

Number Field Sieve の円分数の素因数分解への応用

立教大学理学部 木田 祐司

これは 森本・木田・山崎 上智大学数学講究録 No.42 円分数の素因数分解 (その4) 1999年7月、に収録 (pp154-195) した解説を修正したものである。記録はほとんど更新されているが注釈を施すにとどめ、原文はできるだけもとのままにする。

Number Field Sieve (数体ふるい法) は Pollard によって考案され、最初は $2(2^{128} + 1) = (2^{43})^3 + 2$ のように係数の小さな多項式の特特殊値 (ある整数における値) として表される合成数を分解するものであった。

しかし、勝手な合成数を与えてそれを特殊値とする (係数の大きな) 多項式を作った場合にも、同様に計算量が小さいと評価されることから一般の数の素因数分解にも使われるようになった。これを General Number Field Sieve (GNFS) といい、元来のものを Special Number Field Sieve (SNFS) という。GNFS を真に実用的な方法とするアイディアは Adleman によって出された。

以上の事情については前著

森本・木田・小林 円分数の素因数分解 (その3) 上智大学数学講究録 35(1992)

においてすでに述べた。その翌年には Pollard をはじめ NFS の発展に寄与した人々による総まとめの文献が出版されている。

A.K. Lenstra and H.W. Lenstra (Eds.): *The development of the number field sieve*, Lecture Notes in Mathematics 1554 (1993).

そして 1996 年には 130 桁の数が GNFS によって分解された。これは数の特殊性を利用しない (できない) 一般の数の素因数分解の最高記録であって¹ GNFS がこのくらいの大きさの数に対しては最強であることを立証している。

さて我々の円分数の素因数分解においては SNFS が特に有効である。そこで第1節では SNFS について簡単に復習し、実例をあげる。

次に第2節では GNFS について実験結果を踏まえて解説する。NFS はその計算の流れが二次ふるい法とほとんど同じであるので前著の“複数次多項式二次ふるい法”の項を合わせて読みたい。重複する部分は意識的、無意識的に省いている。GNFS で目新しいのは代数的数の平方根の計算である。第3節では多くのページを割いてこれを解説している。

我々の採用した GNFS 法は今ではやや“古典的”になってしまった。最近ではかなりの部分が新しい技術に置きかえられている。第4節ではこの第二世代ともいえるべき最近の GNFS 法を簡単に紹介する。

¹1999年には 155 桁の数が分解された。

1 SNFS

Special Number Field Sieve は、ターゲットの合成数 N が係数の小さな多項式 $f(x)$ と整数 M により $f(M)$ が N の倍数となっている場合を扱う。

1.1 SNFS 概論

$f(x)$ の根の1つを θ とし、 $K = \mathbb{Q}(\theta)$ とする。
仮定として次を置く

- $\mathbb{Z}[\theta]$ は UFD(素元分解環) である。

$N = f(M)$ の大きさに応じて \mathbb{Z} の素数の集合 F と O_K の素元と単数の集合 G を適当にとる。

a, b を互いに素な有理整数で $a + bM$ は F -smooth とする。つまり

$$(1-1) \quad a + bM = \prod_{p \in F} p^{e(p)}$$

であり、かつ $a + b\theta$ は G -smooth とする。

$$(1-2) \quad a + b\theta = \prod_{\pi \in G} \pi^{e(\pi)}$$

と完全に素元分解されるものとする。

このようなペア (a, b) を $\#F + \#G$ 個より多く集める。すると各べき指数を mod 2 で考えたベクトル

$$((e(p) \bmod 2)_{p \in F}, (e(\pi) \bmod 2)_{\pi \in G})$$

は線形従属となり、適当に組み合わせると、つぎのような関係式が出る。

$$(1-3) \quad \prod (a_i + b_i M) = \left(\prod_{p \in F} p^{e(p)} \right)^2$$

と \mathbb{Z} で平方数になり、かつ

$$(1-4) \quad \prod (a_i + b_i \theta) = \left(\prod_{\pi \in G} \pi^{e(\pi)} \right)^2$$

と O_K で平方数になる。

$$(1-5) \quad \prod (a_i + b_i \theta) = s(\theta)^2, \quad s(x) \in \mathbb{Z}[x]$$

であったとしてこれを準同型

$$\begin{aligned}\phi : \mathbf{Z}[\theta] &\longrightarrow \mathbf{Z}/N\mathbf{Z} \\ g(\theta) &\mapsto g(M) \bmod N\end{aligned}$$

で写せば

$$(1-6) \quad \left(\prod_{p \in F} p^{e(p)} \right)^2 \equiv s(M)^2 \bmod N$$

となり、ふるい法の目標である $x^2 \equiv y^2 \bmod N$ 型の合同式が得られる。

まとめると、SNFS の原理は、

$$\begin{aligned}\text{環準同型 } \phi : \mathbf{Z}[\theta] &\longrightarrow \mathbf{Z}/N\mathbf{Z} \\ g(\theta) &\mapsto g(M) \bmod N\end{aligned}$$

による次の図式を2通りに進んで、その分解の違いを利用することと言える。この図式は $\mathbf{Z}/N\mathbf{Z}$ で考えれば可換だが、 \mathbf{Z} では可換とは限らない。

$$\begin{array}{ccc} a + b\theta & \longrightarrow & O_K \text{での分解} \\ \phi \downarrow & & \downarrow \phi \\ a + bM & \longrightarrow & \mathbf{Z} \text{での分解} \end{array}$$

そこで、すべきことは $a + b\theta$ と $a + bM$ が同時に smooth な a, b のペアをたくさん探すこととなる。

”ふるい”系の素因数分解法ではこの smooth な数をたくさん見つけるところが実行時間の主要な部分となる。高速にするためにはこの検査の対象となる数の絶対値を小さくするのが非常に有効である。Number Field Sieve では、 $a + b\theta$ と $a + bM$ という二つの対象があって、前者は $f(x)$ の次数の増加に比例して絶対値が増加し、後者はその逆になる。そこで両者の中間を探ると最適な次数があって（幸いにも）計算量が優れたものとなったのである。

1.2 SNFS の記録

我々のターゲットである円分数は $x^m - 1$ 型の多項式が元になっているので SNFS には非常になじみが良い。

このプロジェクトで行われた SNFS による素因数分解で一番桁数が大きいものは

$$M(92\ 941) = \frac{941^{46} + 1}{941^2 + 1} \quad (131 \text{ 桁})$$

である²。

これに用いた多項式は

$$\begin{cases} f(x) = 941X^5 + 1 \\ M = 941^9 \end{cases}$$

である³。したがって実際には 137 桁の数 $941^{46} + 1$ を分解している。

factor base は

$$\begin{cases} \text{有理的 factorbase を} & 229,370 \text{ 個} \\ \text{代数的 factorbase を} & 370,096 \text{ 個} \\ \text{二次指標を} & 214 \text{ 個} \end{cases}$$

の計 599,680 個をとった。

a の範囲は $-1, 200,000$ から $1, 200,000$ である。 b は 1 から $600,000$ である。large primes は有理数側、代数的数側それぞれ $88 \cdot 10^6$ と $155 \cdot 10^6$ まで許容している。

ふるいにかかった時間は PentiumII-450MHz で約 400 時間である。

これで集まったデータの個数は

$$\begin{cases} \text{FF} = 307,550 & \text{FP} = 556,813 \\ \text{PF} = 923,407 & \text{PP} = 1,680,199 \end{cases}$$

の合計 3,467,969 個であった⁴。FP, PF, PP を組み合わせることにより 323,015 個の FF が新たに生成された。合計 641,968 個の FF の中から組み合わせの個数が少ないものを 599,680 個選んで行列を作った。

この 599,680 次行列は 0 でない成分が非常に少ない、いわゆる疎な行列であるので intelligent GAUSS 消去法 (の前半) により 86,976 次の密な行列に縮小できた。

この縮小行列を GAUSS 消去して 2,533 個の解を得た。

一連の行列の計算には PentiumII-450MHz で

- 行列の縮小のための関係式の計算に 13 時間
- 関係式から実際に縮小行列を作成することに 26 時間
- 密行列の GAUSS 消去に 44 時間

合計 83 時間かかっている。

なお 86,976 次行列はディスク内で 902Mbytes を占める。

²このプロジェクトでは 2001 年に 200 桁の数を分解した。

³係数の 941 は SNFS としては大きい部類であり、係数がすべて一桁の場合の SNFS に比べるとかなり難しくなっている。

⁴FF,FP,PF,PP などの用語については後述する。

これらの解から 2^{30} 程度の大きさの素数の 32 乗を 2,681 個使った Couveignes の方法 (後述) で平方根を求めた。平方根の計算は一回につき 12 時間かかる。2,533 個の解のうち 3 つ目で自明でない分解を得た⁵。

結果として N は

$$P1 = 556798537281030298582279043972701973(36 \text{桁})$$

$$P2 = 12366444553823418456266302194267876265216715606386$$

$$8919884010298515358001173970029394468858210197(96 \text{桁})$$

という二つの素数の積であった。

世界記録は以下の通りである。

1999 年 4 月 チーム Cabal(Stefania Cavallar, Bruce Dodson, Arjen Lenstra, Paul Leyland, Walter Lioen, Peter Montgomery, Herman te Riele, Paul Zimmermann) は

$N = (10^{211} - 1)/9$ (211 桁) を次の多項式を用いて素因数分解した。

$$\begin{cases} f(X) = 10X^6 - 1 \\ M = 10^{35} \end{cases}$$

factor base には有理数側、代数的数側ともに 2^{24} 以下の素数をとる⁶。large primes は有理数側、代数的数側それぞれ $500 \cdot 10^6$ と $600 \cdot 10^6$ まで許容している。

ふるいは $|a| \leq 6,000,000$ $0 < b \leq 18,000,000$ の範囲で行っている。集められたデータは合計 56,394,064 個に上る。

連立方程式は疎行列のまま Block Lanczos 法を用いる。Cray C90 で 121 時間かかったという⁷。

結果 N は 93 桁と 118 桁の二つの素数 $P1, P2$ の積であった。 $P1$ は次の通り。

$$P1 = 69262455732438962066278232267733671113810848258828$$

$$1739734375570506492391931849524636731866879$$

⁵各解について計算は独立しており、各解が自明でない分解を与える確率は $1/2$ 以上あるので 5 台も平行して実行すれば十分である。

⁶およそ 1,080,000 個

⁷64bit マシンなので 64 個以下の解が求まる。この場合はちょうど 64 個見つかった。

2 GNFS

$\mathbb{Z}[\theta]$ が UFD でない場合も扱おうというのが General NFS である⁸。

円分数の素因数分解では SNFS が非常に有効なため、GNFS が真に有効な場合はまだ少ない。しかしたとえば 200 桁の円分数の分解で、80 桁分の素因数は分かっているが残りの 120 桁は合成数であることは分かるものの楕円曲線法などではそれ以上分解ができない場合などには力を発揮することが期待される。

2.1 GNFS 概論

簡単のため $O_K = \mathbb{Z}[\theta]$ を仮定する。これが成立しない場合でも若干の補正で以下の議論はそのまま有効になる。補正の方法は後に述べる。

一般に整数環 O_K では素元分解ができない。しかし素イデアル分解は可能である。

そこで O_K については有理素数の集合 F と素イデアルの集合 G を適当にとる。さらに結果を平方数にするために何個かの平方剰余記号の集合 H を付け加える。

a, b を互いに素な有理整数で $a + bM$ は F -smooth とする。つまり

$$(2-1) \quad a + bM = \prod_{p \in F} p^{e(p)}$$

であり、かつ単項イデアル $[a + b\theta]$ は G -smooth とする⁹。つまり

$$(2-2) \quad [a + b\theta] = \prod_{\mathcal{P} \in G} \mathcal{P}^{e(\mathcal{P})}$$

と完全に素イデアル分解されるものとする。このとき $\chi \in H$ に関する値を次のようにおく。

$$\chi(a + b\theta) = (-1)^{e(\chi)}$$

ただし $a + b\theta$ は χ の modulus とは互いに素であるとする。このためには modulus を large primes の limit よりも大きくとる必要がある。

