{\pi_i}$$

and thus

$$|\overline{\lambda_i}| < L_{i+1} \left(\frac{17}{16} 2^{s_{i-1}}\right)^3 \left(\frac{16\sqrt{\Delta}}{15 L_{i+1}} 2^{-s_i}\right) \leq \frac{17^3 \cdot 2}{15 \cdot 16^2} \cdot 2^h \Delta^{1/2} < \frac{11}{4} \Delta . \qquad (9)$$

since $3s_{i-1} - s_i = h - b_i$ and $b_i \in \{0, \pm 1\}$. Thus, we find for a signed $3h$-compact representation that

$$H(\lambda_i) < \frac{11}{4} \Delta \qquad (10)$$

for $l - n < i \leq l$. Considering λ_0, we see

$$\frac{15 L_1}{16\sqrt{\Delta}} < \lambda_0 < \frac{17}{16} 2^{s_0} = \frac{17}{8} ,$$

as $s_0 = 1$, and so (10) holds for $i = 0$ as well. For $0 < i \leq l - n$ we have $H(\lambda_i) \in \mathrm{O}(\Delta^{3/2})$, but only for $l - n < 2 + \log_3 h$ terms.

We can now state the definition of a signed $3h$-compact representation. An algorithm to compute such representations is presented in Appendix C.

Definition 3. *For any θ such that $(\theta) = \mathfrak{a}[x] \in \mathcal{O}$, a* signed $3h$-compact representation *of θ is*

$$\theta = \prod_{i=0}^{l} \left(\frac{\lambda_i}{L_i^3}\right)^{b^{l-i}}$$

where the following properties are satisfied:

1. $l = \lceil \log_3 \log_2 \theta \rceil$.
2. $\lambda_i \in \mathcal{O}_\Delta$ *and* L_i *is an integer* $(0 \leq i \leq l)$.

3. $0 < L_i \leq \Delta^{1/2}$ *and* $H(\lambda_i) = \mathrm{O}(\Delta)$ *(*$0 \leq i \leq l$*).*
4. $\pi_j \in \mathcal{O}_\Delta$, $L_{j+1} = |N(\pi_j)|$,

$$\pi_j = \prod_{i=0}^{j} \left(\frac{\lambda_i}{L_i^3}\right)^{3^{j-i}} ,$$

π_j *generates a reduced ideal* $\mathfrak{b}_j$, *where* $\mathfrak{b}_0 = \mathfrak{a}[1]$ *and*

$$L_{i+1}^3 \mathfrak{b}_{i+1} = \lambda_{i+1} \mathfrak{b}_i^3 \quad (0 \leq i \leq l-1) .$$

Theorem 2. *Let* $\theta \in \mathcal{O}_\Delta$ *such that* $\mathfrak{a}[x] = (\theta)$ *for some* $x \in \mathbb{Z}^+$. *The number of bits in a signed* $3h$*-compact representation of* θ *is* $\mathrm{O}((\log_3 \log_2 \theta) \log_2 \Delta)$.

The proof is analogous to that of Theorem 1.

Going back to our running example, we find that a signed $3h$-compact representation of the fundamental unit $\eta_{410286423278424}$ requires 843 bits. Compared to the compact and h-compact representations respectively, the signed $3h$-compact representation saves us 30.7% and 13.7%. Again, note that the asymptotic size is not changed, but having $\log_3$ terms instead of $\log_2$, combined with the size reduction in terms, further improves the O-constant and the size in practice.

6 Using Larger Bases

Can we extend this idea further? What about a 4-compact, 5-compact, or higher representation?

For the time being, we will occupy ourselves with only signed 4- and 5-compact representations. We can compute a signed quaternary representation of an integer x using digit set $\{-1, 0, 1, 2\}$ or $\{-2, -1, 0, 1\}$. For the signed quinary representation we use $\{-2, -1, 0, 1, 2\}$. When using base 4, we require an algorithm which computes an ideal $\mathfrak{a}[4x]$ from an ideal $\mathfrak{a}[x]$ using w-near (f, p)-representations. An analogous algorithm is also required for base 5.

As with base 3, we consider signed $4h$-compact representations as follows. As in the base-3 case, we need to increase h by a further factor of $(1/4) \log_2 \Delta$ to $h = \lceil (3/4) \log_2 \Delta \rceil$. We also must compute the maximal n such that

$$\frac{x}{(4^n - 1)/3} \geq h,$$

and put $y = x + ((4^n - 1)/3)h$. We find that for most of the λ_i in the resulting algorithm

$$H(\lambda_i) < \frac{45}{8} \Delta^{5/4}$$

for a signed $4h$-compact representation. Furthermore, for $\theta \in \mathcal{O}_\Delta$ such that $\mathfrak{a}[x] = (\theta)$ $(x \in \mathbb{Z}^+)$, the total number of bits required to express θ as a signed $4h$-compact representation is $\mathrm{O}((\log_4 \log_2 \theta) \log_2 \Delta^{5/4})$.

