グレブナー基底を用いた
二次体での素イデアル分解アルゴリズム

伊藤充洋

2022 年 5 月 31 日

1　中間発表会までの研究と今回の研究の目的

中間発表会までの研究は，代数体の整数環における様々な計算，特に素イデアル分解の計算をコンピュータ上で簡単に行えるようにすることを目的として行った．手計算が難しい整数環での計算を気軽にコンピュータに実行させることができれば，様々な研究において「その結果を観察して具体例について調べる」という行為に時間を使えるようになり，非常に役に立つと考えたからである．

二次体の整数環において次のような計算を行う関数をプログラミング言語 C++ で実装し，適切に動作することを確認した．以下では特に整数環が $\mathbb{Z}[\sqrt{d}]$ と書ける場合について，すなわち $d \equiv 2, 3 \bmod 4$ である場合について述べる．また，$x \in \mathbb{Q}(\sqrt{d})$ に対して，$\mathrm{N}(x)$ で x の $\mathbb{Q}$ 上のノルムを表す．

- 剰余の計算を行う関数：
  整数環の元 x, r が与えられた時，$x \equiv a + b\sqrt{d} \pmod{r}$ を満たし，a, b が十分小さい値であるものを求める．時間計算量は $O(\log |\mathrm{N}(r)|)$.
- イデアルの生成元を求める関数：
  与えられたイデアルの生成系から，それと同じイデアルを生成する二項を見つける．与えられるイデアルの生成系を F，その要素の 1 つを x として，時間計算量は $O((|\mathrm{N}(x)|^2 + |F|\,|\mathrm{N}(x)|) \log |\mathrm{N}(x)|)$.
- イデアルが，ある元を含むか判定する関数：
  与えられたイデアルと整数環の要素 x について，イデアルがその元を含むか判定する．イデアルの生成元二項がすでに求められている時，時間計算量は $O(|\mathrm{N}(x)| \log |\mathrm{N}(x)|)$.
- 素イデアル分解を行う関数：
  イデアルが与えられた時，その素イデアル分解を計算する．計算量は，イデアルの生成元の一方を x として，$O(|\mathrm{N}(x)|^2 \log^2 |\mathrm{N}(x)|)$.

これらを C++ で実装したことで，二次体における様々な計算がコンピュータを用いて簡単に実行できるようになった．この成果を「第 19 回　高校生・高専生科学技術チャレンジ（JSEC2021）（ `https://manabu.asahi.com/jsec/` ）」に提出し，審査の結果，敢闘賞を頂いた．

一方で，計算量が非常に大きく，あまり大きな値に対して計算ができない（家庭用 PC などで普通に実行していると，数時間待っても終了しない場合がある）という課題が残った．イデアルの生成元が 10^3 程度になると一時間かけても終了しない場合があり，10^2 程度でも数十秒〜数分かかる，といった具合である．これは「具体例の観察を容易にし，他の研究に役立てる」という目的に対しては大きな問題である．小さなケースに対してしか計算が実行できないのでは，観察の対象が減ってしまうからだ．そこで，中間発表会から成果発表会までの間は，これらをより高速に実行できるようにすることを目指した．

2　中間発表会以降の研究

上で述べたように，中間発表会までの成果では大きな値に対する計算が非常に遅く，研究の目的を達成するには高速化が不可欠だったため，中間発表会以降は二次体での計算をより高速にすることを目的として研究を行った．

そこで，整数環における計算を整数係数の多項式環での計算に帰着することを考えた．$\mathbb{Z}[\sqrt{d}] \cong \mathbb{Z}[x]/\langle x^2 - d \rangle$ であり，素イデアルの対応を考えれば，$\mathbb{Z}[\sqrt{d}]$ において $\langle a + b\sqrt{d} \rangle$ を含む素イデアルを計算する代わりに