このようなペア (a, b) を $\#F + \#G + \#H$ 個より多く集める。すると各べき指数を mod 2 で考えたベクトル

$$\left((e(p) \bmod 2)_{p \in F}, (e(\mathcal{P}) \bmod 2)_{\mathcal{P} \in G}, (e(\chi))_{\chi \in H} \right)$$

は一次従属となり、適当に組み合わせると、つぎのような関係式が出る。

$$(2-3) \quad \prod (a_i + b_i M) = \left(\prod_{p \in F} p^{e(p)} \right)^2$$

⁸現在では special と general の区別はこうではなく、係数の絶対値が小さな多項式があらかじめ与えられている場合を special、そうでなく多項式を探さねばならない(そして見つかる多項式も係数の絶対値が大きい)場合を general というのが普通のようなものである。

⁹ここでは区別をはっきりさせるため単項イデアルは生成元を $[\]$ で囲んで表すことにする。

と \mathbf{Z} で平方数になり、かつ

$$(2-4) \quad \prod [a_i + b_i \theta] = \left(\prod_{\mathcal{P} \in G} \mathcal{P}^{e(\mathcal{P})} \right)^2$$

と O_K で平方イデアルになり、かつ

$$(2-5) \quad \chi \left(\prod (a_i + b_i \theta) \right) = 1 \quad \text{for } \forall \chi \in H$$

となる。

(2-4), (2-5) 式は $\prod (a_i + b_i \theta)$ が高い確率で O_K の平方数になっていることを示している。もし実際に

$$(2-6) \quad \prod (a_i + b_i \theta) = s(\theta)^2, \quad s(x) \in \mathbf{Z}[x]$$

であったとすれば後は Special NFS の場合とまったく同様である。

以上での最大の問題点は (2-6) において平方根 $c + d\theta$ の計算をいかにして記憶容量も時間も少なく済ませることができるかである。これは後に述べる。

たとえ UFD であってもそのことを用いないで、この節の方法を適用するのが良い。イデアルの生成元と単数を求める必要がなく、プログラム作りは非常に簡明になる。

2.2 多項式の選択

ターゲットの合成数 N と多項式 $f(x)$ と $\text{mod } N$ の“根” M の間には次の関係があった。

$$f(M) \equiv 0 \pmod{N}$$

$f(x)$ の次数は N が数十桁から二百桁程度ならば 3 から 5 にとるのが良い。後述の Multiple 型の GNFS ならば次数 2 の多項式を 2 個使うことにより百桁くらいでも有効になる。ここでは多項式を 1 個使った場合を述べる¹⁰。

2.2.1 係数を小さくする

$f(x)$ にまず課せられる条件は $a + b\theta$ のノルムが小さいことである。

$$f(x) = c_d x^d + c_{d-1} x^{d-1} + \dots + c_1 x + c_0$$

とすると

$$c_d \text{Norm}(a + b\theta) = (-b)^d f(a/(-b)) = c_d a^d + c_{d-1} a^{d-1} (-b) + \dots + c_1 a (-b)^{d-1} + c_0 (-b)^d$$

¹⁰added on Jan.2002: 多項式の選択は非常に重要である。しかし本質的にここに記述されているような brute force 的な方法しかない。想像されるよりずっと多くの時間を良い条件の多項式の選択にかけるべきである。

であるのでこれを小さくするには、係数の絶対値を小さくすることが有効であろう¹¹。制約は

$$f(M) = c_d M^d + c_{d-1} M^{d-1} + \dots + c_1 M + c_0 \equiv 0 \pmod{N}$$

である。 N を M 進数として展開すれば $|c_i| < M$ にはできるから、係数の絶対値を均等に小さくするには

$$M \sim N^{1/(d+1)}$$

とすれば良い。

もし大半の場合に $a > rb (\exists r > 1)$ が成立しているならノルムの式を見ると

$$|c_d| \sim |c_{d-1}|/r \sim |c_{d-2}|/r^2 \sim \dots \sim |c_0|/r^d$$

が成り立っているのが望ましいことになる。全体でこれを満たすようにとすることは難しいが上位4つの係数に限定して探索の時間をかければかなり良いものが見つかる¹²。

このときは

$$|c_d| \sim |c_{d-1}|/r \sim |c_{d-2}|/r^2 \sim |c_{d-3}|/r^3 \quad \text{かつ} \quad |c_{d-3}| \sim M$$

である。よって

$$M \sim (r^3 N)^{1/(d+1)}$$

となる。実際の探索では先に c_d を決める。

$$c_d \sim (N/r^{3d})^{1/(d+1)}$$

そして M を

$$(N/c_d)^{1/d}$$

が一番近い整数として定める。これから他の係数 c_i は自動的に定まる。これで所望の大きさに収まっていなければ別の c_d に変えて探索を繰り返すことになる。

いくつか望ましい c_d が見つければ次の条件を課すことになる。係数の小ささが失われないくらいの範囲で M を動かして条件を満たすものを探すことになる。

2.2.2 適当な実根を持たせる

$a + b\theta$ のノルムを小さくするには $f(a/(-b))$ の絶対値を小さくすることが有効。 a, b が動く範囲である $[-H_a, H_a] \times [1, H_b]$ で考えると、

$$f(x) \text{ に実根があってその中に } \pm H_a/H_b \text{ に近いものがある}$$

ことが望ましいことになる。

ただし実根があって $\pm H_a/H_b$ の数倍に収まっていれば結果に差は認められなかった。ここを数値化して次の Knuth-Schroepel 関数値と組み合わせる適当な式は今のところ見当がつかない。

¹¹厳密には必要でも十分でもないが。

¹²この辺は原文をかなり修正した。

2.2.3 Knuth-Schroeppel 関数値を大きくする

ふるいの効率は二次ふるい法と同様に Knuth-Schroeppel 関数によって推定することができる。それは

$$\sum_{P \in G} \frac{\log P}{P-1}, \quad \text{ただし } P \text{ はイデアル } \mathcal{P} \text{ に含まれる素数とする}$$

である。これが大きいほど smooth data を多く集められるということになる。

この値を多項式の係数の簡単な性質から判定することはおそらく不可能であろう。少なくとも今のところは多項式の分解 (= 素数の素イデアル分解) から実際に計算するしかない。

実際には factor base の元の個数が数万から数十万にもなるのですべてを計算していたのでは多くの多項式を試すことができない。そこで数百個の \mathcal{P} についての計算で済ますことにする。

実験結果はこの因子が大きく影響することを示している。

2.3 factor base の決定

素数の集合 F は小さい方から順に $\#F$ 個の素数を集めるだけである。

素イデアルの集合 G は素数 q を小さい方から順にとり、その上にある 1 次の素イデアルを $\#G$ 個集める。 $\#F, \#G$ の具体的な値は次の小節で与える。

素数 q の上にある 1 次の素イデアルは $f(x)$ を $\text{mod } q$ で因数分解したときの 1 次因子に対応する¹³。具体的には 1 次因子 $x - s$ には q と $\theta - s$ で生成された O_K の素イデアル $Q(q, s)$ が対応する。

$$f(x) \equiv (x - s)g(x) \pmod{q}, \quad \text{ただし } g(s) \not\equiv 0 \pmod{q}$$

のとき

$$x - s \longleftrightarrow Q(q, \theta - s)$$

多項式の素数を法とする因数分解についてはたとえば次を参照せよ。

D.E.Knuth(中川圭介訳)、準数値算法 / 算術演算、サイエンス社、1986 年
木田祐司、初等整数論、朝倉書店、2001 年

SNFS と違って $Q(q, s)$ の生成元を求める必要はない。この q, s をそのままファイルに保存すれば良い。

GNFS ではさらに平方剰余記号を用いる。 $Q(q, s)$ を G で用いたのと同様な 1 次の素イデアルとする。ただし q は代数的な large prime よりも大きくとる。各 $Q(q, s)$ に対し指

¹³代数的整数論の一般論はたとえば 石田 信、代数的整数論、森北出版、1974 年 を参照せよ。

標 $\chi = \chi(q, s)$ を

$$\chi(a + b\theta) = \left(\frac{a + bs}{q} \right)$$

として定義する。ただし右辺は平方剰余記号である。指標の個数は N の桁数個もあれば十分である¹⁴。

2.3.1 factor base の個数

100 桁前後の合成数 N に対して多項式 $f(x) = c_d x^d + c_{d-1} x^{d-1} + \dots + c_1 x + c_0$ の次数は $d = 5$ が適当である。そして前節で $|c_i| \sim N^{1/(d+1)}$ にとるとき factor base の個数は実験の結果、次のようにするのが最善であるらしいことが分かった。

$f(X) = c_5 X^5 + c_4 X^4 + c_3 X^3 + c_2 X^2 + c_1 X + c_0, \quad c_i \sim N^{1/6}$ による 一般数体ふるい法のパラメータ K は 1000、? は推定 CPU は PentiumII-400MHz						
桁数	70 桁	80 桁	90 桁	100 桁	110 桁	120 桁
有理的 FB の個数 #RFB	25K	56K	115K	220K	416K?	750K?
代数的 FB の個数 #AFB	100K	220K	460K	880K	1660K?	3000K?
指標の個数 #CFB	70	80	90	100	110?	120?
ふるいの幅 H_a	250K	560K	1150K	2200K	4000K?	7500K?
ふるいの幅 H_b	125K	310K	680K	1400K	2800K?	5600K?
実行時間	20 時	5 日	25 日	120 日	20 月?	8 年?
(PPMPQS の実行時間)	(2.5 時)	(17 時)	(6 日)	(40 日)	(12 月?)	(8 年?)
#AFB は #RFB の 4 倍、 H_a は #RFB の 10 倍、 H_b は #RFB の 5 ~ 7 倍である。 そして #RFB は 10 桁で 2 倍に増える。#CFB は 20-30 個程度で十分。						

80 桁未満では $d = 4$ にした方が有利である。しかし 80 桁未満では GNFS 法より、二次ふるい法を用いた方が圧倒的に速いので実用的な意味はない。最近のすべての高速化法を取り入れた GNFS 法と二次ふるい法のどちらが高速になるかの境目は 100 ~ 120 桁にあると言われている。ここでの解説および実験は“古典的”な GNFS 法であって、最新の手法は取り入れていない。より良い方法を取り入れて factor base の個数をずっと減らし、実行時間をかなり短縮することができるであろう。

所要時間を実験とシミュレーションによって求めてみると PentiumII-400MHz を使って 100 桁で約 120 日かかり、10 桁増すごとに所要時間は 5 倍に増える¹⁵。一方、以前に作成した PPMPQS 法のプログラムでは 100 桁で約 50 日かかり、10 桁増すごとに 8 倍に増える。これから、我々のプログラムでは GNFS と PPMPQS の拮抗する桁数は 120 桁前後になると推定される¹⁶。

¹⁴add on Jan.2002: 20 個から 30 個で十分。

¹⁵add on Jan.2002: 最近の良くできたプログラムでは 3-4 倍で済むらしい。

¹⁶GNFS でも PPMPQS でもまだ我々が実際に行ったことのない未知の領域なのでほとんどあてにはならない。ただ 80 桁以下だったり 200 桁以上だったりすることがないくらいは断言できる。また GNFS のプログ

2.4 ふるいの実際

(1-1) の分解は b を $1, 2, 3, \dots, H_b$ と順に動かし、各 b について a を $-H_a, -H_a + 1, \dots, H_a - 1$ と動かして、簡単な(1次の)ふるいにかけて候補をとりだす。

- (1) ふるいのワークエリア $Sieve[0], Sieve[1], \dots, Sieve[2H_a - 1]$ と補助のワークエリア $W[0], W[1], \dots, W[2H_a - 1]$ を確保する。
- (2) $Sieve[]$ を 0 クリア。 $W[]$ は b と互いに素なところに 1 を、素でないところに 0 を置く。
- (3) 各 $p \in F$ について p の倍数に対応するところに $\lfloor \log_2 p \rfloor$ を加えていく。
これは最初の位置さえ計算すれば次の番地は p を加えるだけで済む。ワークエリアの先頭は $-H_a + bM$ に対応するので最初に p の倍数になる番地 r は $r - H_a + bM \equiv 0 \pmod p$ つまり $H_a - bM = tp + r, 0 \leq r < p$ となる r として計算できる。次のスタートは $-M \pmod p$ だけずれることに注意。
- (4) すべての $p \in F$ について (3) を行なったのち $\lfloor \log_2 bM \rfloor - 2\lfloor \log_2 p_{max} \rfloor$ 未満の値の番地はこのテストを通らなかったのに対応する $W[]$ を 0 クリアしておく。