The signed $4h$-compact representation of $\eta_{410286423278424}$ using digits $b_i \in \{-1, 0, 1, 2\}$ requires only 832 bits to store. Compared to the signed $3h$-compact

representation, this represents an additional savings of 1.3%. The signed $4h$-compact representation using $b_i \in \{-2, -1, 0, 1\}$ requires 843 bits. In general, it seems hard to predict *a priori* which signed base will produce a shorter signed compact representation.

If we set $h = \lceil \log_2 \Delta \rceil$, compute the maximal n such that

$$\frac{x}{(5^n - 1)/4} \geq h,$$

and put $y = x + ((5^n - 1)/4)h$, we can compute a signed $5h$-compact representation. Looking at the heights of the λ_i, we see that for most of the λ_i

$$H(\lambda_i) < \frac{47}{4}\Delta^{3/2}$$

and $\mathrm{O}((\log_5 \log_2 \theta) \log_2 \Delta^{3/2})$ bits are needed to store the total representation. The signed $5h$-compact representation of $\eta_{410286423278424}$ requires 875 bits, which is larger than the storage needed for the signed $4h$-compact representations.

At this point, we find the first indications that pursuing this idea to higher powers (i.e., 6-compact and higher representations) may not result in further memory savings. Unfortunately, the increase in size of the individual terms of the compact representations begins to dominate the savings from a decreased overall number of terms. In the remainder of this section, we look at an analytical argument to justify this claim. In the following section, we will present some calculations that confirm, numerically at least, that this analysis is valid.

To determine the overall expected size S_x of the signed base-x h-compact representation in bits, we multiply the base-2 logarithm of the $H(\lambda_i)$ bounds by the corresponding number of terms l. For most of these λ_i ($i \geq 2 + \log_x h$) the $H(\lambda_i)$ bounds are given by some constant B_x multiplied by $\Delta^{(x+1)/4}$. Expanding and converting the logarithms to base 2, we see that

$$\begin{aligned} S_x &= \log_2 \left(B_x \Delta^{(x+1)/4} \right) \cdot \log_x \log_2 \theta \\ &= \left(\frac{\log_2 B_x}{\log_2 x} \right) \log_2 \log_2 \theta + \left(\frac{x+1}{4 \log_2 x} \right) \log_2 \Delta \log_2 \log_2 \theta \,. \end{aligned} \tag{11}$$

The B_x values are given by

$$B_x = \max \left\{ \frac{16^{x-1} \cdot 17}{15^x}, \frac{17^x}{15 \cdot 16^{x-1}} \right\} 2^{\lfloor x/2 \rfloor} = \frac{17}{15} \max \left\{ \frac{16}{15}, \frac{17}{16} \right\}^{x-1} 2^{\lfloor x/2 \rfloor}$$

as 17/15, 16/15, and 17/16 are all greater than 1. Thus,

$$B_x < \frac{17}{15} \left(\frac{16}{15} \right)^{x-1} 2^{\lfloor x/2 \rfloor} < \frac{17}{15} \left(\frac{16}{15} \right)^{x-1} 2^{x-1} = \frac{17}{15} \left(\frac{32}{15} \right)^{x-1}$$

and $\log_2 B_x$ is of size $\mathrm{O}(x)$. Asymptotically then, the $(x+1)/(4 \log_2 x)$ coefficient will dominate this expression as the discriminant increases. Looking at Figure 1, we see this coefficient has a minimum between $x = 3$ and $x = 4$.

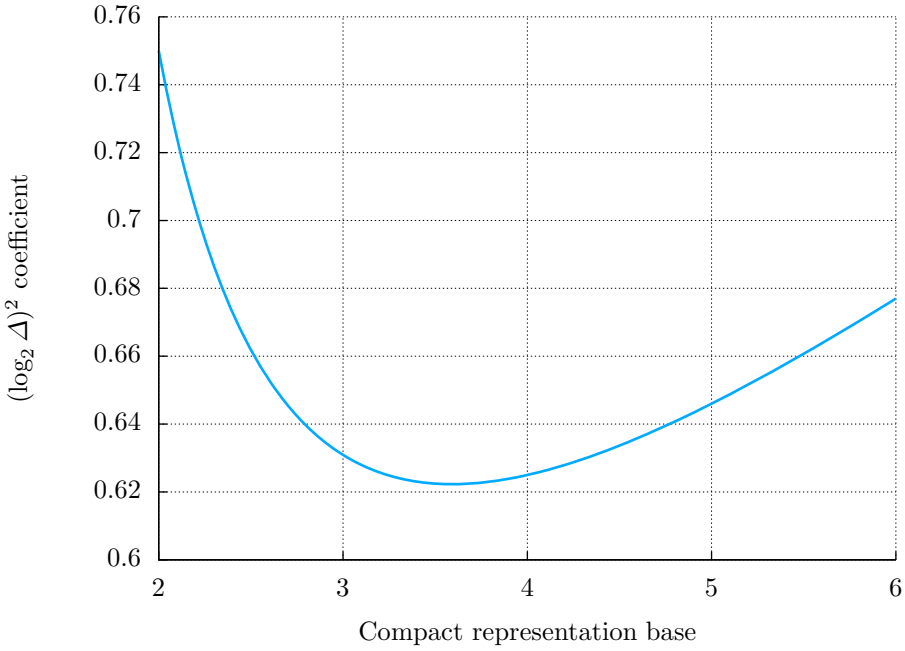

Fig. 1. Plot of $(x+1)/(4\log_2 x)$ coefficients

In this paper, we are most interested in computing compact representations where $\theta = \eta_\Delta$, the fundamental unit. As such, we can write

