数理解析研究所講究録 1456 巻 2005 年 151-158

数論システム NZMATH における複数多項式二次篩法 (MPQS) の実装について

熊木幸司 *

KOUJI KUMAKI

東京都立大学理学研究科

DEPARTMENT OF MATHMATICS, TOKYO METROPOLITAN UNIVERSITY

1 基本原理

篩による素因数分解の基本的なアイディアは, 合成数 N に対してなんらかの方法により

$$x^2 \equiv y^2 \pmod{N}, \quad x \not\equiv \pm y \pmod{N}$$

なる x, y を見つけたならば,

$$(x-y)(x+y) \equiv 0 \pmod{N}, \quad x \pm y \not\equiv 0 \pmod{N}$$

であるから $\gcd(x-y, N)$ を計算する事で N の真の約数が求まるということである. しかしながら, このような x, y は直ちにわかるはずもないので, 問題はどのようにしてこの x, y を見つけていくのかということになってくる.

2 因子基底

そこでまず, 素数の集合を $\mathbb{P}$ として, ある自然数 f 以下の相異なる小さな素数からなる集合

$$B = \{p_i \mid p_i \leq f, p_i \in \mathbb{P}, 1 \leq i \leq k\}$$

を考える. この集合は因子基底 (Factor Base) と呼ばれる.

つぎに,

$$a = \prod_{p_i \in B} p_i^{e_i}$$

とする. このような a は一般に f-smooth な数と呼ばれる.

ここで

$$a_i = \prod_{j=1}^{k} p_i^{e_{i,j}}, \quad a_i \equiv {b_i}^2 \pmod{N} \tag{2.1}$$

なる a_i, b_i を $r(\geq 1+k)$ 個集めて, その中でいくつかの a_i に対して積をとると

*bearandtree2000@yahoo.co.jp

$$\prod_i a_i = \prod_i \prod_{j=1}^{k} p_j^{e_{i,j}} = \prod_{j=1}^{k} p_j^{\sum_i e_{i,j}}$$

となる. ここで a_i, b_i を $r(\geq 1+k)$ 個集めているのでこの中には必ず自明でない線型従属な関係が存在する. したがってすべての j に対して

$$\sum_i e_{i,j} \equiv 0 \pmod 2 \tag{2.2}$$

となる $e_{i,j}$ が存在する. これより

$$\prod_i a_i, \quad \prod_{i=1}^{m} b_i^2 \pmod N$$

はともに平方数となるので,

$$x = \prod_{j=1}^{k} p_j^{(\sum_i e_{i,j})/2}, \quad y = \prod_{i=1}^{m} b_i \pmod N$$

とし $\gcd(x-y, N)$ を計算すればよいすればよい. ここでの問題は大きく分けて二つある.

1. 式 (2.1) ような a_i, b_i をどのようにして選べばよいか
2. 式 (2.2) のような i をどのようにして選べばよいか

ということである.

3 二次篩

3.1 二次篩のアイディア

第 2 節の式 (2.1) の a_i, b_i を効率よく求めていくために考えられたのが二次篩と呼ばれるものである. この方法のアイディアを以下に記す.

ここではまず N を奇合成数として

$$Q(x) = x^2 - N$$

とする. ただし $x \in \mathbb{Z}$ は, ある選ばれた数 M (N に比べて非常に小さい数) に対して区間 $\left[-M+\lfloor\sqrt{N}\rfloor, M+\lfloor\sqrt{N}\rfloor\right]$ からとる. このとき $|Q(x)| \approx 2M\sqrt{N}$ である.

つぎに, 以下の条件を満たす小さな素数の集合

$$B = \left\{ p_i \middle| \left(\frac{N}{p_i}\right) = 1,\ p_i \in \mathbb{P}, i = 1, 2, \ldots, r-1 \right\} \cup \{-1\} \tag{3.1}$$

を考える. これは二次篩における因子基底である. ここで r はある選ばれた数とする. また負の数を考えるために $-1 \in B$ とし, $p_r = -1$ とする. このとき $Q(x) \equiv x^2 \pmod N$ であるから式 (2.1) の b_i については, $x = b_i$ とすれば問題ないので, あとはどのようにして $Q(x)$ の中から B-smooth[1] ものを見つけていけばよいかということになる.