$a + b\theta$ の分解は次の事実を用いれば簡単に行うことができる。

$a + b\theta$ が 素イデアル $Q(q, s)$ で割れるための必要十分条件は
 $a + bs$ が q で割れることである。

- (5) $Sieve[]$ を 0 クリア。
- (6) 各 $\pi = \pi(q, s) \in G$ について q の倍数に対応する $Sieve[]$ に $\lfloor \log_2 q \rfloor$ を加えていく。これは最初の位置さえ計算すれば次の番地は q を加えるだけで良い。ワークエリアの先頭は $a - L$ に対応するので最初に π の倍数となる番地 r は $r - L + bs \equiv 0 \pmod q$ 、つまり $L - bs = tq + r, 0 \leq r < q$ となる r として計算できる。次のスタートは $s \pmod q$ だけずれることに注意。
- (7) すべての $\pi \in G$ について (6) を行なったのち $\lfloor \log_2 |\text{Norm}_{K/Q}(a + b\theta)| \rfloor$ に近いものを取り出す(かなりの誤差を許す)。

これで、かなり絞りこまれたはずなので、実際に確認しよう。

a, b を上の 2 つのふるいを通ったペアとする。

- (8) $a + bM$ を F の素数で素因数分解し、分解しきれない部分を X とする。

ラムは PPMPQS のプログラムに比べて熟成されていないことを考慮しなければいけない。もしこの GNFS のプログラムを 2 倍高速にできればクロスオーバーは 100 桁に下がる。それは十分ありうることである。

(9) $|\text{Norm}_{K/Q}(a + b\theta)|$ を

$$NG = \{\text{Norm}_{K/Q}Q \mid Q \in G\} = \{q \mid Q = Q(q, s) \text{ for some } Q \in G\}$$

の素数で分解する。分解しきれない部分を q とする。 q が素数でなければ捨てる。素数ならば $a + bs \equiv 0 \pmod{q}$ となる s を求めて $Y = Q(q, s)$ とする。

これで次のような分解が得られた。

$$\begin{cases} a + bM &= \prod_{p \in F} p^{e(p)} X \\ [a + b\theta] &= \prod_{Q \in G} Q^{e(Q)} Y \end{cases}$$

X, Y の大きさにより4つの場合に分けてファイルにしまい、十分多く集ったところでそれぞれの処理をする。

(FF型) $X = Y = 1$ の場合。

$a + bM$ を F の元で分解することはすでに済んでいる。単項イデアル $[a + b\theta]$ は G の元で完全に分解される。

実際の計算には

MainReduction

単項イデアル $[a + b\theta]$ が素イデアル $Q = Q(q, s)$ で割り切れるための必要十分条件は
 $a + bs$ が q で割り切れることであり
割り切れるときは
 $\text{Norm}_{K/Q}(a + b\theta)$ の q べき指数と
 $[a + b\theta]$ の Q べき指数は等しい。

を用いれば G による分解を求めることができる。

large prime 部の操作

(FF型) 以外のデータで large prime 部が他のデータのそれと一致しないものはこの先使い道がないので捨ててしまう。それらを選別するには適当なハッシュ関数を用意して、それぞれの large prime が何回現れたかを数えれば良い。ここは無駄なデータを削除するのが目的なのでハッシュ関数の衝突は無視してかまわない。

これにより有理的 large prime X または代数的 large prime Y の少なくとも一方が他のデータでは現れなかったものは捨てる。

また Y は二つのパラメータを持つため処理が面倒である。 Y 成分について並べ替えをして通し番号にしておくこの先の処理が楽である。

(PF型) $X > 1, Y = 1$ の場合。

X が一致するものを X について並べ替えをして見つける。

(FP 型) $X = 1, Y > 1$ の場合。

Y が一致するものを Y について並べ替えをして見つける。

(PP 型) $X > 1, Y > 1$ の場合。

(PF 型) と合わせて X が一致するものから新しい (FP 型) を作る。次に (FP 型) と合わせて Y が一致するものから新しい (PF 型) を作る。

増えた (PF 型), (FP 型) からそれぞれすでに述べたようにして (FF 型) を作る。この処理は2回繰り返す。この結果、新しく得られたデータは次のようなフォーマットで記録されることになる。

$$((a_1, b_1, X_1, Y_1), (a_2, b_2, X_2, Y_2), \dots, (a_r, b_r, X_r, Y_r)),$$

ただし $\prod_{i=1}^r X_i, \prod_{i=1}^r Y_i$ はそれぞれ平方数である

2.5 Free Relations

$a + b\theta$ の中でとくに $b = 0$ の数、つまり有理整数は扱っていなかった。これは次のように分解できたときには特別に良いデータを与える。

$$(2 - 2') \quad [p] = \prod_{Q|p, Q \in G} Q^{e(Q)}$$

このとき $a = p, b = 0$ は (FF 型) のペアとなる。このような関係式を *free relation* という。こういう素数 p がどのくらいあるかは密度定理で調べることができる。 $K = \mathbb{Q}(-\sqrt[3]{2})$ の場合は $\#F/6$ 個あるので、かなりの助けになるが GNFS でもっとも重要な一般の 5 次体の場合は *free relations* の個数はたったの $\#F/120$ であって、わざわざ別処理をする価値はないともいえる。ただし後述の 2 次式を二つ使う MP-GNFS の場合には半分が *free relations* で得られる。

2.6 行列の掃き出し

以上から出来上がる行列は $\#F + \#G + \#H$ 次の正方行列である。これを GAUSS 消去するのだが、PPMPQS のときと同様にまず intelligent GAUSS¹⁷ でサイズを小さくしてから通常の GAUSS を行う。GNFS で得られる行列は PPMPQS のそれよりも疎である (つまり 0 が多い) のでより縮小率は大きくなる。およそ 6 ~ 7 分の 1 の次数に減る。メモリの量で言うと $1/36 \sim 1/49$ ということになる。

2.7 平方数の構成

以上により

$$\prod (a_i + b_i M) = \left(\prod_{p \in F} p^{e(p)} \right)^2$$

¹⁷前著を参照されたい。

と \mathbb{Z} で平方数になり、かつ

$$\prod [a_i + b_i\theta] = \left(\prod_{\mathcal{P} \in G} \mathcal{P}^{e(\mathcal{P})} \right)^2$$

と O_K で平方イデアルになり、かつ

$$\chi \left(\prod (a_i + b_i\theta) \right) = 1 \quad \text{for } \forall \chi \in H$$

となる (a_i, b_i) の組み合わせが分かった。残るは

$$\prod (a_i + b_i\theta) = \beta^2$$

となる $\beta \in \mathbb{Z}[\theta]$ を求めることである。

3 代数的数の平方根

Adleman による GNFS 法の改良の副作用として次の点が問題になった。

- 代数的数の平方根をどうやって求めるか？

これは一応の解決を見た。

Jean-Marc Couveignes: *Computing a square root for the number field sieve*, in Lenstra and Lenstra(Eds.) *The development of the number field sieve*, Lecture Notes in Math. **1554**(1993), 95–102.

しかし、これでもまだ十分に高速とは言い難い。最近は高速性と任意の次数で使えることから

Peter L. Montgomery: *Square Roots of Product of Algebraic Numbers*, Proceedings of Symposia in Applied Mathematics **48**(1994), 567–571.

Phong Nguyen: *A Montgomery-Like Square Root for the Number Field Sieve*, in *ANT III*, Lecture Notes in Computer Sciences **1423**(1998), 151–168.

の方法が有力である。ここでは解説しない。

3.1 平方数の前処理

たとえば $3 + 2\sqrt{2}$ は $(1 + \sqrt{2})^2$ であるから $\mathbb{Z}[\sqrt{2}]$ で平方根を持つが $\mathbb{Z}[2\sqrt{2}]$ では平方根を持たない。したがって不幸にも $\theta = 2\sqrt{2}$ にとってしまった場合はそのままでは平方根が求まらない。

定理 θ は代数的整数でその最小多項式は $f(x)$ であるとする。このとき $\mathbb{Q}(\theta)$ の任意の代数的整数 γ に対して $f'(\theta)\gamma$ は $\mathbb{Z}[\theta]$ の元である。

したがって、安全のためには平方根を求める数 α に $f'(\theta)^2$ をかけておけば平方根は必ず $\mathbb{Z}[\theta]$ で求まる。以下このように処理したものを α と考える。

3.2 UFD の場合

整数環が UFD の場合は (少なくとも理論的には) 簡明である。数を単数と素元によって既約分解し、各々のべき指数を半分にしたべき乗の積を作れば良い。NFS では $a + b\theta$ 型の元を多数掛けたものの平方根が必要であるが、このやり方では積を作らずに済ませることが出来る。

3.3 一般の場合 : Hensel lift を用いる方法

一般の場合には、分解したい数が大きいと α が数万から数百万個の $a_i + b_i\theta$ の積となり α の係数が非常に大きくなって平方根の計算が難しくなる。

しかし多項式がモニックで次数も大きくなければ標準的な Hensel lift によってなんとか平方根を計算することができる。以下ではこの方法を紹介する。

一般の場合にも多くの積の中から平方数になる組み合わせを見つけて取り除いておけばこの場合に帰着できる。

以下 $f(x)$ はモニックな d 次既約多項式とし、根の一つを θ とする。 $K = \mathbb{Q}(\theta)$ と記す。

3.4 $f(x)$ が $\text{mod } p$ で既約となる p が存在する場合

奇素数 p を K でも素数であるもの¹⁸の中から一つとって固定する。かつ α を割らないものとする¹⁹。最初の計算が容易なように小さな素数にとる。

まず

$$\delta_0^2 \alpha \equiv 1 \pmod{p}$$

となる δ_0 を求める。つまり α の逆数の $\text{mod } p$ での平方根である。逆数にするのは次の段階の Hensel lift における逆数計算を避けるためである。

例 $f(x) = x^4 + x^3 + x^2 + x + 1$ とする²⁰。

$$\alpha = 88\theta^3 + 712\theta^2 + 1272\theta + 557$$

の平方根を求めたい。

$p = 3$ としよう²¹。

$$\alpha \equiv \theta^3 + \theta^2 + 2 \pmod{p}$$

である。 r, s, t, u を $0, 1, \dots, p-1$ で動かして

$$(r\theta^3 + s\theta^2 + t\theta + u)^2 (\theta^3 + \theta^2 + 2) \equiv 1 \pmod{p}$$

となるものを探すが、代数的数 θ を $x \pmod{f(x)}$ と同一視すれば

$$(rx^3 + sx^2 + tx + u)^2 (x^3 + x^2 + 2) \equiv 1 \pmod{(p, f(x))}$$

となるものを探ることになる。これは高々 p^d 回のループであるから工夫を凝らす必要はない。

解はすぐに見つかって $\delta = \theta^3 + \theta^2$ である。

これから始めて modulus を p^2, p^2, \dots と上げて行く。希望的な観測としては $p^{2^k} > 2\sqrt{\alpha}$ の係数の絶対値の最大値 となるまで行けば十分であってほしい²²。

¹⁸ $f(x)$ が $\text{mod } p$ で既約ということと同じ。

¹⁹ $a + b\theta$ は一次の素イデアルのみを因子に持つので、自動的に p では割れない。

²⁰1 の 5 乗根のうち 1 を除く 4 つがこの方程式の根である。

²¹この $f(x)$ は非常に性質が良く、素数 p の K における分解は $\text{mod } 5$ での値で決まる。

$p \equiv 1 \pmod{5}$ ならば $f(x)$ は $\text{mod } p$ で異なる 4 つの 1 次式に分解される。

$p \equiv 4 \pmod{5}$ ならば $f(x)$ は $\text{mod } p$ で異なる 2 つの 2 次式に分解される。

$p \equiv 2, 3 \pmod{5}$ ならば $f(x)$ は $\text{mod } p$ で既約である。

これは円分体論の典型的例である。

²²実際に何乗で良いかは事前にはわからないのである程度大きくなったら試しながら進むべき。

3.4.1 Hensel lift

命題

$$\delta_j^2 \alpha \equiv 1 \pmod{P^{2^j}}$$

のとき

$$\delta_{j+1} = \delta_j + \delta_j \frac{1 - \delta_j^2 \alpha}{2} \pmod{P^{2^{j+1}}}$$

