数理解析研究所講究録
986巻 1997年 110-117

一変数多項式の因数分解の効率化[1)]

– 因子の個々の係数上限の利用 –

筑波大学大学院 理工学研究科 塚田 康弘
筑波大学 数学系 & ベンチャービジネスラボ 佐々木 建昭
(Yasuhiro Tsukada & Tateaki Sasaki)

Abstract. 多項式の因数分解に関しては、例外的な場合を除き、アルゴリズムの骨格は限界近くまで効率化されていると言ってよい。しかし、計算機への実装(インプリメンテーション)の観点から細部を検討すると、なお改善の余地がある。本稿では、因子多項式の各係数の上限を利用することにより、計算を効率化させる方法を提示し、それが極めて有効であることを実証する。

1. 現行の因数分解アルゴリズム

因数分解すべき多項式を $F(X) \in \mathbf{Z}[X]$ とし、その既約因子を $G(X)$ とする。

$$F(X) = a_0 X^d + a_1 X^{d-1} + \cdots + a_d \tag{1}$$

$$G(X) = b_0 X^q + b_1 X^{q-1} + \cdots + b_q \tag{2}$$

次に述べるアルゴリズムは、$\mathbf{Z}_p$ 上での因子を並列的に $\mathbf{Z}_{p^{k+1}}$ まで Hensel 構成し、$\mathbf{Z}_{p^{k+1}}$ 上での因子を組み合わせて $\mathbf{Z}$ 上での既約因子を見出すものである。

Step1 [$\mathbf{Z}_p$ 上での因数分解]

素数 p を、$F(X)$ と dF/dX の終結式 $\mathrm{res}(F, dF/dX)$ の約数でないように選ぶ。

Berlekamp のアルゴリズム (参考文献 4 参照) を用いて $\mathbf{Z}_p$ 上で F を既約因子に分解する。

$$F(X) \equiv F_1^{(0)}(X) \cdots F_r^{(0)}(X) \pmod{p} \tag{3}$$

Step2 [Hensel 構成]

因子 $G(X)$ の係数全体の上限 B を次式で計算する (Landau の上限)。

$$|b_0| + |b_1| + \cdots + |b_q| \;\le\; B \stackrel{\mathrm{def}}{=} 2^q \, \|F(X)\|_2 \tag{4}$$

k を $p^{k+1} \ge 2B$ をみたす最小の整数とし、以下のように Hensel 構成を実行する。

$$F(X) \equiv F_1^{(k)}(X) \cdots F_r^{(k)}(X) \pmod{p^{k+1}} \tag{5}$$

1) 本研究は部分的に文部省科研費 (課題番号 06558037) およびベンチャービジネスラボの援助を受けた。

Step3 [試し割り]
下記を $\{1, F_1^{(k)}, \cdots, F_r^{(k)}\}$ の要素のあらゆる積 (これを $G(X)$ とする) について実行する。

1. [積計算] 実際に $G(X)$ を計算する。
2. [試し割り] $\mathbf{Z}$ 上で、$G(X)$ で $F(X)$ を試し割りをする。もし割り切れば、$G(X)$ は $F(X)$ の $\mathbf{Z}$ 上の因子である。割り切らなければ、$G(X)$ を破棄する。
3. 割り切る $G(X)$ が一つもなければ、$F(X)$ は $\mathbf{Z}$ 上で既約である。

(注) 上記は、実は $F(X)$ がモニックな場合に対するアルゴリズムである。実際には以下のようになる。

まず、Step1 ではモニック多項式 $F(X)/a_0$ を $\mathbf{Z}_p$ 上で因数分解する。そして、$G(X)$ の代わりに $\tilde{G}(X) = (a_0/b_0)G(X)$ を構成する。したがって、Step3 では要素の組み合わせの積に a_0 をかけたものを $\tilde{G}$ とし、$\tilde{G}$ で a_0F を試し割りすることになる。以下では、簡単のため、この主係数問題にはふれないことにする。また、上記では同次数因子積への分解を考慮していない。

2. 因子の各係数ごとの上限の導入

(4) 式で用いられている B は、因子の全係数の絶対値の和の上限であるが、実はかなりの過大評価を与えてしまう。これに対して、我々が提案するのは、各係数ごとの絶対値の上限の利用である。

(2) 式の因子の係数 b_i $(i = 1, 2, \cdots, q)$ に対し、我々は次の上限 B_i' を定めた (証明は 7 章で述べる)。

$$|b_i| \leq B_i' \stackrel{\text{def}}{=} |a_0| \binom{q}{i} (\min\{\tilde{z_1}, \tilde{z_2}\})^i \tag{6}$$