$$\log_x \log_2 \theta = \log_x \log_2 \eta_\Delta = \log_x \mathcal{R},$$

where $x \in \{2, 3, \ldots, 6\}$. Recall that we can loosely bound the regulator by $\sqrt{\Delta}$ and, after substitution, we are left with

$$l < \log_x \sqrt{\Delta} \tag{12}$$

as an upper bound on the number of terms in our various compact representations. Specializing (11) using (12), we find

$$S_x = \left(\frac{\log_2 B_x}{2\log_2 x}\right)\log_2 \Delta + \left(\frac{x+1}{8\log_2 x}\right)(\log_2 \Delta)^2 \ .$$

Again, asymptotically, the $(x+1)/(8\log_2 x)$ coefficient will dominate this expression as the discriminant increases and the minimum still occurs between $x = 3$ and $x = 4$. In fact, if we compare the two functions S_3 and S_4, we find that $S_4 < S_3$ for discriminants greater than $10^{16.5}$. In other words, for discriminants larger than about 16 decimal digits, the signed $4h$-compact representation is the most efficient one.

This conclusion supports our initial impression that base-5 and higher representations are not likely to produce shorter compact representations. From an analytic viewpoint, the trade-off between increasing the heights of the individual terms and gaining a representation with a fewer number of terms is no longer working in our favor. Because of this, we will not provide numerical results for the base-5 or higher compact representations in the next section.

7 Numerical Results

Since the preceding discussion only shows the savings in one particular case, we turn to some empirical results to further support our memory-saving claims. We calculated an approximation of the associated regulator for a random sampling of 28,000 discriminants evenly spread from decimal length 5 through 18, and for each discriminant used this to compute various compact representations of the fundamental unit. For each of the regular, h-, signed $3h$-, signed $4h$-, and signed $5h$-compact representations, we computed a best-fit regression line for the data, as well as provided distribution box plots,[1] a 95% confidence interval for our regression line, and a 95% prediction interval for further data points. Figure 2 shows a summary comparison of the average representation length for each discriminant length, along with the associated best-fit curves.

In the previous section, our analysis concluded that the signed $4h$-compact representation should be the most efficient for large enough discriminants. However, the numerical results that follow seems to show that the signed $4h$-compact representation is more efficient all the time. We initially speculated that this discrepancy is caused by our conservative bound on $H(\lambda_i)$. However, this only provides a piece of the answer. In our analysis, we assumed that each digit in the base-3 representation is a 1 and each digit in the base-4 representation is a 2 (or -2). Turning to a probabilistic argument, for a randomly selected number we would expect the proportions of the digits in its representation to approximately be equal. For example, for a signed base-3 representation, we would expect to see 0, 1, and -1 each roughly 33% of the time. If we take this into account, can we derive the following bound on $H(\lambda_i)$ for the signed base-3 case:

$$H(\lambda_i) = \max\left\{\frac{16^2 \cdot 17}{15^3},\ \frac{17^3}{15 \cdot 16^2}\right\}\left(\frac{2^{-1}}{3} + \frac{2^0}{3} + \frac{2^1}{3}\right)\Delta < \frac{43}{20}\Delta\ .$$

Similarly for the signed base-4 case, we have

$$H(\lambda_i) = \max\left\{\frac{16^3 \cdot 17}{15^4},\ \frac{17^4}{15 \cdot 16^3}\right\}\left(\frac{2^{-1}}{4} + \frac{2^0}{4} + \frac{2^1}{4} + \frac{2^2}{4}\right)\Delta^{5/4} < \frac{13}{5}\Delta^{5/4}\ .$$

Using these probabilistic bounds in S_3 and S_4, we find that S_4 is now strictly less than S_3 for discriminants larger than roughly 10^{14}.

We extended these empirical results further by using the series of discriminants from [10, Tbl. 7.8, p. 101], as well as the regulator approximations given there, to

[1] For each box plot, potential outliers have been marked with a "×" symbol.