とすれば (δ_j は $\pmod{P^{2^j}}$ で、 α は $\pmod{P^{2^{j+1}}}$ で計算する)

$$\delta_{j+1}^2 \alpha \equiv 1 \pmod{P^{2^{j+1}}}$$

である。

証明

$$\delta_{j+1} = \delta_j + p^{2^j} \gamma$$

と置く。

$$\begin{aligned} (\delta_j + p^{2^j} \gamma)^2 \alpha &\equiv \delta_j^2 \alpha + 2p^{2^j} \delta_j \gamma \alpha \pmod{P^{2^{j+1}}} \\ &\equiv 1 + (\delta_j^2 \alpha - 1) + 2p^{2^j} \delta_j \gamma \alpha \pmod{P^{2^{j+1}}} \end{aligned}$$

であるから

$$\text{右辺} \equiv 1 \pmod{P^{2^{j+1}}}$$

であるためには

$$\frac{\delta_j^2 \alpha - 1}{p^{2^j}} + 2\delta_j \gamma \alpha \equiv 0 \pmod{P^{2^{j+1}}}$$

であるから

$$\gamma \equiv \frac{1}{2\delta_j \alpha} \frac{1 - \delta_j^2 \alpha}{p^{2^j}} \pmod{P^{2^j}}$$

が必要十分である。ここで $\delta_j^2 \alpha \equiv 1 \pmod{P^{2^j}}$ であったから $(\delta_j \alpha)^{-1} \equiv \delta_j \pmod{P^{2^j}}$ である。したがって

$$\gamma \equiv \delta_j \frac{1 - \delta_j^2 \alpha}{2p^{2^j}} \pmod{P^{2^j}}$$

である。□

以上を繰り返して p^{2^k} が十分大きくなれば

$$\delta_k^2 \alpha \equiv 1 \pmod{P^{2^k}}$$

より

$$(\delta_k \alpha)^2 \equiv \alpha \pmod{P^{2^k}}$$

となって $\delta_k \alpha$ の係数を $[-p^{2^k}/2, p^{2^k}/2)$ でとったものが求める平方根 (のひとつ) になる。

例 今までの計算を続けると

$$\text{mod } 3^2 = 9 \text{ では } \delta_1 = 7\theta^3 + 7\theta^2 + 3\theta$$

$$\text{このとき } \delta_1\alpha \equiv -5\theta^3 - 2\theta^2 - 3\theta - 2 \pmod{3^2}$$

$$\text{mod } 3^4 = 81 \text{ では } \delta_2 = 25\theta^3 + 70\theta^2 + 48\theta + 72$$

$$\text{このとき } \delta_2\alpha \equiv 22\theta^3 - 2\theta^2 - 12\theta - 11 \pmod{3^4}$$

$$\text{mod } 3^8 = 6561 \text{ では } \delta_3 = 3427\theta^3 + 799\theta^2 + 3531\theta + 6147$$

$$\text{このとき } \delta_3\alpha \equiv 22\theta^3 - 2\theta^2 - 12\theta - 11 \pmod{3^8}$$

これより $22\theta^3 - 2\theta^2 - 12\theta - 11$ が解らしいことがわかる。試してみると実際にそうである。

これを UBASIC のプログラムにすると以下の通りである。UBASIC の制約から 2000 桁程度のものにしか使えない。したがって NFS で現れる平方根は扱えない。流れ図として読んでもらいたい。

```
10 'sq4
20 'finding square root of algebraic numbers
30 'LNCS 1554 p71-72
40 '
50 'make a problem
60 D=4
70 dim F(D),A(D-1),B(D-1),C(D-1),FR(D-1,D-1)
80 dim Ap(D-1),Del(D-1),W(2*D-2),W2(D-1)
90 F(0)=1:F(1)=1:F(2)=1:F(3)=1:F(4)=1
100 gosub *SethighpowerX
110 B(0)=-11:B(1)=-12:B(2)=-2:B(3)=22
120 gosub *Mul(&B(),&B()):for I=0 to D-1:A(I)=W(I):next
130 gosub *Print(&A()):'problem: solve ( )^2 = A()
140 'solve a problem
150 Maxc=abs(A(0)):for I=1 to D-1:Maxc=max(Maxc,abs(A(I))):next
160 Maxc=round(sqrt(Maxc))
170 P=3
180 gosub *Hensel_init
190 while P<=2*Maxc
200   P=P^2
210   gosub *Hensel
220 wend
230 gosub *MulP(&Ap(),&Del())
240 gosub *PrintP(&W()):' answer is W()
250 end
260 '
270 *Hensel
```

```

280 for I=0 to D-1:Ap(I)=A(I)@P:next
290 gosub *MulP(&Del(),&Del()):for I=0 to D-1:W2(I)=W(I):next
300 gosub *MulP(&W2(),&Ap())
310 W2(0)=(3-W(0))@P:for I=1 to D-1:W2(I)=(-W(I))@P:next
320 gosub *MulP(&Del(),&W2())
330 C=modinv(2,P)
340 for I=0 to D-1:Del(I)=W(I)*C@P:next
350 return
360 '
370 *Hensel_init
380 for I=0 to D-1:Ap(I)=A(I)@P:next
390 for T=0 to P^D-1:W=T
400   for I=0 to D-1
410     W\=P:Del(I)=res
420   next
430   gosub *MulP(&Del(),&Del()):for I=0 to D-1:W2(I)=W(I):next
440   gosub *MulP(&W2(),&Ap())
450   for I=1 to D-1
460     if W(I)<>0 then cancel for:goto 490
470   next
480   if W(0)=1 then 510
490 next
500 print "no solution":end
510 return
520 '
530 *SethighpowerX
540 local I,J
550 for J=0 to D-1:FR(0,J)=-F(J):next
560 for I=1 to D-1
570   FR(I,0)=0:for J=1 to D-1:FR(I,J)=FR(I-1,J-1):next
580   C=FR(I-1,D-1)
590   for J=0 to D-1:FR(I,J)+=C*FR(0,J):next
600 next
610 return
620 '
630 *Print(&X())
640 local I
650 print "(";
660 for I=0 to D-1:print X(I);:next
670 print ")"

```

```

680 return
690 '
700 *PrintP(&X())
710 local I
720 print "(";
730 for I=0 to D-1
740     if X(I)<P\2 then print X(I); else print X(I)-P;
750 next
760 print ")"
770 return
780 '
790 *Mul(&X(),&Y())
800 local I,J
810 for I=0 to 2*D-2:W(I)=0:next
820 for I=0 to D-1
830     C=X(I)
840     for J=0 to D-1
850         W(I+J)+=C*Y(J)
860     next J
870 next I
880 for I=0 to D-2:C=W(I+D)
890     for J=0 to D-1:W(J)+=C*FR(I,J):next
900 next
910 return
920 '
930 *MulP(&X(),&Y())
940 local I,J
950 for I=0 to 2*D-2:W(I)=0:next
960 for I=0 to D-1
970     C=X(I)
980     for J=0 to D-1
990         W(I+J)+=C*Y(J)@P
1000     next J
1010 next I
1020 for I=0 to D-2:C=W(D+I)
1030     for J=0 to D-1:W(J)+=C*FR(I,J)@P:next
1040 next
1050 for I=0 to D-1:W(I)=W(I)@P:next
1060 return

```

3.5 $f(x)$ が $\text{mod } p$ で既約となる p が存在しない場合

いかなる奇素数 p も K では素数でない場合、言いかえるとどんな p に対しても $f(x)$ が $\text{mod } p$ で可約である場合は前節の方法はそのままでは通用しない。

一般論は話が面倒になるので nfs で必要な場合だけに限定して考える。多項式の次数が 7 次以下でこのような不都合が起こるのは

- (1) 4 次でそのガロア群が 2×2 型の場合
- (2) 6 次でそのガロア群が S_3 の場合

のみである。(1) について解説する。

問題となるのは α に対して

$$\delta_0^2 \alpha \equiv 1 \pmod{pO_K}$$

を満たす δ_0 を探すところである。このとき pO_K は K の素イデアルではないので解 δ_0 が符号の違うもの以外にも存在するのである²³。もちろん α の本当の平方根は 2 つしかないから、初期値 δ_0 のうち 2 つを除いては誤った解に導くことになる。

4 つの初期値は符号が異なるのみの 2 つを組にして 2 組に分かれる。異なる組から一つずつ取り出して始めればどちらか一方は正しい解を導く。

例 $f(x) = x^4 + 1$, $\alpha = 668 + 376\theta - 292\theta^2 - 480\theta^3$, $p = 3$ とすると δ_0 として

$$\begin{aligned} \delta_{0,1} &= 2\theta + \theta^2 \\ \delta_{0,2} &= \theta + 2\theta^2 \\ \delta_{0,3} &= 1 + 2\theta + 2\theta^2 + \theta^3 \\ \delta_{0,4} &= 2 + \theta + \theta^2 + 2\theta^3 \end{aligned}$$

が得られ、後半の 2 つから正しい答 $\beta = 12 + 12\theta + 2\theta^2 - 22\theta^3$ が導かれる。

$pO_K = P_1 P_2$ と素イデアル分解しているとする。

中間結果 $\beta_j = \delta_j \alpha$ に対して

$$\beta_j \equiv \begin{cases} \pm \beta \pmod{P_1^{2j}} \\ \pm \beta \pmod{P_2^{2j}} \end{cases}$$

となり、符号が揃ったものが正しい解を、符号が異なるものが誤った解を与える。

そこで

$$\beta'_j \equiv \begin{cases} \beta_j \pmod{P_1^{2j}} \\ -\beta_j \pmod{P_2^{2j}} \end{cases}$$

として別の β'_j を作れば、 β_j, β'_j の一方は必ず解になる。

²³これはちょうど

$$x^2 \equiv 4 \pmod{13}$$

の解は $\pm 2 \pmod{13}$ しか存在しないが

$$x^2 \equiv 4 \pmod{15}$$

の解は $\pm 2, \pm 7 \pmod{15}$ の 4 つがあることと同様である。

3.6 一般の場合：Couveignes の方法

f を GNFS の基本となる既約多項式で、その一つの根 θ を添加した体を K とする。 f の次数を d 、したがって K の次数も d とおく。

3.7 θ が代数的整数の場合

まず、 θ が代数的整数、つまり f がモニックな整数係数多項式である場合を扱う。モニックであることの利点は次の自然な同型が存在することである。

$$\mathbb{Z}[X]/(f(x)) \sim \mathbb{Z}[\theta]$$

$Norm_{K/\mathbb{Q}}$ を単に \mathcal{N} と記す。

以下の方法で平方根を計算するためには

d は奇数である

と仮定する必要がある²⁴。

問題

$$\alpha = \prod_{i=1}^r (a_i + b_i \theta)$$

は $\mathbb{Z}[\theta]$ の平方数であることはわかっているとき

$$\beta^2 = \alpha$$

となる $\beta \in \mathbb{Z}[\theta]$ を計算せよ。

注意 実際到我々に必要なのは分解したい合成数 N を法とした答えであるがこの節では平方根をそのまま求める。

もっとも直接的な方法は前節で述べた Hensel lift(=Newton 法) を用いるものである。そこでの問題点は係数が巨大になることであった。この節ではそれを克服する準備として中国の剰余定理を用いた方法を述べる。

中国の剰余定理の基盤となる素数を次のようにとる。 $p_i, i = 1, 2, \dots, s$ を奇素数で f が $\text{mod } p_i$ で既約なものとする²⁵。これは p_i が $K = \mathbb{Q}[\theta]$ で分岐も分解もしないことを意味する。そこで K における p_i 上の唯一の素イデアルを P_i と記す²⁶。

²⁴2つの平方根をノルムの符号で区別するため。

²⁵実際に用いる $f(X)$ は高々5次式であるから $\text{mod } p$ で既約であるためには1次、2次の因子がなければ良い。それは $\text{gcd}(X^{p^2} - X, f(X)) = \text{定数}$ であることで判定できる。