$\mathbb{Z}[x]$ において $\langle x^2-d, a+bx\rangle$ を含む素イデアルを列挙できれば十分だからである．考察を進めた結果，参考文献 [4] や [5] で述べられているグレブナー基底や多項式の因数分解に関するアルゴリズムを用いて，素イデアル分解アルゴリズムを高速化することに成功した．以下では，必要となる strong グレブナー基底の性質について触れたあと，考案した素イデアル分解のアルゴリズムについて述べる．

2.1 strong グレブナー基底について

体上の多項式環における計算ではグレブナー基底が有用だが，これは一般の環上の多項式環では直接扱うことができない．そこで，体上の多項式環でなくても良い性質を持つ strong グレブナー基底というものを考える．ここでは，一変数多項式 $f \neq 0$ に対してその最高次の項を $\mathrm{LT}(f)$ と書くことにする．

定義 2.1.1（[4] Definition 2,Remark 3） R を PID とする．有限集合 $G = \{g_1, \ldots, g_n\} \subset R[x]$ は，次を満たす時，イデアル $I \subset R[x]$ の strong グレブナー基底と呼ばれる．
（条件）
任意の $f \in I$ について，$f = \sum_i g_i h_i$ となる $h_1, \ldots, h_n \in R[x]$ であって，次を全て満たすものが存在する．

- $\mathrm{LT}(f) = \mathrm{LT}(g_i h_i)$ となるような i が存在する．
- 任意の $k \neq i$ について，$g_k h_k$ は f より次数が小さい．

イデアルの生成系が与えられた時，その strong グレブナー基底を計算するアルゴリズムについては，文献 [4] の Algorithm 2 を参考にした．イデアルの strong グレブナー基底は，その定義より次の性質を持つ．

命題 2.1.2 イデアル $I \subset R[x]$ とその strong グレブナー基底 G，及び $f \in R[x]$ を考える．この時，次に述べるような剰余を取る操作を行える限り繰り返した結果を h とする．すると，$f \in I \Leftrightarrow h = 0$ である．

- 操作：$\mathrm{LT}(f)$ を $\mathrm{LT}(g)$ が割るような $g \in G$ が存在する時，f を $f - \frac{\mathrm{LT}(f)}{\mathrm{LT}(g)} \cdot g$ で置き換える．

証明 $h = 0 \Rightarrow f \in I$ は明らかなので，逆を示す．$\deg f$ についての帰納法を用いる．$f = 0$ の時は明らか．そうでない時，定義 2.1.1 の条件より，$\mathrm{LT}(f)$ を $\mathrm{LT}(g)$ が割るような $g \in G$ は存在する．この時，$f - \frac{\mathrm{LT}(f)}{\mathrm{LT}(g)} \cdot g$ の次数は f より小さいため，帰納的に命題が成り立つ． □

命題 2.1.3（簡約化） イデアル $I \subset R[x]$ とその strong グレブナー基底 $G = \{g_1, \ldots, g_n\}$ について，$\mathrm{LT}(g_n)$ を $\mathrm{LT}(g_i)$ が割るような $i < n$ が存在するとする．この時，g_n を除いた $\{g_1, \ldots, g_{n-1}\}$ も I の strong グレブナー基底である．

証明 簡単のため，$i = 1$ とする．$g_n = g_1 q + r$ となる多項式 $q = \frac{\mathrm{LT}(g_n)}{\mathrm{LT}(g_1)}, r$ を取る．この時，$r \in I$ で $\deg r < \deg g_n$ である．よって，定義 2.1.1 の条件によって，$r = \sum_{1 \leq j \leq n-1} g_j v_j$ と書け，$v_1 + q$ を改めて v_1 と書けば $g_n = \sum_{1 \leq j \leq n-1} g_j v_j$ とできる．なお，この和は定義 2.1.1 の条件を満たす（$j \geq 2$ に対して $\deg g_j v_j \leq \deg r < \deg g_n$）．よって，$f \in I$ を条件を満たすように $\sum_{1 \leq j \leq n} g_j u_j$ と書けば，これは $\sum_{1 \leq j \leq n-1} g_j (u_j + u_n v_j)$ と等しい．この和について，最高次数をとる項がちょうど 1 つであることを示せば，$\{g_1, \ldots, g_{n-1}\}$ も strong グレブナー基底であることが言える．$\deg g_n u_n = \deg f$ である場合には，$g_n = \sum_{1 \leq j \leq n-1} g_j v_j$ という和が最高次数をとる項もちょうど 1 つであったことから条件を満たす．そうでない場合の成立は明らかである． □