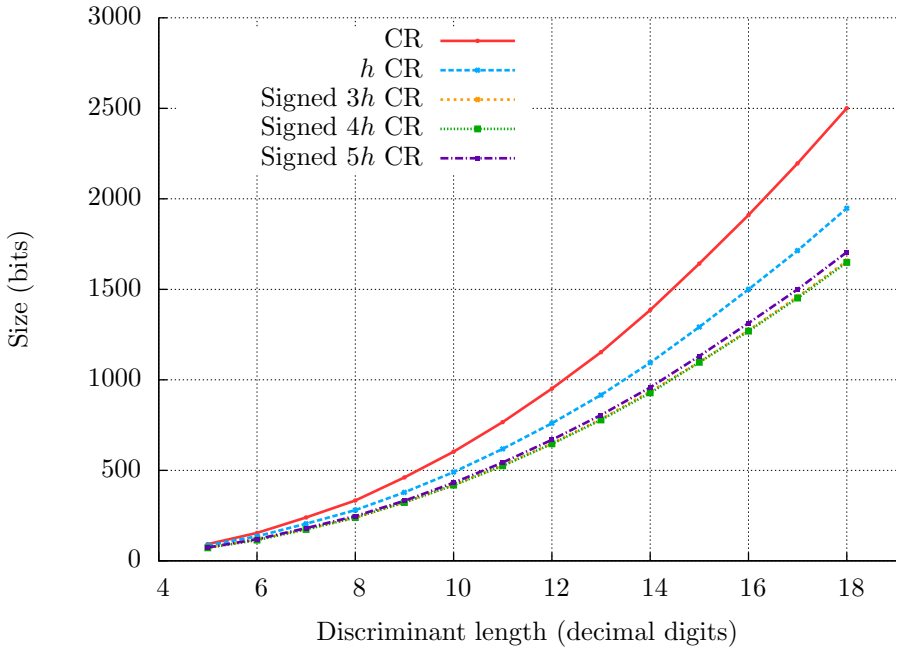

Fig. 2. Comparison of the sizes of some various compact representations for random discriminants (1,000 discriminant values for each length from 5–18)

produce the same variety of compact representations as above for the associated fundamental units. A comparison of the sizes of these representations is shown in Figure 3.

We must be cautious with these extended results. Because of the limited sampling, it is difficult to make a definitive claim on the relative efficiencies of the various h-compact representations at this point. For the majority of these larger discriminants, the signed $4h$-compact representation is the most efficient. However, for a given discriminant, the signed $3h$-compact representation may be just as, or even slightly more, efficient. With further measurements, we expect to find that the signed $4h$-compact representation is most efficient on average.

8 Future Directions

In this paper, we presented two substantial improvements that can be used together to reduce the sizes of a compact representations for certain quadratic integers. The first was noticing that the size of the individual compact representation terms could be reduced by a substantial factor. The second refinement was to notice that the overall number of terms could be reduced by computing larger giant steps on each iteration of the algorithm using bases larger than two. Asymptotically, the signed $4h$-compact representation results in the most

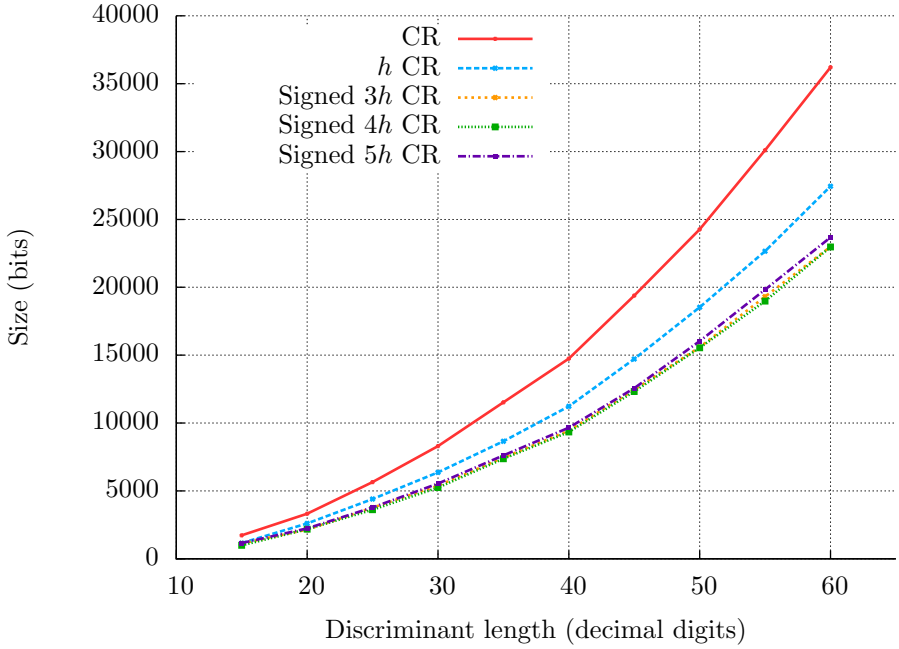

Fig. 3. Comparison of the sizes of some various compact representations for the series of discriminants presented in [10, Tbl. 7.8, p. 101]

efficient balance between larger individual compact representation terms and a reduced overall number of terms. Numerical testing supports this conclusion. In the large-discriminant tests we performed, we found overall memory savings of around 37% as compared to the standard compact representation.

There are other types of number representations we did not investigate which may lead to further memory savings for compact representations. For example, we could consider using a *non-adjacent form* (*NAF*) representation [19], a signed base-2 representation for which the average density of non-zero digits among all NAFs of a given length is approximately 1/3 [11, Thm. 3.29] as opposed to 1/2 for the regular binary representation. A *width-w NAF* (wNAF) uses odd digits less than 2^{w-1} in absolute value and has average density of non-zero digits $1/(w+1)$, so that NAF is a wNAF with $w = 2$. In terms of an overall length the NAF and wNAF representation of an integer is at most one more than the length of its binary representation.