ここで、$\tilde{z_1}$, $\tilde{z_2}$ は $F(X)$ の根の上限であり、以下で与えられる (これも証明は 7 章で述べる)。

$$\tilde{z_1} \stackrel{\text{def}}{=} 1 + \frac{\max\{|a_1|, \cdots, |a_d|\}}{|a_0|} \tag{7}$$

$$\tilde{z_2} \stackrel{\text{def}}{=} \max\{\sqrt[j]{m|a_j|/|a_0|} \mid j = 1, 2, \cdots, d\} \tag{8}$$

ただし、m は $a_1, a_2, \cdots, a_d$ 中で 0 でないものの個数である。i が小さいとき、この上限は B よりはるかに小さい値になることが多い。(注意. (6) 式の右辺に $|a_0|$ をかけたことにより、この上限値は $G(X)$ のみならず $\tilde{G}(X)$ に対しても正しい。)

理論上、この上限はどの係数 $|b_i|$ でも使えるわけだが、我々は特に第 2 係数、すなわち $|b_1|$ の上限 B_1' を用いる事にした。その理由は、B_i' の中で B_1' が 1 番小さくなることと、次章で述べるように、因子候補の係数計算が手軽にできるためである。

3. 第3ステップの改良(第1のアイデア)

Step3においては、$\{1, F_1^{(k)}, \cdots, F_r^{(k)}\}$ の積(これを $G(X)$ とする)を計算するのだが、その前に $G(X)$ の第2係数だけを計算して係数上限 B_1' と比較することを考える。G 全体を求めるには要素の積を計算する必要があるが、その第2係数だけならはるかに簡単に計算できる。例えば、要素の積が $F_1^{(k)} \cdots F_m^{(k)}$ の場合、第2係数 $\equiv a_0 \times \sum_{i=1}^{m}[F_i^{(k)}$ の第2係数 $]$ $(\mathrm{mod}\ p^{k+1})$ となる。

従って、このチェックにより $G(X)$ が因子でないと判定できるならば(以下説明するように、大部分の場合に因子でないと判定されるはず)、Step3が実質上、大幅に効率化できることになる。これが、第1のアイデアである。

Step3' [**門前払いありの試し割り**]

1. [第2係数計算] $\{1, F_1^{(k)}, \cdots, F_r^{(k)}\}$ の2個以上の要素の積 $G(X)$ について、その第2係数 b_1 だけを計算する。
2. [第2係数チェック]
   $|b_1| > B_1'$ ならば、$G(X)$ はもはや因子とはなりえないので破棄する(門前払い)。
   $|b_1| \leq B_1'$ ならば、次のステップに進む。
3. [積計算] $G(X)$ の全体を実際に計算する。
4. [試し割り] $G(X)$ で $F(X)$ を試し割りする。もし割り切れれば、$G(X)$ は $F(X)$ の $\mathbf{Z}$ 上の因子である。割り切れなければ、$G(X)$ を破棄する。
5. 割り切る $G(X)$ が一つもなければ、$F(X)$ は $\mathbf{Z}$ 上既約である。

今、p^{k+1} は B_1' より十分大きいとしよう: $B_1'/p^{k+1} = \eta \leq 1$ 。要素の積 $G(X)$ の第2係数 b_1 は、$G(X)$ が $F(X)$ の $\mathbf{Z}$ 上の因子でないとき、統計的には $[-p^{k+1}/2, p^{k+1}/2]$ の区間に一様に分布すると考えてよいだろう。そうすると、$|b_1| \leq B_1'$ となる確率は η となるから、大抵の場合 $|b_1| > B_1'$ となり、上記ステップ2. での門前払いが高い確率で適用されることになるだろう。

4. 実験の結果とその考察

我々は、門前払いの効果を約100例ほどのランダムに生成した既約多項式(20次、係数2、4、8、10桁に対し、それぞれ20例以上)に対して実験を行った。その結果、門前払いの効果が絶大であることが確認できたので、そのうちの3例をここに紹介しよう。なお、本章の実験には数式処理システム GAL を用いている。

実際に使われている因数分解プログラムでは、1章に述べたものに少し改良が加わっており、法が $2B$ (B : Landau の上限)に達する前に一旦Hensel構成をストップさせ、試験的にStep3(積計算&試し割り)を起動して早めに因子を割り出すようにしている。それでも因子が割り出せない場合に限り、法が $2B$ に達するまでHensel構成をするのである。これに対して、門前払いありの因数分解プログラムでは、法を一気に $2B$ まで上げることにした。