また一般にはこのような p_i が存在しない場合がある。例えば二つの2次体の合成であるような4次体を生成する元の最小多項式、 $x^4 + 1$ や $x^4 - 10x^2 + 1$ などである。しかし我々が実的に用いる3次式、5次式、7次式のように次数が素数の場合はこのような p_i は必ず存在する(存在すれば無限個存在する)。また次数が合成数であってもこのような p_i が存在しないような f は少数派であってアルゴリズムの実行可能性についてはまったく問題がない。

²⁶唯一なのだから p_i と P_i は区別しなくても支障はないが、どこで考えているかをはっきりさせるために別の記号を用いる。

3.7.1 $\beta_i^2 \equiv \alpha \pmod{P_i}$ となる β_i を求める。

p が小さい場合は前節のように総当りですべて探することも可能であるが奇数次の場合はもっとスマートに答えを見つけることができる²⁷。

添字 i は省略する。

命題 $b^2 \equiv \mathcal{N}\alpha \pmod{p}$ となる b をとり²⁸

$$\beta = \alpha^{\frac{1+p+p^2+\dots+p^{d-1}+1}{2}} / b \pmod{P}$$

とすると

$$\beta^2 \equiv \alpha \pmod{P}$$

かつ

$$\mathcal{N}\beta \equiv b \pmod{p}$$

である。

証明 p は K/\mathbb{Q} で分岐も分解もしていないので $O_K \pmod{P}$ は d 次の有限体 \mathbb{F}_{p^d} (に同型) である。

$\text{Norm}_{\mathbb{F}_{p^d}/\mathbb{F}_p}$ も \mathcal{N} と記す。

$\alpha \in \mathbb{F}_{p^d}^\times$ が平方数。

$$\Leftrightarrow \alpha^{\frac{p^d-1}{2}} = 1$$

$$\Leftrightarrow \left(\alpha^{\frac{p^d-1}{p-1}} \right)^{\frac{p-1}{2}} = 1$$

$$\Leftrightarrow (\mathcal{N}\alpha)^{\frac{p-1}{2}} = 1$$

$$\Leftrightarrow \mathcal{N}\alpha \text{ が } \mathbb{F}_p^\times \text{ で平方数。}$$

そして $b \in \mathbb{F}_p$ が $b^2 = \mathcal{N}\alpha$ をみたすならば

$$\begin{aligned} & \left(\alpha^{\frac{1+p+p^2+\dots+p^{d-1}+1}{2}} \right)^2 \\ &= \left(\alpha^{\frac{1}{2} \frac{p^d-1}{p-1} + \frac{1}{2}} \right)^2 \\ &= \alpha^{\frac{p^d-1}{p-1} + 1} \\ &= \alpha \mathcal{N}\alpha \\ &= \alpha b^2 \end{aligned}$$

b を移項して最初の式を得る²⁹。

²⁷前節ではただ一つの素数を用いたが、ここでは数万個の素数を用いるのでかなり大きめの素数も用いなければいけない。

²⁸ α は $a + b\theta$ の積だから P_i では割れない。したがって $b \neq 0$ である。

²⁹ d が奇数なので $(1+p+p^2+\dots+p^{d-1}+1)/2$ は整数。

次に

$$\begin{aligned}
 \mathcal{N}\beta &\equiv (\mathcal{N}\alpha)^{\frac{1+p+p^2+\dots+p^{d-1}+1}{2}} / \mathcal{N}b \pmod{P} \\
 &\equiv b^{1+p+p^2+\dots+p^{d-1}+1} / b^d \pmod{p} \\
 &\equiv b^{(p-1)+(p^2-1)+\dots+(p^{d-1}-1)+1} \pmod{p} \\
 &\equiv b \pmod{p}
 \end{aligned}$$

3.7.2 $\beta_i^2 \equiv \alpha \pmod{P_i^{2^k}}$ となる β_i を求める。

中国の剰余定理では互いに素な数を法として用いるが、素数そのものでなく素数のべきを用いた方が良いことがある。この問題のように Hensel lift が有効に働く場合である。多数の素数について平方根を計算するより一つの素数について Hensel lift を行う方が速いのである。しかしべきを上げすぎると係数が巨大になって一つ一つの計算が遅くなる、あるいは計算そのものが実行できなくなってしまうのであった。このトレードオフはシステムに依存する。

添字 i は省略する。

Hensel lift により1段ずつ上げて行く。

命題

$$\delta = \alpha^{\frac{(1+p+p^2+\dots+p^{d-1})(p-2)-1}{2}} \cdot b \pmod{P}$$

とおくと

$$(\delta\alpha)^2 \equiv \alpha \pmod{P}$$

かつ

$$\mathcal{N}(\delta\alpha) \equiv b \pmod{p}$$

である。

証明 $\alpha^{\frac{p^d-1}{2(p-1)}} \equiv 1$ であることを用いて前証明にならえば良い。□

これより δ に対して

$$new\delta = \delta + \delta \frac{1 - \delta^2\alpha}{2} \pmod{P^{2^{j+1}}}$$

は

$$(new\delta\alpha)^2 \equiv \alpha \pmod{P^{2^{j+1}}}$$

を満たしている。必要なだけ先の j まで計算すればそのときの $\beta = \delta\alpha$ が答えである。

3.7.3 $\beta^2 = \alpha$ となる β を求める。

$\beta_i^2 \equiv \alpha \pmod{P_i^{2^k}}$ となる β_i が見つかったところで本来の平方根を求めたい。 $f(x)$ の次数 d は奇数としていたので θ として実数がとれる。つまりすべての計算は実数体の中で済んでしまう。とくに正負の性質を利用できる。

β として $N\beta$ が正となる方の解をとることにする。次数が奇数であるから必ず両方の符号のものがある。 $b = N\beta$ は $N\alpha$ の平方根として簡単に求まる。この b を用いてこれまでの計算を行えば

$$\begin{cases} x^2 \equiv \alpha \pmod{P_i^{2^k}} & \text{for } \forall i \\ Nx \equiv N\beta \pmod{p_i^{2^k}} & \text{for } \forall i \end{cases}$$

となる解が得られる。法が十分たくさんとられていればこれから β そのものが求まる。

いつもの通り $M = \prod p_i^{2^k}$ は β の係数の最大の 2 倍以上にとる。しかしあらかじめこの値がわからないのでおおまかな予測をする。問題の積が $a_i + b_i\theta$ の何個の積になっているかということ factor base の個数の 6 割から 8 割というのが大まかな目安になるだろう。そこで a_i, b_i の最大値の (factor base の個数の 8 割) 乗くらいで α の係数が押さえられると考えると β の係数は楽観的に見てその平方根くらいのもの。そこで

$$\text{法の個数} = 0.4 \frac{\#FB \cdot \log(\max a_i)}{2^k \log p}$$

くらいが適当であると見積もられる³⁰。

これを増やすのはふるいの条件には影響しないからデータを集めた後で増やして再計算することが可能である³¹。

3.7.4 例

$$f(x) = x^3 + 2, \quad \theta = \sqrt[3]{-2}$$

の場合。

$$N(a\theta^2 + b\theta + c) = 4a^3 - 2b^3 + c^3 + 6abc$$

である。

このとき

$$\alpha = (1 + \theta)(-2 + \theta)(2 + \theta)(3 + 2\theta)(-5 + 4\theta)(-7 + \theta)$$

は平方数である(らしい)ので平方根 β を計算したい。

α の生成する単項イデアルは平方イデアルであり

$$N\alpha = (2 \cdot 3 \cdot 5 \cdot 11 \cdot 23)^2$$

となることはわかる。したがって

$$b = N\beta = 2 \cdot 3 \cdot 5 \cdot 11 \cdot 23$$

³⁰general NFS の場合は各 a_i に多項式の最高次係数が掛かるので非常に多くなる。

³¹NFS においては失敗する理由には積が実際には平方数でない場合もある。これは平方剰余記号の個数が足りない場合に起こりうる。この場合は中国の剰余定理の法の個数をいくら増やしても無駄である。factor base の平方剰余記号の部分を増やして smooth data に対応するベクトルの作り直しをしなければならない。しかし 100 桁の数に対しても 100 個もあれば十分過ぎるくらいなので気にすることはない (added on Jan.2002: 10 個から 30 個で十分)。

である。これも一般には巨大数になるのでそのものを計算するのではなく各素数（今の場合は 7, 13）ごとに mod で積を計算する。

$p = 7$ で $\alpha \equiv 4\theta^2 + 3\theta + 2 \pmod{7}$ であり $b \equiv 2 \pmod{7}$ であるから

$$\begin{aligned}\beta &\equiv \alpha^{\frac{1+7+7^2+1}{2}}/2 \pmod{7} \\ &\equiv 3\theta^2 + 3\end{aligned}$$

同様にして $p = 13$ に対して、 $2\theta^2 + 5\theta + 9$ を得る。よって $\beta = a\theta^2 + b\theta + c$ とおいて

$$\begin{aligned}a &\equiv \begin{cases} 3 & \pmod{7} \\ 2 & \pmod{13} \end{cases} \\ b &\equiv \begin{cases} 0 & \pmod{7} \\ 5 & \pmod{13} \end{cases} \\ c &\equiv \begin{cases} 3 & \pmod{7} \\ 9 & \pmod{13} \end{cases}\end{aligned}$$

を解けば

$$\begin{cases} a \equiv 80 \pmod{91} \\ b \equiv 70 \pmod{91} \\ c \equiv 87 \pmod{91} \end{cases}$$

となる。実際

$$(-11\theta^2 - 21\theta - 4)^2 = \alpha$$

である。

3.7.5 中国の剰余定理の変形

素因数分解で必要になる値は、ターゲットの数 N を法とした値であることが多い。そのため中国の剰余定理も実際の値そのものではなく \pmod{N} の値を求めるものに変形しておくというメリットが出てくる。

問題 m_1, m_2, \dots, m_s は二つずつ互いに素な自然数とし、 $M = \prod_{i=1}^s m_i$ とする。

$-0.49M < x < 0.49M$ にある x が³²

$$x \equiv \begin{cases} x_1 \pmod{m_1} \\ x_2 \pmod{m_2} \\ \dots \\ x_s \pmod{m_s} \end{cases}$$

を満たしているときに $0 < N < M$ なる N に対して $x \pmod{N}$ を計算せよ。

通常の解法にならえば

³²0.49 は誤差を見込んでいる。本来は 0.5 である。

$$M_i = M/m_i, a_i = 1/M_i \bmod m_i$$

とにおいて³³

$$z = \sum_{i=1}^s a_i x_i M_i$$

とすれば

$$x \equiv z \bmod M$$

となるのであった。これから $-0.49M < x < 0.49M$ となる x を定めて $\bmod N$ で落とせば良い。問題は M のオーダーでなく N のオーダーの計算で済ませたいということ。

$x = z - rM, r \in \mathbb{Z}$ とすると

$$-0.49 < \frac{z}{M} - r < 0.49$$

より $r = \text{round}(\frac{z}{M})$ であるから

$$r = \text{round}\left(\sum_{i=1}^s \frac{a_i x_i}{m_i}\right)$$

となり, m_i 程度の大きさの実数の s 個の和であるから倍精度程度の計算で十分である。これで $x = z - rM$ が確定する。この式の各項を $\bmod N$ で計算すれば良い。

3.7.6 例

前の例を用いる。

$$a \equiv \begin{cases} 3 & \bmod 7 \\ 2 & \bmod 13 \end{cases}$$

については $r = \text{round}(6 \cdot 3/7 + 2 \cdot 2/13) = \text{round}(2.88\cdots) = 3$ であるから

$$a = z - 3M = 262 - 3 \cdot 91 = -11$$

で同じ結果を得る。この最後の計算を $\bmod N$ で行えば多倍長計算を避けることができる。 b, c についても同様である。

³³ a_i のための M_i は $\bmod m_i$ で計算すれば良いが、次の z のための M_i は $\bmod N$ の値であることに注意。つまり二通りに計算しなければいけない。

3.7.7 高速化

$$\alpha = \prod_{i=1}^r (a_i + b_i\theta)$$

の右辺の積を二つに分ける

$$\begin{aligned}\alpha_{odd} &= \prod_{i=1, i \text{ odd}}^r (a_i + b_i\theta) \\ \alpha_{even} &= \prod_{i=1, i \text{ even}}^r (a_i + b_i\theta)\end{aligned}$$