Another example is the *double-base* representation of the integer x, given by

$$x = \sum_{i,j} b_{i,j} 2^i 3^j \ ,$$

where $b_{i,j} \in \{0, 1\}$. These representations only require $O(\log n / \log \log n)$ digits to store and near-canonic representations can be computed via a number of methods [1,5,6,8]. The advantage to this representation over a standard binary or ternary representation is that we require fewer terms. It could be quite beneficial if this numeric representation could be applied to create a double-base compact representation. First, by reducing the overall number of terms of the representation, we would reduce the overall storage requirements by moving from a compact to a 3-compact to a 4-compact representation. Furthermore, by restricting ourselves to the bases 2 and 3, we have the potential of avoiding the per-term expansion we encountered due to the increasing bound on $H(\lambda_i)$.

References

1. Avanzi, R., Dimitrov, V., Doche, C., Sica, F.: Extending scalar multiplication using double bases. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 130–144. Springer, Heidelberg (2006)
2. Buchmann, J., Thiel, C., Williams, H.C.: Short representation of quadratic integers, Mathematics and its Applications, vol. 325, pp. 159–185. Kluwer Academic Publishers, Amsterdam (1995)
3. Buchmann, J., Vollmer, U.: Binary Quadratic Forms, Algorithms and Computation in Mathematics, vol. 20. Springer (2007)
4. Cohen, H.: A Course in Computational Algebraic Number Theory, Graduate Texts in Mathematics, 4th edn., vol. 138. Springer, New York (2000)
5. Dimitrov, V., Imbert, L., Mishra, P.K.: Efficient and secure elliptic curve point multiplication using double-base chains. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 59–78. Springer, Heidelberg (2005)
6. Dimitrov, V.S., Jullien, G.A., Miller, W.C.: An algorithm for modular exponentiation. Information Processing Letters 66, 155–159 (1998)
7. Dixon, V., Jacobson Jr., M.J., Scheidler, R.: Improved exponentiation and key agreement in the infrastructure of a real quadratic field. In: Hevia, A., Neven, G. (eds.) LatinCrypt 2012. LNCS, vol. 7533, pp. 214–233. Springer, Heidelberg (2012)
8. Doche, C., Imbert, L.: Extended double-base number system with applications to elliptic curve cryptography. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 335–348. Springer, Heidelberg (2006)
9. Fung, G.W., Williams, H.C.: Compact representation of the fundamental unit in a complex cubic field (1991) (unpublished manuscript)
10. de Haan, R.: A fast, rigorous technique for verifying the regulator of a real quadratic field. Master's thesis, University of Amsterdam (2004)
11. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, New York (2004)
12. Hühnlein, D., Paulus, S.: On the implementation of cryptosystems based on real quadratic number fields (extended abstract). In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 288–302. Springer, Heidelberg (2001)
13. Imbert, L., Jacobson Jr., M.J., Schmidt, A.: Fast ideal cubing in imaginary quadratic number and function fields. Advances in Mathematics of Communications 4(2), 237–260 (2010)

14. Jacobson Jr., M.J., Scheidler, R., Williams, H.C.: The efficiency and security of a real quadratic field based key exchange protocol. In: Alster, K., Urbanowicz, J., Williams, H.C. (eds.) Public-Key Cryptography and Computational Number Theory, September 11-15 (2000); Walter de Gruyter GmbH & Co., Warsaw (2001)
15. Jacobson Jr., M.J., Scheidler, R., Williams, H.C.: An improved real quadratic field based key exchange procedure. J. Cryptology 19, 211–239 (2006)
16. Jacobson Jr., M.J., Williams, H.C.: Solving the Pell Equation. CMS Books in Mathematics. Springer (2009)
17. Lagarias, J.C.: Succinct certificates for the solvability of binary quadratic diophantine equations (extended abstract). In: Proc. 20th IEEE Symp. on Foundations of Computer Science, pp. 47–54 (1979)
18. Lagarias, J.C.: Succinct certificates for the solvability of binary quadratic diophantine equations. Tech. Rep. Technical Memorandum 81-11216-54, Bell Labs, 28 (1981)
19. Reitwiesner, G.W.: Binary arithmetic. Advances in Computers 1, 231–308 (1960)
20. Shanks, D.: The infrastructure of a real quadratic field and its applications. In: Proc. 1972 Number Theory Conference, University of Colorado, Boulder, pp. 217–224 (1972)
21. Silvester, A.K.: Improving regulator verification and compact representations in real quadratic fields. Ph.D. thesis, University of Calgary, Calgary, Alberta (2012)

A EWNEAR

We present a version of EWNEAR [16, Alg. 12.1, pp. 286 and 457] that also works when $k > w$. As the new version merely adds some key values which allow the determination of a relative generator, the proof of correctness of EWNEAR will remain unchanged.

Algorithm 3: EWNEAR

Input: $(\mathfrak{b}, d, k)$, w, p, where $(\mathfrak{b}, d, k)$ is a reduced (f, p) representation of some $\mathcal{O}$-ideal $\mathfrak{a}$. Here $\mathfrak{b}[Q/r, (P+\sqrt{D})/r]$, where $P+\lfloor\sqrt{D}\rfloor \geq Q$, $0 \leq \lfloor\sqrt{D}\rfloor - P \leq Q$.