---

[1] 以下では $\prod_{p_i \in B} p_i^{e_i}$ なるものを B-smooth と呼ぶことにする

これに対しては, エラトステネスの篩を真似ればよい. 今, 各 $p_i \in B$ に対して

$$Q(x) \equiv 0 \pmod{p_i} \Longleftrightarrow x^2 \equiv N \pmod{p_i}$$

つまり N の法 p_i における平方根 $\left(\left(\frac{N}{p_i}\right) = 1$ であるから必ず根を持つ$\right)$ をそれぞれ $x_{i1}, x_{i2} \in \left[-M + \lfloor\sqrt{N}\rfloor, -M + \lfloor\sqrt{N}\rfloor + p_i\right]$ とすればこのそれぞれの解から p_i おきに $p_i | Q(x)$ となる x が見つけられる (p_i の巾についても同様に考えられる). また $Q(x)$ の値が負の場合はその x は因子基底 p_r の部分を 1 にする. これをすべての p_i について行えば $x \in \left[-M + \lfloor\sqrt{N}\rfloor, M + \lfloor\sqrt{N}\rfloor\right]$ のなかで $Q(x)$ が B-smooth となる x がわかる.

3.2 篩の方法

実際にどのようにして B-smooth な $Q(x)$ を求めていくのか, 具体的な篩の方法を以下に記す.

1. 各 $x \in \left[-M + \lfloor\sqrt{N}\rfloor, M + \lfloor\sqrt{N}\rfloor\right]$ に対応した位置に 0 を置いたテーブル (2M 個の 0 があるテーブル) を用意する. さらに, 各 $x \in \left[-M + \lfloor\sqrt{N}\rfloor, M + \lfloor\sqrt{N}\rfloor\right]$ に対して $\log_e Q(x)$ の値をとったテーブルも用意する.

2. 各 $p_i \in B$ に対して $\log_e p_i$ の値を求める.

3. 合同式 $x^2 \equiv N \pmod{p_i}$ の解 x_1, x_2 を $\left[-M + \lfloor\sqrt{N}\rfloor, -M + \lfloor\sqrt{N}\rfloor + p_i\right]$ の範囲でとる. そして x_1, x_2 に対応したテーブルの位置に $\log_e p_i$ を足した後, そこから p_i おきに $\log p_i$ を足していく. また p_i の巾 ($p_i{}^k \leq 2M$) に対しても同様に解を求め, そこから $p_i{}^k$ おきに $\log p_i$ を足していく. もし $Q(x)$ の値が負の場合はその x は因子基底 p_r の部分を 1 にする.

4. そして
$\left(x \in \left[-M + \lfloor\sqrt{N}\rfloor, M + \lfloor\sqrt{N}\rfloor\right]$ に対応した位置のテーブルの値$\right) >$ ($\log_e Q(x)$ のテーブルの値)
となるようなものをピックアップしていく. ピックアップされた $Q(x)$ は B-soomth となる. またこの $Q(x)$ に対して因子基底の素数で試し割り算を行い, 因子基底の積の形で表したときの各因子基底の指数部分を取り出してリストにしておく.

後はこのようにして求めた B-smooth な $Q(x)$ をいくつか組み合わせて, 平方数となるようなものを選べばよい.

3.3 二次篩法の例

今 $N = 17873$, $M = 50$, $r = 5$ ($x \in [83, 183]$, $B = \{-1, 2, 7, 11, 23\}$) とする.
この場合上記の方法で B-smooth なものをピックアップし, その $Q(x)$ の因子基底の指数部分のリストをならべると次の表のとおりである.

x	$Q(x)$	-1	2	7	11	23
87	-10304	1	6	1	0	1
129	-1232	1	4	1	1	0
133	-184	1	3	0	0	1
135	352	0	5	0	1	0
137	896	0	7	1	0	0
143	2576	0	4	1	0	1
151	4928	0	6	1	1	0
179	14168	0	3	1	1	1

$\Longrightarrow$

x	$Q(x)$	-1	2	7	11	23
87	-10304	1	6	1	0	1
133	-184	1	3	0	0	1
135	352	0	5	0	1	0
151	4928	0	6	1	1	0

この中から右の表の 4 つを見れば, 各 p_i の指数を足すと偶数になるので,

$$\begin{aligned} X &= 87 \cdot 133 \cdot 135 \cdot 151 \\ Y &= 2^{10} \cdot 7 \cdot 11 \cdot 23 \end{aligned}$$

とすれば,

$$X^2 \equiv Y^2 \pmod{17873}.$$

よって $\gcd(X-Y, 17873) = 61$ となり因数が求まる.