そして

$$\bar{\alpha} = c^2 \frac{\alpha_{odd}}{\alpha_{even}}$$

を考える。ここで c は両辺を代数的整数にするために掛ける数である。係数を小さくするという点では代数的数をとるべきであるがなかなか難しい。有理整数にとれば簡単ではある。そこで

$$\bar{\gamma}^2 = \bar{\alpha}$$

となる $\bar{\gamma}$ を計算して

$$\gamma = \bar{\gamma}\alpha_{even}/c$$

とおけば

$$\gamma^2 = \alpha$$

である。

簡単のため奇数番目と偶数番目に分けたが、目的は分子・分母の相殺にあるのだから large primes が一致するペアは分子・分母に分かれるようにしたい。それにはあらかじめ $a_i + b_i\theta$ を large primes が消し合うペアが続き番号になるようにしておけば良い。

NFS の場合は α_{even} を計算する必要はない。有理数側も

$$\frac{\text{奇数番の項の積}}{\text{偶数番の項の積}}$$

を考えれば良い。

3.7.8 例

前と同じ例を用いる。

$$f(x) = x^3 + 2, \quad \theta = \sqrt[3]{-2}$$

の場合。

このとき

$$\alpha = (1 + \theta)(-2 + \theta)(2 + \theta)(3 + 2\theta)(-5 + 4\theta)(-7 + \theta)$$

は平方数である(らしい)ので平方根 β を計算したい。

$$\alpha_{odd} = (1 + \theta)(2 + \theta)(-5 + 4\theta)$$

$$\alpha_{even} = (-2 + \theta)(3 + 2\theta)(-7 + \theta)$$

である。

$$\mathcal{N}\alpha_{odd} = -2 \cdot 3 \cdot 11 \cdot 23$$

$$\mathcal{N}\alpha_{even} = -2 \cdot 3 \cdot 5^2 \cdot 11 \cdot 23$$

となることはわかる。したがって

$$\bar{\alpha} = 5^2 \frac{\alpha_{odd}}{\alpha_{even}}$$

は平方数であり、代数的整数である³⁴。

$$\mathcal{N}\bar{\alpha} = 5^4$$

であるので

$$b = 5^2$$

である。

$$\bar{\alpha} \equiv 2\theta^2 + \theta \pmod{7}$$

であるから

$$\begin{aligned} \bar{\beta} &\equiv \bar{\alpha}^{\frac{1+7+7^2+1}{2}} / b \pmod{7} \\ &\equiv -\theta^2 - 2\theta + 1 \end{aligned}$$

である。すると

$$\beta = (-\theta^2 - 2\theta + 1)\alpha_{even}/5 = -11\theta^2 - 21\theta - 4$$

となる。実際

$$(-11\theta^2 - 21\theta - 4)^2 = \alpha$$

である。今回は mod 7 だけで解が求まった。□

3.8 θ が代数的整数でない場合

つまり $f(x)$ がモニックでない:

$$f(x) = \ell x^d + c_{d-1}x^{d-1} + \cdots + c_1x + c_0, \quad \ell \neq 1$$

場合である。最高次係数をとくに ℓ で表しておく。

³⁴実際 $\bar{\alpha} = 2\theta^2 - 6\theta - 7$

問題 θ は上記 $f(x)$ の根とする。

$$\alpha = \prod_{i=1}^r (a_i + b_i \theta), \quad a_i, b_i \in \mathbf{Z}$$

は $\mathbf{Q}[\theta]$ の平方数であることはわかっているとき

$$\beta^2 = \alpha$$

となる $\beta \in \mathbf{Q}[\theta]$ を計算せよ。

α は $\mathbf{Z}[\theta]$ の元の積であるが、 $f(\theta) = 0$ によって簡約すると整数係数とは限らないので前節の方法をそのまま用いることができないのである。 θ あるいは $f(x)$ を取り替えなければならぬ。

例 $f(x) = 2x^2 - x + 1$, $\theta = (1 + \sqrt{-7})/4$ とすると $\theta^2 = (\theta - 1)/2$ である。

補題 $\ell\theta$ は代数的整数である。

証明

$$\ell^{d-1} f(x) = (\ell x)^d + c_{d-1} (\ell x)^{d-1} + \cdots + c_1 \ell^{d-2} (\ell x) + c_0 \ell^{d-1}$$

なので

$$g(y) = y^d + c_{d-1} y^{d-1} + \cdots + c_1 \ell^{d-2} y + c_0 \ell^{d-1}$$

は $\ell\theta$ を根とするモニックな整数係数多項式である。□

以下、 r は偶数であるとする³⁵。

$$\alpha_1(\omega) = \ell^r \alpha(\theta) = \prod_{i=1}^r (\ell a_i + b_i \omega)$$

として $\beta_1(\omega)^2 = \alpha_1(\omega)$ となる $\beta_1(\omega)$ を前節と同様にして計算すれば

$$\prod_{i=1}^r (a_i + b_i M) = X^2$$

であるとき

$$\beta_1(\ell M)^2 \equiv (\ell^{r/2} X)^2 \pmod{n}$$

である。

$\mathcal{N}_1(a + b\theta) = \ell \mathcal{N}(a + b\theta)$ で \mathcal{N}_1 を定義すると

$$\ell \mathcal{N}(a + b\theta) = (-b)^d f(-a/b) = \ell a^d + c_{d-1} a^{d-1} (-b) + \cdots + c_r a^r (-b)^{d-r} + \cdots + c_0 (-b)^d$$

であるから

³⁵奇数のものを用いない。あるいは factor base に、任意の $a + b\theta$ に対して 1 をとるものを追加しておく。

$$\begin{aligned}
\mathcal{N}\alpha_1 &= \mathcal{N}\ell^r \prod_{i=1}^r (a_i + b_i\theta) \\
&= \ell^{r(d-r)} \prod_{i=1}^r \ell\mathcal{N}(a_i + b_i\theta) \\
&= \ell^{r(d-1)} \prod_{i=1}^r \mathcal{N}_1(a_i + b_i\theta)
\end{aligned}$$

となるので

$$\mathcal{N}\beta_1 = \ell^{r(d-1)/2} \sqrt{\prod_{i=1}^r \mathcal{N}_1(a_i + b_i\theta)}$$

である。

これで

問題 θ は上記 $f(x)$ の根とする。

$$\alpha_1 = \prod_{i=1}^r (\ell a_i + b_i\omega), \quad a_i, b_i \in \mathbf{Z}$$

は $\mathbf{Z}[\omega]$ の平方数であることはわかっているとき

$$\beta_1^2 = \alpha_1$$

となる $\beta_1 \in \mathbf{Z}[\omega]$ を計算せよ。

という問題になった。これは解決済み。

ただし、ここで a_i が ℓa_i になった。これは $\alpha_1(\omega)$ の係数が非常に大きくなることを意味する。これは $\omega = \ell\theta$ と置いたがための結果である。なんとか小さくしないと、中国の剰余定理に用いる素数の個数が膨大になってしまう。ここでも α_1 の積を半分に分けて分子、分母を作ることによりほとんど相殺することができる。

4 Multiple Polynomial Number Field Sieve について

General Number Field Sieve(GNFS)法の新しい高速化の方法として Multiple Polynomial GNFS 法が考案された³⁶。

MP-GNFS は P.L.Montgomery により考案され (未発表)

M. Huizing: *A Multiple Polynomial General Number Field Sieve*, Lecture Notes in Computer Science **1122**(1996), 99–114.

M. Huizing: *An implementation of the number field sieve*, Experimental Mathematics **5**(1996), 231–253.

で紹介されている。以下この論文を参考にして解説する。

ただし今のところ有効に働いているのは二つの2次の多項式の場合であり 100桁前後の合成数については前節までの“古典的”な GNFS よりもこちらが主に使われているようである³⁷。

MP-GNFS の場合は、ターゲットの合成数 N に対し、複数の多項式 $f_t(x)$ ($t = 1, 2, \dots$) と共通の整数 M を選び $f_t(M)$ が N の倍数であるようにする。 $f_t(x)$ の根の1つを θ_t とし、 $K_t = \mathbb{Q}(\theta_t)$ とする。

以下では添字が煩雑になるのを避けるために個数が 2 の場合に限定する。また3個以上にしても効果はないようである。

\mathbb{Z} の素数の集合 F は共通にとり O_{K_t} の素イデアルの集合 G_t を適当にとる。さらに何個かの平方剰余記号の集合 H_t を付け加える。

a, b を互いに素な有理整数で $a + b\theta_t$ は G_t -smooth とする。つまり

$$(1-2) \quad [a + b\theta_t] = \prod_{\mathcal{P} \in G_t} \mathcal{P}^{e(\mathcal{P})}$$

と完全に素イデアル分解されるものとする。このとき $\chi \in H_t$ に対して次のようにして $e(\chi)$ を定める。

$$\chi(a + b\theta_t) = (-1)^{e(\chi)}$$

このようなペア (a, b) を $\#G_1 + \#H_1 + \#G_2 + \#H_2$ 個より多く集める。すると各べき指数を mod 2 で考えたベクトル

$$((e(\mathcal{P}) \bmod 2)_{\mathcal{P} \in G_1}, (e(\chi))_{\chi \in H_1}, (e(\mathcal{P}) \bmod 2)_{\mathcal{P} \in G_2}, (e(\chi))_{\chi \in H_2})$$

は一次従属となり、SP-GNFS と同様に、適当に組み合わせて

$$\prod_i (a_i + b_i \theta_t) \text{ は高い確率で } O_{K_t} \text{ の平方数 } (t = 1, 2)$$

³⁶これは Quadratic Sieve の Multiple Polynomial 版が本当に多数の多項式を用い、それらの良いところばかりを利用するのは趣を異にし、2-4 個の多項式しか用いない。

³⁷added on Jan.2002: 期待に反して MP-GNFS 法は使われていない。3次以上の二つの多項式をうまく作ることができないからである。

となるものを得る。

もし実際

$$\prod (a_i + b_i \theta_i) = (c_t + d_t \theta_t)^2 \quad (t = 1, 2)$$

であったとするとこれを準同型

$$\begin{aligned} \phi_t : \mathbf{Z}[\theta_t] &\longrightarrow \mathbf{Z}/N\mathbf{Z} \\ g(\theta_t) &\mapsto g(M) \bmod N \end{aligned}$$

で写せば

$$(\phi_1(c_1 + d_1 \theta_1))^2 \equiv (\phi_2(c_2 + d_2 \theta_2))^2 \pmod{N}$$

となり、ふるい法の目標である $x^2 \equiv y^2 \pmod{N}$ 型の合同式が得られる。

まとめると、MP-GNFS の原理は、

$$\begin{aligned} \text{環準同型 } \phi_t : \mathbf{Z}[\theta_t] &\longrightarrow \mathbf{Z}/N\mathbf{Z} \\ g(\theta_t) &\mapsto g(M) \bmod N \end{aligned}$$

による次の図式を2通りに進んで、その分解の違いを利用することと言える。この図式は $\mathbf{Z}/N\mathbf{Z}$ で考えれば可換だが、 \mathbf{Z} では可換とは限らない。ただし UFD であることを仮定していないので数としての分解は常に可能なわけではない。最終的に適当に組み合わせて両方で分解できた場合はなしである。

$$\begin{array}{ccc} (a, b) & \longrightarrow & a + b\theta_1 \text{ の } O_{K_1} \text{ での分解} \\ \downarrow & & \downarrow \phi_1 \\ a + b\theta_2 \text{ の } O_{K_2} \text{ での分解} & \xrightarrow{\phi_2} & \mathbf{Z} \text{ での分解} \end{array}$$

実例を掲げる。

$$\begin{cases} N & = & 1363 \\ M & = & 765 \\ f_1(x) & = & x^2 - 6x + 3 \\ f_2(x) & = & 6x^2 + 5x + 2 \end{cases}$$

とすると判別式は $D_1 = 24 = 8 * 3$, $D_2 = -23$ であり、類数はそれぞれ 1, 3 である。

ノルムは次の通り

$$\begin{aligned} \mathcal{N}_1(a + b\theta_1) &= a^2 + 6ab + 3b^2 \\ \mathcal{N}_2(a + b\theta_2) &= (6a^2 - 5ab + 2b^2)/6 \end{aligned}$$