Output: $(\mathfrak{c}, g, h)$ a w-near $(f + 9/8, p)$ representation of $\mathfrak{a}$ and a, b, where $\varkappa = (a + b\sqrt{D})/Q$ and $\mathfrak{c} = \varkappa\mathfrak{b}$.

1: **case 1:** $k < w$
2: Put $B_{-2} = 1$, $B_{-1} = 0$.
3: Find $s \in \mathbb{Z}^{\geq 0}$ such that $2^s Q \geq 2^{p+4}$. Put $Q_0 = Q$, $P_0 = P$, $M = \lceil 2^{p+s-k+w} Q_0/d \rceil$, $Q_{-1} = (D - P^2)/Q$, $T_{-2} = -2^s P_0 + \lfloor 2^s\sqrt{D}\rfloor$, $T_{-1} = 2^s Q_0$, $i = 1$.
4: **while** $T_{i-2} \leq M$ **do**
5: $q_{i-1} = \lfloor (P_{i-1} + \lfloor\sqrt{D}\rfloor)/Q_{i-1}\rfloor$
6: $P_i = q_{i-1}Q_{i-1} - P_{i-1}$
7: $Q_i = Q_{i-2} - q_{i-1}(P_i - P_{i-1})$
8: $T_{i-1} = q_{i-1}T_{i-2} + T_{i-3}$
9: $B_{i-1} = q_{i-1}B_{i-2} + B_{i-3}$
10: $i \leftarrow i + 1$
11: **end while**
12: Put $e_{i-1} = \lceil 2^{p-s+3} T_{i-3}/Q_0 \rceil$

13: **if** $de_{i-1} \le 2^{2p-k+w+3}$ **then**
  14: Put $\mathfrak{c} = [Q_{i-2}/r, (P_{i-2} + \sqrt{D})/r]$, $e = e_{i-1}$, $a = (T_{i-3} - \lfloor 2^s\sqrt{D}\rfloor)/2^s$, $b = B_{i-3}$.
15: **else**
  16: Put $\mathfrak{c} = [Q_{i-3}/r, (P_{i-3} + \sqrt{D})/r]$, $e = \lceil 2^{p-s+3}T_{i-4}/Q_0\rceil$, $a = (T_{i-4} - \lfloor 2^s\sqrt{D}\rfloor)/2^s$, $b = B_{i-4}$.
17: **end if**
18: Find t such that

$$2^t < \frac{ed}{2^{2p+3}} \le 2^{t+1} .$$

19: Put

$$g = \left\lceil \frac{ed}{2^{p+t+3}} \right\rceil , \quad h = k + t .$$

20: **end case**
21: **case 2:** $k > w$
  22: Put $B^*_{-2} = 1$, $B^*_{-1} = 0$.
  23: Put $s = p + 4$, $Q^*_0 = Q$, $P^*_0 = P$, $M^* = d2^{k-w+4}$, $Q^*_1 = (D - P^2)/Q$, $T^*_{-2} = 2^sQ^*_0$, $T^*_{-1} = 2^sP^*_0 + \lfloor 2^s\sqrt{D}\rfloor$, and $i = 1$.
  24: **while** $T^*_{i-2} < Q^*_iM^*$ **do**
    25: $q^*_i = \lfloor (P^*_{i-1} + \lfloor\sqrt{D}\rfloor)/Q^*_i\rfloor$
    26: $P^*_i = q^*_iQ^*_i - P^*_{i-1}$
    27: $Q^*_{i+1} = Q^*_{i-1} - q^*_i(P^*_i - P^*_{i-1})$
    28: $T^*_{i-1} = q^*_iT^*_{i-2} + T^*_{i-3}$
    29: $B^*_{i-1} = q^*_iB^*_{i-2} + B^*_{i-3}$
    30: $i \leftarrow i + 1$
  31: **end while**
  32: Put $q^*_i = \lfloor (P^*_{i-1} + \lfloor\sqrt{D}\rfloor)/Q^*_i\rfloor$, $P^*_i = q^*_iQ^*_i - P^*_{i-1}$, $e = \lceil T^*_{i-2}/2Q^*_i\rceil$, $e' = \lceil T^*_{i-3}/2Q^*_{i-1}\rceil$, $j = 3$.
  33: **while** $e' \ge d2^{k-w+3}$ **do**
    34: $e \leftarrow e'$
    35: $e' \leftarrow \lceil T_{i-2-j}/2Q^*_{i-j}\rceil$
    36: $j \leftarrow j + 1$
  37: **end while**
  38: Find t (t') such that

$$2^{t-1} \le \frac{e}{8d} < 2^t . \quad \left(2^{t'-1} \le \frac{e'}{8d} < 2^{t'} .\right)$$

  39: Put $\mathfrak{c} = [Q^*_{i-j+3}/r, (P^*_{i-j+3} + \sqrt{D})/r]$, $g = \lceil 2^{p+3+t}d/e\rceil$, $h = k - t$, $a = (T^*_{i-2} - B^*_{i-2}\lfloor 2^s\sqrt{\Delta}\rfloor)/2^s$, $b = B^*_{i-j+2}$.
40: **end case**

B ETRIPLEX

In order to implement a triple-and-add algorithm to use compact representations, a precision analysis for cubing an (f,p)-representation is required. To this end, we include the following theorem.