これらの性質により，多項式のイデアルにある多項式が属するか，といった判定は strong グレブナー基底

がわかっている状況では簡単に行うことができる．また，簡約化が行えることでグレブナー基底を簡単な形に変形できる場合がある．以上の性質を実際の計算に活用した．

2.2 整数環における計算の高速化

以下では簡単のため，二次体 $\mathbb{Q}(\sqrt{d})$ とその整数環について，整数環が $\mathbb{Z}[\sqrt{d}]$ と書ける場合についてのみ（すなわち，$d \equiv 2, 3 \pmod 4$ である場合のみ）議論する．

$\mathbb{Z}[\sqrt{d}]$ におけるイデアル $\langle a + b\sqrt{d} \rangle$ について議論する代わりに，$\mathbb{Z}[x]$ における $\langle x^2 - d, a + bx \rangle$ について計算することを考える．ここで，PID 上の多項式環の strong グレブナー基底を計算するアルゴリズムを文字式のまま実行し，計算すると，次が示せる．

命題 2.2.1 $\mathbb{Z}[x]$ において，$\langle x^2 - d, a + bx \rangle$ の strong グレブナー基底の 1 つとして $\{x^2 - d, gx + (at + bsd), \frac{a^2 - b^2 d}{g}\}$（ただし $g = \gcd(a, b)$ とし，s, t は $as + bt = g$ を満たす整数の組とする）がとれる．
特に，$\gcd(a, b) = 1$ の時，$\{x + (at + bsd), a^2 - b^2 d\}$ がとれる．

証明 前半についての証明は場合分けが非常に煩雑であり，また，その上でアルゴリズムに沿った計算を行うのにもかなりの記述量となってしまう．そこで，ここではイデアルとしての一致のみ示す．（証明のあとに，アルゴリズムに沿った計算の例を挙げる．）

- $gx + (at + bsd), \frac{a^2 - b^2 d}{g} \in \langle x^2 - d, a + bx \rangle$ であること：
  $(sx + t)(a + bx) - bs(x^2 - d) = gx + (at + bsd)$,
  $\frac{a - bx}{g}(a + bx) + \frac{b^2}{g}(x^2 - d) = \frac{a^2 - b^2 d}{g}$ であるから成立．
- $a + bx \in \langle x^2 - d, gx + (at + bsd), \frac{a^2 - b^2 d}{g} \rangle$ であること：
  $\frac{b}{g}(gx + (at + bsd)) + s(\frac{a^2 - b^2 d}{g}) = bx + \frac{abt + a^2 s}{g} = bx + a\frac{as + bt}{g} = a + bx$ となり成立．

したがって，$\langle x^2 - d, a + bx \rangle = \langle x^2 - d, gx + (at + bsd), \frac{a^2 - b^2 d}{g} \rangle$ である．

後半については，前半の主張において $g = 1$ とすることで strong グレブナー基底として $\{x^2 - d, x + (at + bsd), a^2 - b^2 d\}$ が取れることが分かる．さらに，$x^2 - d$ は $x + (at + bsd)$ で簡約化できることから，結局命題の主張が従う． □