このとき $a = 1, b = 1$ とすると

$$\begin{array}{ccc}
(1, 1) & \longrightarrow & 1 + \theta_1 = (1 - \theta_1)(4 - \theta_1) \\
\downarrow & & \downarrow \phi_1 \\
1 + \theta_2 \text{ は数としては既約} & \xrightarrow{\phi_2} & 1 + M = 766 \equiv (1 - M)(4 - M) = 581404 \pmod{N}
\end{array}$$

という違いがでる。

4.1 多項式の選択

GNFS では用いる多項式の選定が非常に難しい。しかし二つの二次式を用いる場合には例外的に小さな係数を持つ多項式を選ぶことができる。これが一番のポイントである。

多項式

$$f(x) = c_d x^d + \dots + c_1 x + c_0$$

に求められる性質は次の通りである

- (1) $a + b\theta$ のノルムの絶対値は小さい (a, b は絶対値の小さな整数で θ は $f(x)$ の根)。
- (2) ある整数 M に対して $f(M) \equiv 0 \pmod{N}$

条件 (1) は次に置き換えてほとんど十分

- (1') c_i の絶対値は小さい。

M を一つ固定して考えれば (2) を満たす (c_0, c_1, \dots, c_d) は \mathbb{Z}^{d+1} の階数 $d+1$ の sublattice L_0 の元である。

$$L_0 = \{(x_0, x_1, \dots, x_d) \mid x_0 + x_1 M + \dots + x_d M^d \equiv 0 \pmod{N}\}$$

L_0 は次のベクトルで生成される sublattice L を含む³⁸。

$$\left\{ \begin{array}{l}
\mathbf{v}_0 = (M, -1, 0, \dots, 0) \\
\mathbf{v}_1 = (0, M, -1, \dots, 0) \\
\vdots = \quad \quad \quad \vdots \\
\mathbf{v}_{d-1} = (0, \dots, M, -1) \\
\mathbf{u}_0 = (N, 0, 0, \dots, 0) \\
\mathbf{u}_1 = (0, N, 0, \dots, 0) \\
\vdots = \quad \quad \quad \vdots \\
\mathbf{u}_d = (0, 0, 0, \dots, N)
\end{array} \right.$$

これから L の最小基底を求めたい。実用的には LLL アルゴリズムによって L の LLL-既約基底を求めることで十分であろう。

³⁸いつ $L = L_0$ になるのか？

L の元は多項式の言葉に翻訳すれば

$$(a_0 + a_1x + \cdots + a_dx^d)N + (b_0 + b_1x + \cdots + b_{d-1}x^{d-1})(x - M), \quad a_i, b_i \in \mathbf{Z}$$

という形の多項式に他ならない。今の問題はこの中で絶対値(係数の平方和の平方根)ができるだけ小さな多項式を求めることである。

例 $d = 2$ のとき $N = 1363$ に対して $M = 12$ とすれば $\mathbf{c} = (1, 12, 144)$ から計算して最小ベクトルとして $(12, -1, 0)$, $(0, 12, -1)$, $(-5, 6, 9)$ を得る。
したがって用いることできる多項式は

$$9x^2 + 6x - 5 + Nr(x) + (x - 12)s(x), \quad r(x), s(x) \in \mathbf{Z}[x]$$

である。

したがって問題はむしろ M をどうとるかである。それは

$$(p, pM \bmod N, pM^2 \bmod N, \dots, pM^d \bmod N)$$

の大きさを最小にするような $p = p(M)$ を求め、さらにそれを最小にする M を見つけることである。

lattice の言葉で言い換えると

$$\begin{cases} \mathbf{w} &= (1, M, M^2, \dots, M^d) \\ \mathbf{u}_0 &= (N, 0, 0, \dots, 0) \\ \mathbf{u}_1 &= (0, N, 0, \dots, 0) \\ \dots &= \dots \\ \mathbf{u}_d &= (0, 0, 0, \dots, N) \end{cases}$$

で生成される lattice の最小ベクトルを求め、さらにそれが最小になるような M を見つける問題である。

4.1.1 2次式の場合

2次式の利点に free relations がたくさんあるということがある。さらに large primes を両方の体で 2個ずつ用いる (4 large primes) 場合は large primes の範囲まで free relations が有効である (もちろん FF 型ではなく PPPP 型ではあるが)。

Montgomery の提案は次の通り³⁹。

- 素数 p を N が $\bmod p$ の平方剰余になるようにとる。
- $c_1^2 \equiv N \bmod p$ となる c_1 を $\left(0, \frac{p}{2}\right]$ にとる。

³⁹普通の方法で2次式をとると係数の大きさは $n^{1/3}$ 程度であり、多項式は2つあるから大雑把にいうと $n^{2/3}$ 程度の数の smoothness を調べることになり使い物にならない。

- すると $c_2 = \frac{(c_1^2 - N)}{p}$ である。

このとき $M \equiv c_1 p^{-1} \pmod{N}$ である⁴⁰。

ベクトル

$$\mathbf{c} = (p, c_1, c_2)$$

の成分の大きさはおよそ $p, p/4, N/p$ 程度である。したがってこのベクトルの絶対値を小さくするには p を \sqrt{N} に近くとる必要がある。

直交補空間の最小ベクトルは LLL アルゴリズムを使わなくても explicit に記述できる。

- $c_1 r \equiv c_2 \pmod{p}$, $-\frac{p}{2} < r < \frac{p}{2}$ で r を定め

$$\begin{aligned} \mathbf{a} &= (c_1, -p, 0) \\ \mathbf{b} &= \left(\frac{c_1 r - c_2}{p}, -r, 1 \right) \end{aligned}$$

を定める。

すると $\mathbf{a} \times \mathbf{b} = -\mathbf{c}$ となり、 $\gcd(p, c_1, c_2) = 1$ であるから \mathbf{a}, \mathbf{b} は \mathbb{Z}^3 における \mathbf{c} の直交補空間の基底である⁴¹。実際、 $M \equiv c_1 p^{-1} \pmod{N}$ とすれば

$$\begin{aligned} \text{(a)} \quad & p(c_1 - pM + 0M^2) \equiv p(c_1 - c_1) = 0 \pmod{N} \\ \text{(b)} \quad & p^2 \left(\frac{c_1 r - c_2}{p} - rM + M^2 \right) \equiv p(c_1 r - c_2 - r c_1) + c_1^2 \equiv -p c_2 + p c_2 + N \equiv 0 \pmod{N} \end{aligned}$$

あとはこれから最小基底を計算することになる。 \mathbf{c} の絶対値は $O(N^{1/2})$ であるから最小基底の各ベクトルの絶対値は二つが同じような大きさでかつ直交していれば $O(N^{1/4})$ である。

実例をあげる。

$N = 1363$ の場合。

$p = 41$ とすると $c_1 = 16, c_2 = -27$ となる。 $r \equiv c_2 c_1^{-1} \pmod{p}$ より $r = 6$ である。

これより

$$\mathbf{a} = (16, -41, 0) \quad \text{と} \quad \mathbf{b} = (3, -6, 1)$$

である。これから最小基底を求めると

$$(3, -6, 1) \quad \text{と} \quad (2, 5, 6)$$

である。この多項式が前節であげたものである。このときは $M = 765$ である⁴²。

一般の次数 d でも同様に $O(N^{1/2d})$ のオーダーのものがみつかることが期待される。

Montgomery の提案を 3 次式に拡張することは難しいようである。

⁴⁰dual polynomial NFS では M を小さくする必要がない!!

⁴¹ここは \pmod{N} を考慮していない。

⁴²当然だが $M = 1363 - 765$ では第二成分の符号を変えたものが出る。

4.2 lattice sieve

Smooth な数 $a + bM$ は Factor Base の素数 p でどう分解されるだろうか。Factor Base を3つに分けて考えてみよう。

$$\begin{aligned}P_{low} &= \{p \mid p \leq p_{low}\} \\P_{mid} &= \{p \mid p_{low} < p < p_{high}\} \\P_{high} &= \{p \mid p_{high} \leq p\}\end{aligned}$$

$p \in P_{low}$ では高い確率で割れるし、 $p \in P_{high}$ に関してはごくたまにしか割り切れることはない。そして $p \in P_{mid}$ については次のようなことが起こらないだろうか。

p_{low}, p_{high} をうまくとると

各 $a + bM$ は P_{mid} の少なくとも一つの素数で必ず割り切れる。

このとき、ふるいは各 $q \in P_{mid}$ の倍数ごとに分けて行うことができる。

つまり

$$S = \{(a, b) \mid -H_a \leq a < H_a, 0 < b \leq H_b\}$$

でのふるいは次のような部分集合におけるふるいに分割される。

$q \in P_{mid}$ に対して

$$S_q = \{(a, b) \mid -H_a \leq a < H_a, 0 < b \leq H_b, a + bM \equiv 0 \pmod{q}\}$$

S_q では $a + bM$ がすでに q で割れることが分かっているのだから smoothness はかなり高くなる。

これが Pollard のオリジナルな lattice sieve の考え方である。

J.M. Pollard: *The lattice sieve*, The development of the number field sieve, Lecture Notes in Mathematics 1554(1993), 43–49.

オリジナルな lattice sieve の問題点として

- step q でふるいをするが q が大きいために一度に複数の b の間を渡ることになる。この b の交替が非常にコスト高である。

ということがある。そこで Huizing らは $b = 1$ に固定するという大胆な策に出た。

M.Huizing: *An implementation of the number field sieve*, Experimental Mathematics 5(1996), 231-253.

つまり

$$S_q = \{(a, 1) \mid -H_a \leq a < H_a, a \equiv -M \pmod{q}\}$$

である⁴³。

⁴³added on Jan.2002: 原文では“こうするともはや lattice というイメージはなくなるので line sieve と呼ぶ”と書いたが誤解だったようでこの用語は使われていない。むしろ普通の sieve を line-by-line sieve と呼び、略して line sieve とするのが一般的になった。

lattice sieve($b = 1$) の手順

1. p_{low}, p_{high} を適当にとり $P_{mid} = \{q_1, q_2, \dots, q_m\}$ を確定させる。
2. 配列 $S[0], S[1], \dots, S[r]$, $r = 2H_a/q_1$ を用意する (よほど小さな数でないかぎり全部用意することは不可能である。そのときは以下のふるいを分割して行う)。
3. 各 $q \in P_{mid}$ に対して次を行う。
 - $S[0], \dots, S[H_a/q]$ を 0 に初期化
 - $(-H_a + s_q) + M \equiv 0 \pmod{q}$ となる最小の s_q を計算する ($s_q + iq + M$ に $S[i]$ が対応する)
 - 各 $p \in P_{low} \cup P_{mid} \cup P_{high}$ に対して $s_q + iq + M \equiv 0 \pmod{p}$ となる $i = i_p$ を計算して $S[i_p], S[i_p + p], S[i_p + 2p], \dots$ に $\log_2 p$ を加える。
 - 値が十分大きな $S[i]$ に対して $s_q + iq + M$ を smooth data の候補として出力する ($q < p \in P_{mid}$ なる p で割り切れる数は二重登録を防ぐために捨てる)

最後の二重登録の防止はふるいのときにそういう p についてはその後の操作が無効になるような特別な値を置くことでやるのが良いかもしれない。

この方法では H_a が巨大になるので smooth な数が十分とれるかという懸念が出てくる。次の 4-primes large prime procedure を用いなければいけない。

4.3 Large prime procedure with 4 large primes

通常の large prime procedure では有理数側、代数的数側でそれぞれ1個の large prime を用いる ((1+1)LP)。これは二次ふるい法での1個の large prime を用いた場合 (1LP) に相当する。二次ふるい法では 80桁を超えるものについては2個の large primes を用いること (2LP) が有効であった。これの NFS 版が有理数側、代数的数側でそれぞれ2個の large primes を用いるもの ((2+2)LP) である。

この方法は非常にコストがかかるため、有理数側は1個だけで代数的数の側では2個にするなどの限定的な方法がとられることが多かった。

しかし lattice sieve と組み合わせると4個を用いても効率的に実行できることを

R.A.Golliver, A.K.Lenstra and K.S.McCurley: *Lattice sieving and trial division*, ANTS'94, Lecture Notes in Computer Science **877**(1994), 18–27.