Theorem 4 ([16, Thm. 11.2, p. 268]). *Let $(\mathfrak{b}, d', k')$ be an (f', p) representation of an $\mathcal{O}_\Delta$-ideal $\mathfrak{a}$. If $d'^3 \leq 2^{3p+1}$, put $d = \lceil d'^3/2^{2p} \rceil$ and $k = 3k'$. If $2^{3p+1} < d'^3 \leq 2^{3p+2}$, put $d = \lceil d'^3/2^{2p+1} \rceil$ and $k = 3k' + 1$. If $d'^3 > 2^{3p+2}$, put $d = \lceil d'^3/2^{2p+2} \rceil$ and $k = 3k' + 2$. Then $(\mathfrak{b}^3, d, k)$ is an (f,p) representation of the product ideal $\mathfrak{a}^3$, where $f = 1 + 3f' + 3f'^2/2^p + f'^3/2^{2p}$.*

Proof. Let $\mathfrak{b} = \theta\mathfrak{a}$ for $\theta \in \mathbb{K}$. By the definition of d in the theorem, it is easy to see that $2^p < d \leq 2^{p+1}$. From the definition of an (f,p) representation, we know

$$\left| \frac{2^{p-k'}\theta}{d'} - 1 \right| < \frac{f'}{2^p},$$

and rearranging this inequality gives

$$\frac{d'}{2^p}\left(1 - \frac{f'}{2^p}\right) < \frac{\theta}{2^{k'}} < \frac{d'}{2^p}\left(1 + \frac{f'}{2^p}\right).$$

As $2^p < d' \leq 2^{p+1}$ and $f'/2^p < 1/16$, we have

$$\frac{d'}{2^p}\left(1 - \frac{f'}{2^p}\right) > 1 \cdot \left(1 - \frac{1}{16}\right) > 0 \quad \text{and} \quad \frac{d'}{2^p}\left(1 + \frac{f'}{2^p}\right) < 2 \cdot \left(1 + \frac{1}{16}\right) < 4,$$

and thus

$$\left(1 - \frac{f'}{2^p}\right)^3 < \frac{2^{3(p-k')}\theta^3}{d'^3} < \left(1 + \frac{f'}{2^p}\right)^3.$$

If we set $f^* = 3f' + 3f'^2/2^p + f'^3/2^{2p}$ then

$$1 - \frac{f^*}{2^p} = 1 - \frac{3f'}{2^p} - \frac{3f'^2}{2^{2p}} - \frac{f'^3}{2^{3p}} < 1 - \frac{3f'}{2^p} + \frac{3f'^2}{2^{2p}} - \frac{f'^3}{2^{3p}} = \left(1 - \frac{f'}{2^p}\right)^3$$

and

$$\left(1 + \frac{f'}{2^p}\right)^3 = 1 + \frac{3f'}{2^p} + \frac{3f'^2}{2^{2p}} + \frac{f'^3}{2^{3p}} = 1 + \frac{3f' + 3f'^2/2^p + f'^3/2^{2p}}{2^p} = 1 + \frac{f^*}{2^p}.$$

Hence

$$1 - \frac{f^*}{2^p} < \frac{2^{3p-3k'}\theta^3}{d'^3} < 1 + \frac{f^*}{2^p}. \tag{13}$$

Now suppose that $d'^3 \leq 2^{3p+1}$. Since $d = d'^3/2^{2p} + \epsilon$ for $0 \leq \epsilon < 1$, (13) becomes

$$1 - \frac{f^*}{2^p} < \frac{2^{p-k}\theta^3}{d - \epsilon} < 1 + \frac{f^*}{2^p}$$

and as $d - \epsilon = d(1 - \epsilon/d)$,

$$\left(1 - \frac{\epsilon}{d}\right)\left(1 - \frac{f^*}{2^p}\right) < \frac{2^{p-k}\theta^3}{d} < \left(1 - \frac{\epsilon}{d}\right)\left(1 + \frac{f^*}{2^p}\right).$$

Looking at the right-hand side of this inequality, $(1 - \epsilon/d) < 1$ so

$$\left(1 - \frac{\epsilon}{d}\right)\left(1 + \frac{f^*}{2^p}\right) < 1 + \frac{f^*}{2^p} < 1 + \frac{1}{2^p} + \frac{f^*}{2^p} = 1 + \frac{f}{2^p};$$

considering the left-hand side, $\epsilon < 1$ and $2^p < d$ so $2^p\epsilon < d$. Rearranging this inequality gives $1 - 1/2^p < 1 - \epsilon/d$ and thus

$$1 - \frac{f}{2^p} = 1 - \frac{1}{2^p} - \frac{f^*}{2^p} < \left(1 - \frac{1}{2^p}\right)\left(1 - \frac{f^*}{2^p}\right) < \left(1 - \frac{\epsilon}{d}\right)\left(1 - \frac{f^*}{2^p}\right).$$