4 複数多項式二次篩

4.1 複数多項式二次篩のアイディア

二次篩法に改良を加えさらに効率のよい方法として考えられたのが, 複数多項式二次篩 (MPQS) である. 二次篩法との最大の違いは, 篩の対象となる多項式の違いである. 因子基底に関しては式 (3.1) と同じものを用いる. 複数多項式二次篩のアイディアを以下に記す.

まず N を奇合成数 $a, b, c, d \in \mathbb{Z}$ として, 多項式 $F(x) = ax^2 + bx + c$, $b^2 - 4ac = N$, $a = d^2$ (a は平方数) を考える. このとき,

$$\begin{aligned} (2d)^2 F(x) &= 4a^2x^2 + 4abx + 4ac \\ &= (2ax+b)^2 - N \\ &\equiv (2ax+b)^2 \pmod{N}. \end{aligned}$$

ここで d を $\gcd(2d, N) = 1$ となるようにとり, 法 N における $2d$ の逆元を y とすれば,

$$F(x) \equiv (y(2ax+b))^2 \equiv H^2 \pmod{N} \tag{4.1}$$

であるから, あとは二次篩法同様に $H = b_i$ として $F(x)$ の中から B-smooth となるものを見つけてくればよい.

このように $F(x)$ をとることによってどんなメリットがあるのか. 最大のメリットは

- 二次式 $F(x)$ の係数を変えても式 (4.1) が成り立っているので, どの $F(x)$ に対しても同じ因子基底で篩にかけることができる

ということである. したがって B-smooth なものを目的の個数分集めるためにたくさんの多項式を用いれば, それぞれで動かす x の範囲は二次篩法のそれと比べて小さくすることができるのである. もうひとつのメリットは

- 変数 x の動かす範囲を $[-M, M]$ として $a \approx \sqrt{N}/\sqrt{2}M$ とすれば $|F(x)| \approx M\sqrt{N}/2\sqrt{2}$

ということである. 二次篩の場合 x を同じ範囲 $2M$ で動かしたとき $|Q(x)| \approx 2M\sqrt{N}$ であったので若干 $F(x)$ のほうが小さい値としてとれるのである.

4.2 係数の決め方

では, 具体的に多項式の係数をどのように決めるのか. まず N が奇合成数であり $b^2 - 4ac = N$ であるから b は奇数でなければならないので $b^2 \equiv 1 \pmod 4$. よって $N \equiv 1 \pmod 4$ としなければならない. そこで $N \equiv 3 \pmod 4$ となる N に対しては $kN \equiv 1 \pmod 4$ なる適当な k をとってくる. この k を multiplier と呼ぶことにする. もし $N \equiv 1 \pmod 4$ の場合は $k = 1$ とする. 以下の目標としては最初に a を決め, そこから $b^2 \equiv kN \pmod a$ となる b を決めることである.

そこでまず M を篩の範囲として, 最初に $a \approx \sqrt{kN}/\sqrt{2}M$ となるように選びたいので d を $\sqrt{\sqrt{kN}/\sqrt{2}M}$ に近い素数で $d \equiv 3 \pmod 4$, $\left(\dfrac{kN}{d}\right) = 1$ となるように選ぶ.