例 2.2.2（命題 2.2.1 の例） $d = -5, a = 6, b = 4$ の場合の計算例を挙げる．計算には [4] の Algorithm 2 を用いた．初め，入力は $I = \langle x^2 + 5, 4x + 6 \rangle$ より，$G = \{x^2 + 5, 4x + 6\}, P = \{\mathrm{spoly}(x^2 + 5, 4x + 6), \mathrm{gpoly}(x^2 + 5, 4x + 6)\} = \{6x - 20, x^2 + 5\}$ である．なお，以下での計算において，元のアルゴリズムでは P に加えている多項式も，G による剰余が 0 であれば結局 P から除くこととなるため，初めから P に加えていない．初めの P からも $x^2 + 5$ は除いておく．

1. $h = 6x - 20$ とする．
   - P から h を除き $\{\}$ とする．
   - h を G による剰余に置き換える．$\mathrm{LT}(h)$ を $\mathrm{LT}(f)$ が割るような $f \in G$ はないので次へ進む．
   - G に h を加え $\{x^2 + 5, 4x + 6, 6x - 20\}$ とする．
   - P に $\mathrm{spoly}(h, f), \mathrm{gpoly}(h, f)$ $(f \in G)$ を加え $\{58, 2x - 26\}$ とする．
2. $h = 58$ とする．
   - P から h を除き $\{2x - 26\}$ とする．

- h を G による剰余に置き換える．$\mathrm{LT}(h)$ を $\mathrm{LT}(f)$ が割るような $f \in G$ はないので次へ進む．
- G に h を加え $\{x^2+5, 4x+6, 6x-20, 58\}$ とする．
- P に $\mathrm{spoly}(h,f), \mathrm{gpoly}(h,f)$ $(f \in G)$ を加え $\{2x-26, 2x+90, 2x-200\}$ とする．

3. $h = 2x-26$ とする．
   - P から h を除き $\{2x+90, 2x-200\}$ とする．
   - h を G による剰余に置き換える．$\mathrm{LT}(h)$ を $\mathrm{LT}(f)$ が割るような $f \in G$ はないので次へ進む．
   - G に h を加え $\{x^2+5, 4x+6, 6x-20, 58, 2x-26\}$ とする．
   - P に $\mathrm{spoly}(h,f), \mathrm{gpoly}(h,f)$ $(f \in G)$ を加え $\{2x+90, 2x-200\}$ とする（後から削除される多項式は初めから追加していないため，ここでは多項式を一切追加していない）．
4. $h = 2x+90, 2x-200$ とすると G による剰余が 0 となるため，これらを P から削除する．
5. 以上により P が空となったため，アルゴリズムを終了する．

これにより得られた strong グレブナー基底 G を簡約化して，$\{x^2+5, 2x-26, 58\}$ を得る．命題 2.2.1 において $g=2, s=1, t=-1$ とすれば，$\{x^2-d, gx+(at+bsd), \frac{a^2-b^2d}{g}\} = \{x^2+5, 2x-26, 58\}$ となり，確かに一致していることが確かめられる．

以上の事実を用いて，単項生成イデアルの素イデアル分解を $O(\sqrt{N})$（N は生成元のノルム）程度で実行するアルゴリズムを作る．アイデアとしては，$\langle a+b\sqrt{d}\rangle = \langle \gcd(a,b)\rangle\langle a'+b'\sqrt{d}\rangle$ という形に変形し，各々を分解することである．これは，$\langle p\rangle$（p は素数）の分解が $\mathbb{F}_p$ での因数分解の計算に帰着できること，$\langle a+b\sqrt{d}\rangle$ $(\gcd(a,b)=1)$ に対応する多項式環のイデアルの strong グレブナー基底が簡単な形で書けること，を用いてそれぞれ計算できる．以下でまずそれぞれの分解方法を述べ，最後にまとめて「単項生成イデアルの素イデアル分解のアルゴリズム」として提示する．

まずは $\langle n\rangle$ という形のイデアルの分解である．これには Cantor-Zassenhaus アルゴリズム（参考文献 [5]6.6 節）を用いる．