It follows that

$$\left|\frac{2^{p-k}\theta^3}{d} - 1\right| < \frac{f}{2^p}$$

and $(\mathfrak{b}^3, d, k)$ is an (f, p) representation of $\mathfrak{a}^3$, where $\mathfrak{b}^3 = \theta^3\mathfrak{a}^3$. The theorem follows by applying similar arguments when $2^{3p+1} < d'^3 \leq 2^{3p+2}$ and when $d'^3 > 2^{3p+2}$. □

In practice, ideal cubing could be accomplished by a square and a multiplication. Another option is to use the dedicated ideal cubing algorithm NUCUBE [13, Alg. 4] described by Imbert, Jacobson, and Schmidt. An extended version of this algorithm for w-near representations is presented in [7]. Finally, ETRIPLEX is obtained by extending the algorithm further to also produce the corresponding relative generator. For a complete description, see [21, §§ 5.3].

C Algorithm to Compute Signed 3*h*-Compact Representations

We present here an algorithm to compute a signed $3h$-compact representation, based on the ideas of Section 5.

Algorithm 5: 3HCRAX

Input: x, p, where $x \in \mathbb{Z}^+$ and $2^p > 11.2x \max\{16 \log_2 x\}$.

Output: $(\mathfrak{a}[x], d, k)$, (m_i, n_i), and L_i, where $(\mathfrak{a}[x], d, k)$ is an x-near (f, p) representation of $\mathfrak{a} = (1)$ with $f < 2^{p-4}$, (m_i, n_i) are pairs of integers, and $L_i \in \mathbb{Z}^+$ for $i = 0, 1, \ldots, l$ where l is such that $x = \sum_{j=0}^{l} 3^{l-j} b_j$ and $b_0 \neq 0$, $b_j \in \{0, 1, 2\}$.

1: Put $h = \lceil (1/2) \log_2 \Delta \rceil$ and compute the maximal n such that

$$\frac{x}{(3^n - 1)/2} \geq h,$$

and put $y = x + ((3^n - 1)/2)h$.

2: Compute the signed ternary representation of y with

$$y = \sum_{i=0}^{l} 3^{l-i} b_i \text{ and } b_0 \neq 0, b_i \in \{-1, 0, 1\} \quad (1 \leq i \leq l).$$

3: Put

$$Q = r, \quad P = r \left\lfloor \frac{\lfloor \sqrt{\Delta} \rfloor - r + 1}{r} \right\rfloor + r - 1, \quad (\mathfrak{b}, d, k) = ([Q, P], 2^p + 1, 0),$$

$s = b_0$, $L_0 = 1$, and $i = 0$.

4: Put $((\mathfrak{b}_0, d_0, k_0), m_0, n_0) = \text{EWNEAR}((\mathfrak{b}, d, k), s, p)$.

5: **while** $i < l - n$ **do**

6: Put $L_{i+1} = N(\mathfrak{b}_i)$ and

$$((\mathfrak{b}_{i+1}, d_{i+1}, k_{i+1}), m_{i+1}, n_{i+1}) = \text{ETRIPLEX}((\mathfrak{b}_i, d_i, k_i), s, p).$$

7: Set $s \leftarrow 3s + b_{i+1}$.

8: **if** $b_{i+1} \neq 0$ **then**

9: Put $N = N(\mathfrak{b}_{i+1})$ and set

$$((\mathfrak{b}_{i+1}, d_{i+1}, k_{i+1}), m'_{i+1}, n'_{i+1}) \leftarrow \text{EWNEAR}((\mathfrak{b}_{i+1}, d_{i+1}, k_{i+1}), s, p).$$

10: Set $(m_{i+1}, n_{i+1}) \leftarrow \text{IMULT}(m_{i+1}, n_{i+1}, m'_{i+1}, n'_{i+1}, N)$.

11: **end if**

12: Set $i \leftarrow i + 1$.

13: **end while**

14: **while** $i < l$ **do**

15: Put $L_{i+1} = N(\mathfrak{b}_i)$ and

$$((\mathfrak{b}_{i+1}, d_{i+1}, k_{i+1}), m_{i+1}, n_{i+1}) = \text{ETRIPLEX}((\mathfrak{b}_i, d_i, k_i), s, p).$$

16: Set $s \leftarrow 3s + b_{i+1} - h$.

17: Put $N = N(\mathfrak{b}_{i+1})$ and set

$$((\mathfrak{b}_{i+1}, d_{i+1}, k_{i+1}), m'_{i+1}, n'_{i+1}) \leftarrow \text{EWNEAR}((\mathfrak{b}_{i+1}, d_{i+1}, k_{i+1}), s, p).$$

18: Set $(m_{i+1}, n_{i+1}) \leftarrow \text{IMULT}(m_{i+1}, n_{i+1}, m'_{i+1}, n'_{i+1}, N)$.

19: Set $i \leftarrow i + 1$.

20: **end while**

21: Put $L_{l+1} = N(\mathfrak{b}_l)$ and $(\mathfrak{a}[x], d, k) = (\mathfrak{b}_l, d_l, k_l)$.