ここで, 今

$$\begin{aligned} h_0 &\equiv (kN)^{(d-3)/4} \pmod d, \\ h_1 &\equiv h_0 \cdot kN \pmod d \end{aligned}$$

とする. このとき,

$$\begin{aligned} {h_1}^2 &\equiv kN \cdot kN \cdot {h_0}^2 \pmod d \\ &\equiv kN \cdot (kN)^{(d-1)/2} \pmod d \\ &\equiv kN \pmod d. \end{aligned}$$

また $h_0 h_1 \equiv {h_0}^2 \cdot kN \equiv (kN)^{(d-1)/2} \equiv 1 \pmod d$ より法 d における h_1 が h_0 の逆元である.
そして,

$$\begin{aligned} h_2 &\equiv (2h_1)^{-1} \left(kN - {h_1}^2\right)/d \pmod d, \\ b &\equiv h_1 + h_2 d \pmod a \end{aligned}$$

とすれば

$$\begin{aligned} b^2 &\equiv {h_1}^2 + 2h_1 \cdot h_2 d + {h_2}^2 d^2 \pmod a \\ &\equiv {h_1}^2 + kN - {h_1}^2 \pmod a \\ &\equiv kN \pmod a. \end{aligned}$$

ここでもし b が偶数ならば a は奇数なので $b - a$ を新たに b とすることで $b^2 \equiv kN \pmod a$ をみたす奇数となる. また c については $c = (b^2 - kN)/4a$ で求まる.

ここで複数多項式二次篩法についてのアルゴリズムを大まかにまとめると以下のようになる.

1. まず k を決める. この k は $kN \equiv 1 \pmod 4$ を満たし $\left(\dfrac{kN}{p_i}\right) = 1$, $p_i \in B$ なる p_i がなるべく多くとれるものが望ましい. とりわけ $\left(\dfrac{kN}{2}\right) = 1$ とするには $kN \equiv 1 \pmod 8$ となる k を選ぶ.
2. 第 4.2 節の手順に従い, 多項式を決定する.

(a) まず $\sqrt{\sqrt{kN}/\sqrt{2M}}$ に近い素数で $d \equiv 3 \pmod 4$, $\left(\frac{kN}{d}\right) = 1$ なる d を選び $a = d^2$ とする.

(b) 次に $h_0 \equiv (kN)^{(d-3)/4} \pmod d$, $h_1 \equiv h_0 \cdot kN \pmod d$, $h_2 \equiv (2h_1)^{-1}\left(kN - {h_1}^2\right)/d \pmod d$ をそれぞれ計算し $b \equiv h_1 + h_2 d \pmod a$ とする.

3. 各 $p_i \in B$ に対して $F(x) \equiv 0 \pmod{p_i}$ なる x を求めて, 篩を行い B-smooth な $F(x)$ を探す.

4. B-smooth なものが必要な個数集まるまで (ii),(iii) を何回か繰り返し, その中から $\prod F(x)$ が平方数になるようなものを探す.

5. 式 (4.1) より $s^2 \equiv t^2 \pmod N$ なる s, t が求められるので $\gcd(s-t, N)$ が自明でない因数ならば終了.

5 複数多項式二次篩の実装

第 4 節を元に複数多項式二次篩の実装を行った. このプログラムは数論システム NZMATH の factor モジュールのプログラム mpqs として使用されている. 数論システム NZMATH では Python という言語で開発が行われており, したがって今回の実装も Python 言語でプログラミングされている. それぞれの詳細は数論システム NZMATH については [12] と [13], Python 言語については [14] を参照のこと.

5.1 プログラムの仕様

このプログラムは, 合成数 N を入力値として受け取り N が完全に素因数分解できた場合には素因数すべて, できなかった場合は見つかった素因数と因数を出力するプログラムである. 起動方法は NZMATH がインストールされている状態で Python を起動した後,

```
>>>import nzmath
>>>nzmath.factor.mpqs(N)
```

と入力することでプログラムが起動する. また, 引数としては合成数 N だけでなく, 篩の範囲 (M とする), 因子基底の数 (F とする) を指定することも可能である. 篩の範囲, 因子基底の数を指定する場合には,

```
>>>nzmath.factor.mpqs(N,M,F)
```

と入力すればよい.

またプログラム実行時には multiplier, 篩の範囲, 因子基底の個数, 多項式を変えた回数数とそれに要した時間, 得られた B-smooth なものの個数とそれに要した時間, ガウスの消去法に要した時間とそれにより見つかった解の個数が各行程ごとに随時表示されるようになっている. この各実行行程の詳細については, http://tnt.math.metro-u.ac.jp/labo/master/2004/kumaki/mypaper.pdf を参照のこと.