命題 2.2.3 次のアルゴリズムは整数 n に対し，十分高い確率で $\Theta(\sqrt{n})$ 時間で $\langle n\rangle$ の素イデアル分解を計算する．

Input: 整数 n
Output: $\langle n\rangle$ の素イデアル分解

1. 整数 n を素因数分解する．
2. 各素因数 p に対して x^2-d の $\mathbb{F}_p$ での因数分解 $\prod_i f_i(x)$ を求める．
3. $\langle p\rangle = \prod_i \langle p, f_i(\sqrt{d})\rangle$ であるから，n の素因数全体にわたるこれらの積を出力とする．

証明 まず，因数分解に用いる Cantor-Zassenhaus アルゴリズムは確率的アルゴリズムである．[5]6.6 節命題 6.28 及び 6.29 より，一回の因数分解の試行が 1/2 程度の確率で成功することがわかり，したがって，例えば 30 回試行を繰り返すことで非常に高い確率で因数分解が成功する（これで分解できない場合は非常に高い確率で既約である）．また一度の試行は，多項式の次数が定数（今回は 2 次）である時，$\Theta(\log p)$ で行うことができる．よって，ここでは $\mathbb{F}_p$ での二次多項式の因数分解が $\Theta(\log p)$ で計算できるとして解析を進める．

さて，n の素因数分解に素朴な $\Theta(\sqrt{n})$ 時間のアルゴリズムを適用することで，1. は $\Theta(\sqrt{n})$ 時間で終了する．2. については，n の素因数の種類数が $\log n$ で抑えられることを考えれば，上の解析と併せて $O(\log^2 n)$ で実行できる．したがって，このアルゴリズムは $\Theta(\sqrt{n})$ 時間で実行可能である． □

命題 2.2.4 次のアルゴリズムは互いに素な整数 a, b に対し，$\langle a+b\sqrt{d}\rangle$ の素イデアル分解を $\Theta(\sqrt{N})$ 時間で計算する．ただし $N = \left|a^2 - b^2 d\right|$ はノルムの絶対値である．

Input: 整数 a, b
Output: $\langle a+b\sqrt{d}\rangle$ の素イデアル分解

1. 素因数分解 $N = \prod_i p_i^{e_i}$ を求める．
2. $\prod_i \langle a+b\sqrt{d}, p_i\rangle^{e_i}$ を出力する．

証明 素因数分解に素朴な試し割りのアルゴリズムを用いることで $\Theta(\sqrt{N})$ 時間で計算が終了するのは明らかなので，これが正しく素イデアル分解を計算していることを示す．$\{x+s, N\}$ が $\langle a+b\sqrt{d}\rangle$ の strong グレブナー基底となるような整数 s を取っておく．

$\mathbb{Z}[\sqrt{d}]/\langle a+b\sqrt{d}\rangle \cong \mathbb{Z}[x]/\langle a+bx, x^2-d\rangle \cong \mathbb{Z}[x]/\langle x+s, N\rangle \cong \mathbb{Z}/N\mathbb{Z}$ である．イデアルの対応を考えると，$\mathbb{Z}[\sqrt{d}]$ において $\langle a+b\sqrt{d}\rangle$ を含むイデアルは $\mathbb{Z}$ において $N\mathbb{Z}$ を含むイデアルに対応する．よって，N の素因数 p に対する $p\mathbb{Z}$ という素イデアルに対応する $\mathbb{Z}[\sqrt{d}]$ の素イデアルを考えれば良い．素因数 p を一つ固定し，$p^e \mid N$ だが $p^{e+1} \nmid N$ となる e をとる．この時，$\mathbb{Z}$ において $p^e\mathbb{Z}$ は $N\mathbb{Z}$ を含むが $p^{e+1}\mathbb{Z}$ は $N\mathbb{Z}$ を含まない．したがって，$\mathbb{Z}[\sqrt{d}]$ における $\langle a+b\sqrt{d}\rangle$ の素イデアル分解には，$p\mathbb{Z}$ に対応する素イデアルがちょうど e 個現れる．さて，同型による対応を追うと，$p\mathbb{Z}$ に対応する $\mathbb{Z}[\sqrt{d}]$ のイデアルは $\langle p, a+b\sqrt{d}\rangle$ である．以上より，上のアルゴリズムは正しく素イデアル分解を計算する． □