下記の3例のうち最初の2例は8桁および2桁の係数をランダムに生成した20次の多項式であり、最後の1例は Swnnerton-Dyer Polynomial と呼ばれる多項式である。

```
例1 F := 1767274722 x^20 + 1675546029 x^19 + 1647418052 x^18
      + 1388290519 x^17 + 1644289366 x^16 + 1149758321 x^15
      + 2111915288 x^14 + 790425851 x^13 + 595337866 x^12 + 836760821 x^11
      + 180171308 x^10 + 267834847 x^9 + 1062517886 x^8 + 486256185 x^7
      + 1508029952 x^6 + 368800899 x^5 + 2035015474 x^4 + 1147902781 x^3
      + 662824084 x^2 + 377401575 x + 1103527590

(1) ### Current GAL version ###  (現行アルゴリズム)

 *** Zp-Factorization, done ..... 110 milli-seconds ***    ステップ1
     Modulus p selected ......... 11   (最初に選んだ法)
     Number of Zp-factors ........ 8    (Zp上の因子の数)
 *** Hensel construction, done ... 460 milli-seconds ***    ステップ2(その1)
     Modulus p^(k+1) lifted ..... 61159090448414546291     (一旦Stopさせた法)
 *** Combining factors, done ..... 1790 milli-seconds ***   ステップ3(その1)
 *** Hensel construction, done ... 120 milli-seconds ***    ステップ2(その2)
     Modulus p^(k+1) lifted ..... 814027493868397611133321  (上限まで上げた法)
 *** Combining factors, done ..... 1500 milli-seconds ***   ステップ3(その2)

 === Spent 3990 + 820 milli-seconds ======(既約と判断されるまでの時間)========

(2) ### 1st Version of Preliminary Testing ### (門前払いありのアルゴリズム)

 *** Zp-Factorization, done ..... 100 milli-seconds ***    ステップ1
     Modulus p selected ......... 11   (最初に選んだ法)
     Number of Zp-factors ........ 8    (Zp上の因子の数)
 *** Hensel construction, done ... 550 milli-seconds ***    ステップ2
     Modulus p^(k+1) lifted ..... 814027493868397611133321(上限まで上げた法)
 *** Combining factors, done ..... 10 milli-seconds ***     ステップ3'
 *** Bound for all the roots ..... 2.1950124458352  (根の上限)
 *** Number of combi's checked ... 162 (上限Checkを受けた第2係数の数)
 *** Number of products formed ... 0   (門前払いをくぐり抜けた積の数)

 === Spent 680 + 380 milli-seconds =======(既約と判断されるまでの時間)========
```

第1の例では、既約と判断するまでの時間を約4秒から0.7秒へと劇的に減少させることができた。これは、この多項式の $\mathbf{Z}_p$ 上の因子が8個と多いために従来のアルゴリズムでは因子組み合わせに多量の時間がかかっていたのが、門前払いありのアルゴリズムではほとんど無視できる時間まで短縮できたためである。実際、従来は162個の組み合わせについてそれぞれ積を計算して試し割りしたのが、我々のアルゴリズムではこの162個全てが門前払いではじかれたのである。

他の20次の多項式をみると、$\mathbf{Z}_p$ 上の因子が5個のときに全計算時間のおよそ30から40％、6個以上のときには60％以上削ることができ、期待通りの成果が得られた。

```
例2 F := 48 x^20 + 43 x^19 + 98 x^18 + 73 x^17 + 36 x^16 + 63 x^15 + 42 x^14
       + 9 x^13 + 20 x^12 + 51 x^11 + 18 x^10 + 97 x^9 + 92 x^8 + 27 x^7
       + 50 x^6 + 81 x^5 + 92 x^4 + 15 x^3 + 50 x^2 + 53 x + 36

 (1) ### Current GAL version ###

  *** Zp-Factorization, done ..... 60 milli-seconds ***
      Modulus p selected ......... 11
      Number of Zp-factors ........ 3
  *** Hensel construction, done ... 20 milli-seconds ***
      Modulus p^(k+1) lifted ...... 14641
  *** Combining factors, done ..... 0 milli-seconds ***
  *** Hensel construction, done ... 50 milli-seconds ***
      Modulus p^(k+1) lifted ...... 214358881
  *** Combining factors, done ..... 10 milli-seconds ***

  === Spent 150 + 0 milli-seconds ==============================================

 (2) ### 1st Version of Preliminary Testing ###

  *** Zp-Factorization, done ..... 60 milli-seconds ***
      Modulus p selected ......... 11
      Number of Zp-factors ........ 3
  *** Hensel construction, done ... 60 milli-seconds ***
      Modulus p^(k+1) lifted ...... 214358881
  *** Combining factors, done ..... 0 milli-seconds ***
  *** Bound for all the roots ..... 3.0416666666667
  *** Number of combi's checked ... 3
  *** Number of products formed ... 0

  === Spent 130 + 0 milli-seconds ==============================================
```