5.2 実際の使用例

実際にプログラムを使用したときの例を以下に記す. 入力値である合成数を 354190525335205945979 4529 としたときの実行結果である.

```
The number is  354190525335205945979 4529 MPQS starting
25 - disits Number
Multiplier is 601
Sieve range is [ -5000 , 5000 ] , Factorbase size = 181 , Max Factorbase 2393
*/ Total 23 times changing poly report /*
Time of deciding coefficient = 0.192363500595 sec
Sieving Time =  4.71326851845 sec
Found smooth numbers are 186 / 181
Total time of getting enough smooth numbers =  4.90708303452 sec
Time of Gaussian Elimination =  0.751587867737 sec
Found 18 liner dependent relations
Total time = 5.971752882 sec
Factored completely !
[(830613846817L,1) ,(4264202031937L,1)]
```

また篩の範囲を 1000 , 因子基底の個数を 150 としたときの実行結果は次のようになる.

```
>>> nzmath.factor.mpqs(354190525335205945979 4529,1000,150)
The number is  354190525335205945979 4529 MPQS starting
25 - disits Number
Multiplier is 601
Sieve range is [ -1000 , 1000 ] , Factorbase size = 151 , Max Factorbase 1931
*/ Total 71 times changing poly report /*
Time of deciding coefficient = 0.479235649109 sec
Sieving Time =  2.9754652977 sec
Found smooth numbers are 151 / 151
Total time of getting enough smooth numbers =  3.45796322823 sec
Time of Gaussian Elimination =  0.459499120712 sec
Found 10 liner dependent relations
Total time = 4.05427694321 sec
Factored completely !
[(830613846817L,1), (4264202031937L,1)]
```

篩の部分の途中経過は多項式を 50 回変えるたびに記すようになっている.

参 考 文 献

[1] Richard Crandall and Carl Pomerance. *Prime numbers.* Springer-Verlag, New York, 2001.

[2] Neal Koblitz. *A course in number theory and cryptography*, Graduate Texts in Mathematics 114. Springer-Verlag, New York, second edition, 1994.

[3] A.K.Lenstra and H.W.Lenstra Jr.(Eds.) *The development of number field sieve*,Lecture Notes in Mathematics 1554 Springer-Verlag, Berlin , 1993.

[4] A.K.Lenstra and M.S.Manasse. *Factoring with tow large primes*, Math.Comp.63 (1994) ,No.208 ,785–798

[5] P.Leyland, A.Lenstra, B.Dodson, A.Muffett, and S.Wagstaff. *MPQS with Three Large Primes* Alogorithmic number theory (Sydney,2002), 446–460, Lecture Notes in Computer Science 2369. Springer-Verlag, Berlin , 2002.

[6] Peter L. Montgomery. *A BLock Lanczos Algorithm for Finding Dependencies over GF(2)*, Advances in cryptology—EUROCRYPTO'95(Saint-Malo,1995), 106–120, Lecture Notes in Computer Science 921 . Springer-Verlag, Berlin , 1995.

[7] Ken Nakamula. *A Survey on the Number Field Sieve*, Number Theory and its Application , Kluwer Academic Publishers ,1999, 263–272

[8] Carl Pomerance. *The quadratic sieve factoring algorithm*, Advances in cryptology—EUROCRYPTO'84(Paris,1984), 169–182, Lecture Notes in Computer Science 209 . Springer-Verlag, Berlin , 1985.

[9] Robert D. Silverman *The multiple polynomial quadratic sieve* Math. Comp. 48 (1987) , No.177 329–339.

[10] 伊豆哲也 , 木田祐司. 素因数分解の現状について , 日本応用数理学会論文誌 Vol.13 , No2 , 2003 , 289–303.

[11] 木田祐司. 暗号アルゴリズムの詳細評価に関する報告書 , `http://www2.nict.go.jp/ns/s801/102/PDF/outrep_data/rep_ID0021.pdf`.

[12] 中村憲. 数論アルゴリズムの研究と数論システムの開発 , 研究成果報告集 , 2003

[13] `http://tnt.math.metro-u.ac.jp/nzmath/`

[14] `http://www.python.org/` , `http://www.python.jp/Zope/`