以上 2 つを用いて，次のような単項生成イデアルの素因数分解アルゴリズムができる：

Input: 整数 a, b
Output: $\langle a+b\sqrt{d}\rangle$ の素イデアル分解

1. $g = \gcd(a, b)$ を求める．
2. $\langle g\rangle, \left\langle \frac{a}{g} + \frac{b}{g}\sqrt{d}\right\rangle$ をそれぞれ素イデアル分解し，積をとったものを出力とする．

これは，g の計算を $O(\log\min(|a|, |b|))$ 時間，残りの計算を $\Theta\left(\sqrt{g} + \sqrt{\frac{N}{g^2}}\right)$ 時間で行う（N は生成元のノルム）．よって，全体で $O(\log\min(|a|, |b|) + \sqrt{N})$ 時間で素イデアル分解の計算が可能である．これは，以前の方法と比べ大幅に高速になっている．さらに，これらを用いて二項生成の場合の素イデアル分解も計算できることがわかった．

命題 2.2.5 $\mathbb{Z}[\sqrt{d}]$ の二項生成イデアル $\langle a_1+b_1\sqrt{d}, a_2+b_2\sqrt{d}\rangle$ $(a_1, b_1, a_2, b_2 \in \mathbb{Z}, \gcd(a_1, b_1, a_2, b_2) = 1)$ に対応する $\mathbb{Z}[x]$ のイデアル $\langle x^2-d, a_1+b_1x, a_2+b_2x\rangle$ について，その strong グレブナー基底の 1 つとして $\langle x+s, t\rangle$ $(s, t \in \mathbb{Z})$ という形のものが取れる．

証明 $g_1 = \gcd(a_1, b_1), g_2 = \gcd(a_2, b_2)$ とする．単項生成の場合の strong グレブナー基底として計算したものを用いれば，

$$\langle x^2-d, a_1+b_1x, a_2+b_2x\rangle = \langle x^2-d, g_1x+c_1, n_1, g_2x+c_2, n_2\rangle$$

となる c_1, n_1, c_2, n_2 が存在し，計算できる．いま，$\gcd(g_1, g_2) = 1$ であるから，$g_1u + g_2v = 1$ を満たす $u, v \in \mathbb{Z}$ が取れる．この u, v を用いれば，このイデアルは $x+s$ という形の多項式を要素に持つことがわかる（$s = c_1u + c_2v$ とすれば良い）．よって，$x^2-d, a_1+b_1x, a_2+b_2x$ などを $x+s$ で割ったあまりの整数に置き換え（一次モニックによる剰余は必ず整数である），現れた整数たちの最大公約数を求めて t とすることで，

$\langle x^2-d, a_1+b_1x, a_2+b_2x\rangle = \langle x+s, t\rangle$ となる．$\{x+s, t\}$ は $\langle x+s, t\rangle$ の strong グレブナー基底になっているため，この命題が成り立つ．□

この命題の s, t は，証明から分かるように高速に計算可能である．したがって，あとは単項生成の場合と同じ議論を適用することで，$\gcd(a_1, b_1, a_2, b_2) = 1$ であるような場合に素イデアル分解が可能となった．$\gcd(a_1, b_1, a_2, b_2) \neq 1$ である場合には，その部分を分離し，$\langle \gcd(a_1, b_1, a_2, b_2)\rangle$ を別途分解すれば良い．したがって，二項生成までのイデアルの素イデアル分解が非常に高速に計算可能になった．（なお，$|n_1|, |n_2|$ がそれぞれ生成元のノルムの絶対値であり，かつ t が n_1, n_2 を割ることを考えれば，$t < \min(|n_1|, |n_2|)$ であるから，計算量はノルムの平方根で抑えられる．）