逆に、$\mathbf{Z}_p$ 上の因子が3個しかないのが第2例である。見ての通り、たいして速くならなかった。他の例も大方この程度であった。これまた予想通りといえる。

実験で使用した既約20次多項式について、驚いたことに門前払いをくぐり抜けた積(因子候補)が1個もないことに気付いた。20次ぐらいの多項式では、(4)式の 2^q 因子のため $B_1' \ll B$ となり、100％に近い確率で門前払いに引っかかってしまうため、と推察される。

```
例3 F := x^8 - 40 x^6 + 352 x^4 - 960 x^2 + 576  (Swinnerton-Dyer Polynomial)

 (1) ### Current GAL version ###

  *** Zp-Factorization, done ..... 20 milli-seconds ***
      Modulus p selected ......... 11
      Number of Zp-factors ........ 4
  *** Hensel construction, done ... 10 milli-seconds ***
      Modulus p^(k+1) lifted ...... 14641
```

```
*** Combining factors, done ..... 10 milli-seconds ***
*** Hensel construction, done ... 0 milli-seconds ***
    Modulus p^(k+1) lifted ...... 161051
*** Combining factors, done ..... 10 milli-seconds ***

=== Spent 50 + 0 milli-seconds ================================================

(2) ### 1st Version of Preliminary Testing ###

*** Zp-Factorization, done ...... 10 milli-seconds ***
    Modulus p selected .......... 11
    Number of Zp-factors ........ 4
*** Hensel construction, done ... 20 milli-seconds ***
    Modulus p^(k+1) lifted ...... 161051
*** Combining factors, done ..... 0 milli-seconds ***
*** Bound for all the roots ..... 12.649110640674
*** Number of combi's checked ... 10
*** Number of products formed ... 2

=== Spent 30 + 0 milli-seconds ================================================
```

門前払いされない因子候補があるのが第3例であり、式の真ん中あたりの係数がふくらんでいるタイプの多項式である。門前払いをくぐり抜け2個の積が実際に計算されて、試し割りされている。その結果、既約と判断されたが、この例でも8個の候補が門前払いすることができ、そのぶんの無駄な積計算を避けられた。

5. Hensel構成の早期停止 (第2のアイデア)

因子の第2係数の上限 B_1' には別の利用法がある。
因数分解を、1章で述べた並列Hensel構成ではなく、次のように実行することを考える。

[並列でないHensel構成を用いた因数分解アルゴリズム]

Step1 [$\mathbf{Z}_p$ 上での因数分解]$\cdots$ 1章と同じ

Step2 下記を $\{F_1^{(0)}, \cdots, F_r^{(0)}\}$ のあらゆる組み合わせについて実行する。

2.1 $\{F_1^{(0)}, \cdots, F_r^{(0)}\}$ を2つの組に分け、それぞれの積を $G^{(0)}$ と $H^{(0)}$ とする。

$$F \equiv G^{(0)} H^{(0)} \pmod{p}$$

2.2 [Hensel構成] 法が $2B$ を超えるまでHensel構成をする。

$$F \equiv G^{(k)} H^{(k)} \pmod{p^{k+1}}$$

2.3 [試し割り] $\mathbf{Z}$ 上で $G^{(k)}$ で F を試し割りする。もし割り切れば、$G^{(k)}$ は F の $\mathbf{Z}$ 上の因子である。割り切れなければ、$G^{(k)}$ を破棄する。

Step3 割り切る $G^{(k)}$ が一つもなければ、F は $\mathbf{Z}$ 上既約である。

このアルゴリズムを用いる場合、ステップ 2.2 の Hensel 構成を $2B$ まで行うのではなく、法が $2B_1'$ より十分に大きくなった時点で Hensel 構成を一旦停止し、$G^{(k)}$ と $H^{(k)}$ の第 2 係数が B_1' 以下であるか否かをチェックする。もしも上限 B_1' を超えていれば、その組み合わせは因子とは成り得ないから、その時点で棄却する。こうすることにより、余分な Hensel 構成を行わなくてすむ。これが、第 2 のアイデアである。