が示した。

B.Dodson and A.K.Lenstra: *NFS with four large primes: An Explosive Experiment*, Advances in Cryptology-CRYPTO'95, Lecture Notes in Computer Science **963**(1995), 372–385.

によると、large primes を持つデータを組み合わせることができる Fully Factored なデータの個数はある時点で突然爆発的に増加する。

4.4 Block Lanczos method

平方数を作る組み合わせを求めるために行列の計算を行うが、通常は Gauss 消去法である。

しかし、行列のサイズが大きくなるとこれは時間的に不可能になってしまう。この難点を避けるために消去系の方法を止めて、一般の工学における大型の疎行列で盛んに用いられる反復系のアルゴリズムを取り入れることになった。

ところが、我々の計算は mod 2 で行われるので一般の実あるいは複素数値の計算をそのまま持ってくることはできない⁴⁴。Montgomery は Lanczos 法で複数のベクトルを同時に扱う (block 化) ことにすれば計算がうまく進むことを示し、実際に計算時間の縮小に成功した。

P.L.Montgomery: *A block Lanczos algorithm for finding dependencies over GF(2)*, Advances in Cryptology-EUROCRYPTO'95, Lecture Notes in Computer Science **921**(1995), 106–120.

⁴⁴実の場合は正定値対称行列という条件をおく。つまり 0 でないベクトルのその行列により定義されるノルムは 0 でないということであるが、mod 2 の計算ではそのような行列はない。

5 Number Field Sieve の計算量

Lenstra, Lenstra, Manasse and Pollard: *The Number Field Sieve*, Lecture Notes in Mathematics 1554(1993).

にしたがって簡単に述べる。ここでの結論はすべて予想である。

5.1 SNFS の計算量

記号

$$L_x[\nu, \lambda] = \exp(\lambda(\log x)^\nu(\log \log x)^{1-\nu})$$

例

$$\begin{aligned}x &= \exp(\log x) &= L_x[1, 1] \\x^\lambda &= \exp(\lambda \log x) &= L_x[1, \lambda] \\(\log x)^\lambda &= \exp(\lambda \log \log x) &= L_x[0, \lambda]\end{aligned}$$

つまり $\log x$ を基準に考えると、第一パラメータ ν が 1 のものが指数時間で 0 のものが多項式時間である。この中間でできるだけ 0 に近づけるのが目標である⁴⁵。

定理 $L_x[\nu, \lambda]$ 以下の自然数が $L_x[\omega, \mu]$ -smooth である確率は

$$L_x[\nu - \omega, -(\nu - \omega)\lambda/\mu + o(1)] \quad \text{as } x \rightarrow \infty$$

である。

議論を省略していきなり最適値をとるところから始める。
分解したい数 N に対して次数 d を次のようにとる⁴⁶。

$$d = (3/2)^{1/3} \left(\frac{\log N}{\log \log N} \right)^{1/3}$$

Factor Base の個数、ふるいの範囲を共に

$$L_N[1/3, (2/3)^{2/3}]$$

とすれば $a + bM$ と $\mathcal{N}(a + b\theta)$ の大きさはおよそ

$$L_N[2/3, (2/3)^{1/3}]$$

である⁴⁷。定理によればこれが $L_N[1/3, (2/3)^{2/3}]$ -smooth になる確率は

$$L_N[1/3, -(1/3)(2/3)^{1/3}/(2/3)^{2/3}] = L_N[1/3, -(1/18)^{1/3}]$$

⁴⁵もちろん 0 が究極の目標だがそれはちょっと望みすぎだろう。

⁴⁶計算すると 35桁から 3、101桁から 4、227桁から 5、432桁から 6。これはあくまでオーダーの低い要素を無視した概算であるが 100桁前後では 4次式が適当だというのは本当らしい。

⁴⁷ここで多項式の係数は無視できるだけ小さいとしている。ここが実行時間の解析に special という仮定を使っているところ。

となる。したがって同時に smooth になる確率は

$$\left(L_N[1/3, -(1/3)(2/3)^{1/3}/(2/3)^{2/3}]\right)^2 = L_N[1/3, -(4/9)^{1/3}]$$

である。こういうものを Factor Base の個数個以上集めるのだから

$$\text{FactorBaseの個数/確率} = L_N[1/3, (32/9)^{1/3}]$$

だけの手間がかかることになる。これが大雑把に考えた Special NFS の実行時間である。

5.2 GNFS の計算量

多項式 $f(x)$ の係数を新たに考慮しなければならなくなる。

d を固定し

$$M^{d+1} \geq N$$

となるように M を定めれば N を M 進展開して d 次の多項式 $f(X)$ を作ることができる。このとき $f(X)$ の係数は $N^{1/(d+1)}$ 以下である。したがって

$$\begin{aligned} c_d \mathcal{N}(a + b\theta) &= (-b)^d f(-a/b) \\ &= c_d a^d + c_{d-1} a^{d-1} (-b) + \dots + c_r a^r (-b)^{d-r} + \dots + c_0 (-b)^d \end{aligned}$$

の絶対値はおよそ $O(N^{1/(d+1)} \max(a, b)^d)$ である。

ここで次数 d を次のようにとる。 λ は最後に最適値を決める。最適値は $(1/3)^{1/3}$ であるので最初からそういうつもりで読んで構わない。

$$d = \lambda^{-1} \left(\frac{\log N}{\log \log N} \right)^{1/3}$$

Factor Base の個数、ふるいの範囲を共に

$$L_N[1/3, 2\lambda^2]$$

とすれば $\mathcal{N}(a + b\theta_1)$ の大きさはおよそ

$$\begin{aligned} &L_N[1/3, 2\lambda^2]^d \cdot N^{1/(d+1)} \\ &= \exp\left(2\lambda(\log N)^{2/3}(\log \log N)^{1/3}\right) \cdot \exp\left(\lambda(\log N)^{2/3}(\log \log N)^{1/3}\right) \\ &= L_N[2/3, 3\lambda] \end{aligned}$$

であり $M = N^{1/(d+1)}$ の大きさはおよそ

$$\begin{aligned} &N^{1/(d+1)} \\ &= \exp\left(\lambda(\log N)^{2/3}(\log \log N)^{1/3}\right) \\ &= L_N[2/3, \lambda] \end{aligned}$$

である。

定理によればこれらが同時に $L_N[1/3, 2\lambda^2]$ -smooth になる確率は

$$L_N[1/3, -3(1/3)/(2\lambda)] \cdot L_N[1/3, -(1/3)/(2\lambda)] = L_N[1/3, -2/(3\lambda)]$$

となる。smooth data がこの a - b 区間内に Factor Base の個数個以上なければならぬから

$$L_N[1/3, -2/(3\lambda)] L_N[1/3, 2\lambda^2]^2 \geq L_N[1/3, 2\lambda^2]$$

である。したがって

$$\lambda^3 \geq 1/3$$

smooth data は Factor Base の個数個以上集めるのだから

$$(\text{FactorBase の個数})/(\text{確率}) = L_N[1/3, 2\lambda^2 + (2/3)/\lambda] \geq L_N[1/3, (64/9)^{1/3}]$$

だけの計算量になる。

5.3 Dual Polynomial General NFS の実行時間

二つの d 次多項式 $f_1(x), f_2(x)$ の係数の大きさは共におよそ $N^{1/(\mu d)}$ にとれたとしよう。ここで次数 d を次のようにとる。

$$d = \lambda^{-1} \left(\frac{(1 + o(1)) \log N}{\log \log N} \right)^{1/3}$$

Factor Base の個数、ふるいの範囲を共に

$$L_N[1/3, \nu\lambda^2]$$

とすれば $\mathcal{N}(a + b\theta_1)$ と $\mathcal{N}(a + b\theta_2)$ の大きさはおよそ

$$\begin{aligned} & L_N[1/3, \nu\lambda^2]^d \cdot N^{1/(\mu d)} \\ &= \exp\left(\nu\lambda(\log N)^{2/3}(\log \log N)^{1/3}\right) \cdot \exp\left((\lambda/\mu)(\log N)^{2/3}(\log \log N)^{1/3}\right) \\ &= L_N[2/3, (\nu + 1/\mu)\lambda] \end{aligned}$$

である。この場合はふるいの範囲と多項式の係数の大きさが競合関係にあり、 λ を上記のように割り振れば両者が揃った大きさになる。

定理によればこれらが同時に $L_N[1/3, \nu\lambda^2]$ -smooth になる確率は

$$(L_N[1/3, -(\nu + 1/\mu)/(3\nu\lambda)])^2 = L_N[1/3, -2(\nu + 1/\mu)/(3\nu\lambda)]$$

となる。smooth data がこの a - b 区間内に Factor Base の個数個以上なければならぬから

$$L_N[1/3, -2(\nu + 1/\mu)/(3\nu\lambda)] L_N[1/3, \nu\lambda^2]^2 \geq L_N[1/3, \nu\lambda^2]$$

である。したがって

$$\lambda^3 \geq 2(\nu + 1/\mu)/(3\nu^2)$$

smooth data は Factor Base の個数個以上集めるのだから

$$\begin{aligned} (\text{FactorBase の個数})/(\text{確率}) &= L_N[1/3, \nu\lambda^2 + 2(\nu + 1/\mu)/(3\nu\lambda)] \\ &\geq L_N[1/3, (32/9)^{1/3}((\nu + 1/\mu)^2/\nu)^{1/3}] \end{aligned}$$

だけの計算量になる。右辺は $\nu = 1/\mu$ で最小値 $L_N[1/3, (128/(9\mu))^{1/3}]$ をとる。

$\mu = 2$ にとれば

$$L_N[1/3, (64/9)^{1/3}]$$

になり、より大きな μ が選べればより少ない計算量になる。

5.4 理想的な Dual Polynomial General NFS の実行時間

ではこれが理想的（奇跡的）に行われた場合を考えよう。

ある自然数 q と微小正数 ε に対して次数 d を

$$d = \left(\frac{(q + \varepsilon) \log N}{(q - 1) \log \log N} \right)^{1/q}$$

として定めたとき二つの d 次多項式 $f_1(x), f_2(x)$ の係数の絶対値が共に

$$L_N[1/(q + \varepsilon), O(1)]$$

のオーダーで抑えられたとしよう（そういう N の無限系列があったとしよう）⁴⁸。

Factor Base の個数、ふるいの範囲を共に

$$L_N[1/q, \lambda]$$

とすれば $\mathcal{N}(a + b\theta_1)$ と $\mathcal{N}(a + b\theta_2)$ の大きさはおよそ

$$\begin{aligned} L_N[1/q, \lambda]^d &= \exp\left(\lambda(q/(q-1))^{1/q}(\log N)^{2/q}(\log \log N)^{(q-2)/q}\right) \\ &= L_N[2/q, \lambda(q/(q-1))^{1/q}] \end{aligned}$$

である⁴⁹。定理によればこれらが同時に $L_N[1/q, \lambda]$ -smooth になる確率は

$$\left(L_N[1/q, -(1/q)(q/(q-1))^{1/q}] \right)^2 = L_N[1/q, -(2/q)(q/(q-1))^{1/q}]$$

となる。smooth data がこの a - b 区間内に Factor Base の個数個以上なければならぬから

$$L_N[1/q, -(2/q)(q/(q-1))^{1/q}] L_N[1/q, \lambda]^2 \geq L_N[1/q, \lambda]$$

である。したがって

$$\lambda \geq (2/q)(q/(q-1))^{1/q}$$

smooth data は Factor Base の個数個以上集めるのだから

$$L_N[1/q, (2/q)(q/(q-1))^{1/q} + \lambda] \geq L_N[1/q, (4/q)(q/(q-1))^{1/q}]$$

となって最小値をとって計算量はおよそ $L_N[1/q, (4/q)(q/(q-1))^{1/q}]$ である。 q は任意であるから多項式時間に限りなく近いと言っても良いであろう。

⁴⁸もっともわかりやすいのは N がいくら大きくなっても係数が全部 32bit にとれるとき、係数が $\log N$ の有限乗で抑えられるときなど。

⁴⁹係数は L_N 関数の第一因子がわずかに小さいので無視される。