また，素イデアル分解の他にも，グレブナー基底を計算したことで，様々な計算が高速に実行可能となった．これまでに述べたものを含め，この研究を通して考案したアルゴリズムを以下に列挙する．

- 高々二項生成のイデアルが与えられた時，その素イデアル分解を計算する．
- イデアルの生成系を与えた時，それと同じイデアルを生成する二項を見つける．
  二項生成のイデアルの素イデアル分解の際に行ったことと同様のアルゴリズムを繰り返し実行すれば，

$$\langle x^2-d, a_1+b_1x, a_2+b_2x, \dots\rangle = \langle \gcd(a_1, b_1, a_2, b_2, \dots)\rangle \cdot \langle x+s, t\rangle$$

  となる整数 s, t, n が見つけられる．すると $\langle a_1\sqrt{d}+b_1, a_2\sqrt{d}+b_2, \dots\rangle = \langle \gcd(a_1, b_1, a_2, b_2, \dots)\rangle \cdot \langle \sqrt{d}+s, t\rangle$ となり，素イデアル分解が計算できる．
- イデアルの所属判定，一致判定など．
  多項式環でのイデアルの strong グレブナー基底がわかっているため，非常に高速に計算できる．

以上を C++ で実装したコードは GitHub 上で公開している．リンクはこちら：
`https://github.com/mitsu-a/KdEi-quadratic-field`

2.3 実行時間の比較

計算量のオーダーが大幅に改善されたため，前回のアルゴリズムと今回新たに作ったアルゴリズムの実行時間を測定した．以下では全て $d = -5$ の場合に計算を行う．方法は以下の通り．

- テストケースの生成
  単項生成の場合は 2 つ，二項生成の場合は 4 つの整数をランダムに生成し，各々 $\langle a+b\sqrt{-5}\rangle$ などと見なす．この時，それらの整数の絶対値が x 以下になるように取る，という条件を以下で $\mathrm{P}(x)$ と書く．（テストケースの生成には C++ の標準ライブラリに実装されているメルセンヌ・ツイスタ std::mt19937_64 などを用いた．）
- 実行時間の測定
  以上で生成したケースを素イデアル分解し，その実行時間を測定する．この際，実行時間の揺れを減らすため複数ケースを実行した．

以上の方法を用いて，まずは旧アルゴリズム・新アルゴリズムに対して同じケースを与え，実行時間を測定した．次の表に示すように，旧アルゴリズムでは 2 時間かかっても終わらないケースがあるのに対し，新アルゴリズムは全て 0.03 秒程度で終了した．このことから，以前の研究から大幅に改善していることがわかる．

表 1　単項生成の場合における，旧アルゴリズムとの実行時間の比較

ケース数	制約	旧アルゴリズム	新アルゴリズム
20	P(10)	0.29 秒	0.018 秒
20	P(100)	4081 秒	0.021 秒
20	P(1000)	>7200 秒	0.024 秒
20	P(10000)	>7200 秒	0.031 秒

表 2　二項生成の場合における，旧アルゴリズムとの実行時間の比較

ケース数	制約	旧アルゴリズム	新アルゴリズム
20	P(10)	0.67 秒	0.018 秒
20	P(100)	89 秒	0.021 秒
20	P(1000)	>7200 秒	0.024 秒
20	P(10000)	>7200 秒	0.030 秒

次に，新アルゴリズムの性能を測るため，新アルゴリズムのみでより多くのより大きなケースについて実行時間を測定した．次に示すように，10^5 ケースを試行しても 20 秒程度で実行が終了している．1 ケースごとにかかっている時間は 10^{-4} 秒程度であり，実験・観察のために使うには十分に高速だろう．