こちらのアイデアはまだデータをとれていない。

6. まとめと今後の課題

本稿では、整係数 1 変数多項式の因数分解を効率化するために、因子多項式の第 2 係数の上限を利用する 2 つの方法を提案した。

多数の実例により、第 1 の方法が簡単でありながら極めて有効なものであることを確認した。その方法とは、因数分解を (1) $\mathbf{Z}_p$ 上での因数分解、(2) $p \to p^{k+1}$ への Hensel 構成、(3) $\mathbf{Z}_{p^{k+1}}$ 上での因子の組み合わせ、の 3 ステップに分けるとき、その第 3 ステップを効率化するものである。

よく知られているように、第 3 ステップは計算量の観点からいうと理論上は最もやっかいなステップで、それを改善するために、格子算法が考案されたほどである。しかし、格子算法はかなり複雑なアルゴリズムであり、実際にインプリメントしてみるとそれほど速くはない。それに対して、我々の第 1 の方法は $\mathbf{Z}_p$ 上での因子数が少ない場合は大した改善とはならないが、因子数が多く出て従来のアルゴリズムが非常に遅くなってしまう場合に絶大なる効果を発揮するのである。

我々の考案した第 2 の方法は、因数分解における極めて有効な方法である並列 Hensel 構成と共存できないため、因数分解においてはそれほど有用ではないであろう。しかしながら、初めから 2 つの因子の Hensel 構成を行う演算、例えば GCD 計算や無平方分解においては有効性を期待できる。今後、この点を明らかにしていくつもりである。

さらに、第 1 の方法は、実は多変数多項式の因数分解にも拡張できるのである。多変数多項式において実際にどの程度効率が改善されるかも今後の課題としたい。

7. 付録：定理と証明

Theorem 1 (代数方程式の根の上限) (証明は参考文献 3 参照)
$F(X)$ の任意の根を z とすると次が成り立つ。

$$|z| \leq 1 + \frac{\max\{|a_1|, \cdots, |a_d|\}}{|a_0|} \ (= \tilde{z_1}) \quad \square$$

Theorem 2 (根のもう一つの上限) (証明は参考文献 5 参照)
$F(X)$ の任意の根を z とすると次が成り立つ。

$$|z| \leq \max\{\sqrt[j]{m|a_j|/|a_0|} \mid j = 1, 2, \cdots, d\} \ (= \tilde{z_2}) \quad \square$$

ただし、m は $a_1, a_2, \cdots, a_d$ のなかで 0 でないものの個数である。

Theorem 3 (各係数ごとの上限 B_i')
$\tilde{z}$ を根の上限とする。G の任意の係数 b_i について次が成り立つ。

$$|b_i| \leq |a_0| \binom{q}{i} \tilde{z}^i$$

Proof. 因子 G の根を $z_1, z_2, \cdots, z_q$ とすると、根と係数の関係より次式が成立する。

$$\frac{|b_i|}{|b_0|} \leq |z_1 \cdots z_{i+1}| + \cdots + |z_{q-i} \cdots z_q|$$

$|z_1|, |z_2|, \cdots, |z_q|$ はそれぞれ $\tilde{z}$ で抑えられ、項は全部で $\binom{q}{i}$ 個あるから、

$$\frac{|b_i|}{|b_0|} \leq \tilde{z}^i + \cdots + \tilde{z}^i = \binom{q}{i} \tilde{z}^i$$

$|a_0| \geq |b_0|$ より、

$$|b_i| \leq |a_0| \binom{q}{i} \tilde{z}^i$$

(証明終わり)
上記の 3 つの定理より、(6) 式を得る。

参 考 文 献

[1] Mignott, M : Some Useful Bounds. *Computer Algebra: Symbolic and Algebraic Computation*, Springer-Verlag, Wien New York, 1982, pp. 259-263.

[2] Mignott, M : Some Inequalities about Univariate Polynomials. *Proc. of 1981 Symp. on Symbolic and Algebraic Computation*, 1981, pp. 195-199.

[3] 高木貞治、代数学講義 (改訂新版)、第 3 章、共立出版 (東京)、1981。

[4] 佐々木建昭 他、岩波講座応用数学「計算代数と計算幾何」、第 4 章、岩波書店 (東京)、1993。

[5] 伊理正夫、理工系基礎の数学 12 「数値計算」、第 3 章、朝倉書店 (東京)、1981。

[6] Zassenhaus, H : On Hensel Factorization, Part I. *J. Number Theory 1*, 1969, pp. 291-311.

[7] Wang, P.S. : Parallel p-adic Construction in the Univariate Polynomial Factoring Algorithm. *Proc. of the 1979 MACSYMA Users' Conference*, 1979, pp. 310-318.

[8] Collins, G.E. and Encarnacion, M.J. : Improved Techniques for Factoring Univariate Polynomials. *J. Symbolic Compution*, 1996, pp. 313-327.