表 3　単項・二項生成の場合における新アルゴリズムの実行時間

ケース数	制約	単項生成の場合	二項生成の場合
10^5	P(10)	1.24 秒	0.40 秒
10^5	P(10^2)	1.51 秒	0.44 秒
10^5	P(10^3)	1.71 秒	0.46 秒
10^5	P(10^4)	2.14 秒	0.51 秒
10^5	P(10^5)	4.37 秒	0.57 秒
10^5	P(10^6)	19.64 秒	0.60 秒
10^5	P(10^7)	21.07 秒	0.64 秒

2.4　結論と今後の課題

今回の研究で，二次体においては，様々な計算が非常に高速に行えるようになった．2.3 節でも述べた通り，10^7 オーダーの整数に対して単項・二項生成イデアルを考えても，十分高速に計算が終了するようになったのは大きな成果である．これだけ大きな値に対しても計算ができれば，「具体例の観察を容易にし，他の研究に役立てる」という目的は十分果たすことが可能だろう．

一方で，課題もいくつか残っている．

まず，オーバーフローの懸念である．厳密な解析ができていないため，どのくらいの値を入力として与えた時にオーバーフローが起こるか明確には分かっていない．ただ，特にイデアルの生成系から同じイデアルを生

成する二つの要素を見つけるアルゴリズムにおいて，オーバーフローが起こりやすそうだと考えている．これは，二元一次不定方程式を繰り返し解き，それらいくつかの積を取ることで n 元一次不定方程式の整数解を求める，という手法を取っているためである．もちろんどのような実装でも限界はあるため，非常に大きな値に対してアルゴリズムを実行する際は必ず整数を扱う型として 64bit 整数よりも大きな型を使うなどの対策が必要となるが，できる限り改善したいと考えている．

次に，より一般の代数体の整数環での計算を行えるようにすることである．今回は二次体の整数環に限定して様々な実装を行ったが，より計算を行える範囲が広がれば，具体的な研究に使える幅が広がるだろう．今回も用いた strong グレブナー基底などを活用し，どうすれば一般の代数体でも計算が可能となるか考えていきたい．

2.5 謝辞

中間発表会までの研究を含め，この研究を進めるにあたっては研究部の皆様に大変お世話になりました．成果発表会までの一年間アドバイザーを担当して頂いた東京理科大学理学部第一部応用数学科の石原侑樹助教には，グレブナー基底など多項式を用いて計算するというアイデアに対し [4][5] の資料の紹介をして頂くなど，研究を進めていくための様々なサポートをして頂きました．また，同じ数理科学グループに属するメンバー・アドバイザーの皆様には，毎月の研究において多くの刺激やアドバイスを頂きました．本当にありがとうございました．さらに，学校法人角川ドワンゴ学園及びその研究部の運営に携わっている方々には，このような研究の機会及び同じように研究をする中高生と交流する機会を提供して頂いたことに心から感謝を申し上げます．最後に，主催の株式会社朝日新聞社・株式会社テレビ朝日の皆様をはじめとした，JSEC2021 を開催し，審査を行なってくださった皆様に感謝いたします．

参考文献

[1] 雪江明彦，『整数論 1 初等整数論から p 進数へ』，日本評論社，2013
[2] 雪江明彦，『整数論 2 代数的整数論の基礎』，日本評論社，2013
[3] M.F.Atiyah and I.G.MacDonald （新妻弘訳），『可換代数入門』，共立出版，2006
[4] Christian Eder and Tommy Hofmann, ”Efficient Gröbner Bases Computation over Principal Ideal Rings,” *Journal of Symbolic Computation*, volume 103(2021), pp.1-13.
[5] 野呂正行，『計算機代数入門』，2005（ `http://www.math.kobe-u.ac.jp/Asir/ca.pdf` ）
[6] 伊藤充洋，『二次体における効率的な素イデアル分解のアルゴリズム』（JSEC2021 へ提出した内容.）