数理解析研究所講究録 1335 巻 2003 年 135-140

有限体上の多変数多項式の因数分解について (その 2)

野呂 正行

MASAYUKI NORO

神戸大理学部

FACULTY OF SCIENCE, KOBE UNIVERSITY*

1 はじめに

本稿では, [2] で述べた, 有限体上での 2 変数多項式の因数分解を基礎として, 一般の多変数多項式の GCD, 無平方分解, 因数分解アルゴリズムおよびその実装について述べる.

2 多変数多項式の無平方分解と GCD

現在 Risa/Asir で用いているアルゴリズムは, 以下に述べるように, Bernardin の無平方分解アルゴリズム [1] を modify したものである.

F を標数 p の有限体とし, $f \in F[x_1, \ldots, x_n]$ とする. $'$ が d/dx_1 を表すとする.

$$f = FGH, F = \prod f_i^{a_i}, G = \prod g_j^{b_j}, H = \prod h_k^{c_k}$$

(f_i, g_j, h_k は無平方, 互いに素で, $f_i' \neq 0$, $p \not| a_j$, $p|b_j$, $h_k' = 0$) と書くと $f' = F'GH$ が成り立つ. すると

$$\mathrm{GCD}(f, f') = \mathrm{GCD}(F, F')GH$$

で, $\mathrm{GCD}(F, F') = \prod f_i^{a_i - 1}$ だから

$$f/\mathrm{GCD}(f, f') = \prod f_i.$$

$\prod f_i$ で f を繰り返し割り, 割り切れなくなった段階で, その商と $\prod f_i$ の GCD を計算することで, F 中の重複度最小の因子 f_1 が求まる. これを繰り返すと F が全て無平方分解できる. 残りの f は $f = GH$ と書ける. ここで $f' = 0$ が成り立つことに注意する. 以上の手続きを各 x_i について繰り返して残った f は,

$$df/dx_1 = \ldots = df/dx_n = 0$$

を満たす. これは, 全ての指数が p で割り切れることを意味する. すると, F は標数 p の有限体だから, $f = g^p$ と書けることになる. この g に対して, 以上の手続きを再帰的に適用することで, f の無平方分解が得られる.

このアルゴリズムにおいて, $g = \mathrm{GCD}(f, f')$ の計算が必要となる. 標数が 0 の場合, $\mathrm{GCD}(g, f'/g) = 1$ が保証されるため, この計算に多変数の Hensel 構成を用いることができる. しかし, 正標数の場合には, $\mathrm{GCD}(g, f'/g) = \mathrm{GCD}(\mathrm{GCD}(F, F')GH, F'/\mathrm{GCD}(F, F'))$ となり, 因子 GH の存在のため, この GCD が 1 とは限らない. このため, やむなく Brown のアルゴリズム (中国剰余定理による GCD の計算) を用いている.

*noro@math.kobe-u.ac.jp

GCD の計算
入力：$f_1,\ldots,f_m \in K[X]$ (K は体, X は変数の集合)
出力：$\mathrm{GCD}(f_1,\ldots,f_m)$
$y \leftarrow$ 適当な変数; $Z \leftarrow X \setminus \{y\}$
$<\leftarrow K[Z]$ の適当な項順序; 以下 $f_i \in K[y][Z]$ とみなす
$h_i(y) \leftarrow \mathrm{HT}_<(f_i)$; $h_g(y) \leftarrow \mathrm{GCD}(h_1,\ldots,h_m)$
$g \leftarrow 0$; $M \leftarrow 1$
do
  $a \leftarrow$ 未使用の K の元
  $g_a \leftarrow \mathrm{GCD}(f_1|_{y=a},\ldots,f_m|_{y=a})$
  if $g \neq 0$ かつ $\mathrm{HT}_<(g) = \mathrm{HT}_<(g_a)$ then
    $adj \leftarrow \cdot h_g(a)/\mathrm{HC}_<(g_a)\cdot g_a - g(a))$
    if $adj = 0$ かつ, すべての f_i に対し $g|hg\cdot f_i$ then
      return $\mathrm{pp}(g)$
    endif
    $g \leftarrow g + adj\cdot M(a)^{-1}\cdot M$; $M \leftarrow M\cdot(y-a)$
  else if $\mathrm{tdeg}(\mathrm{HT}_<(g)) > \mathrm{tdeg}(\mathrm{HT}_<(g_a)$ then
    $g \leftarrow g_a$; $M \leftarrow y-a$
  else if $\mathrm{tdeg}(\mathrm{HT}_<(g)) = \mathrm{tdeg}(\mathrm{HT}_<(g_a))$ then
    $g \leftarrow 0$; $M \leftarrow 1$
  endif
end do

HT は頭項, HC は頭係数, tdeg は全次数, pp は原始的部分を表す.

このアルゴリズムでは, $f_1,\ldots,f_m$ に対し, ある変数 y にさまざまな値 a を代入して GCD g_a を計算し, それらを中国剰余定理で結合する. その際, 残りの変数 Z に関して適当な項順序 $<$ を設定し, その項順序に関する頭項が等しい間は, 真の GCD の正しい射影になっていると仮定する. それまでと異なる頭項を持つ g_a が得られた場合, その頭項の全次数が, それまでの頭項の全次数より真に大きい場合には, 明らかに正しくないので, 捨てる. また, 真に小さい場合には, これまでの結果は正しくないことになり, 新たに g_a からリスタートする. もし, 全次数が等しければ, 両方の結果が正しくないことになり, 両方を捨ててリスタートする.

また, 真の GCD の $<$ に関する頭係数が f_i の頭係数の GCD である h_g の因子になっていることを用い, 中国剰余定理を適用する際, 頭係数が h_g になるように調節している.

新たな a で得た g_a と, これまで得た g の a での値が等しい場合に, 実際に g が $h_g f_i$ をすべて割り切るかチェックを行っている. 実際の実装においては, 適当な f_i を選んでおき, cofactor のほうも中国剰余定理で構成していき, そちらでも割算チェックを行うことで, GCD, cofactor いずれかが復元できた時点でアルゴリズムが終了できる.

このアルゴリズムでは, GCD の y に関する次数より多い K の元が必要になる. K の位数がこれに満たない場合には, K を拡大する. これについては後で述べる.

3 無平方多項式の因数分解

以下, $f \in K[X]$ は無平方とする. f の因数分解は次のように行う.

3.1 主変数 x の選択

f の因数分解は, ある変数 x 以外の変数に適当な値を代入して得られた x の一変数多項式の因数分解をタネから順次 Hensel 構成により計算する (EZ 法). x は, x に関する微分が消えないように選ぶ必要がある. また, x に関する次数が大きい程, タネとなる因子の数が大きくなる可能性があるため, なるべく次数が小さいような x を選んでいる.

3.2 従変数 y の選択

従変数とは妙な用語であるが, ここでは, 多変数の因数分解を, 2 変数の因数分解から Hensel 構成により得るため, そのための変数をもう一つ選んでおく必要がある. すなわち, ある変数 y を選び, x, y 以外の変数に適当な値を代入した 2 変数多項式の因数分解を [2] で述べた方法により計算する. それを基に, 残りの変数に関して Hensel 構成を行う. ここで, Hensel 構成は $K[X] = K[y][x, Z]$ $(Z = X \setminus \{x, y\} = \{z_1, \ldots, z_{n-2}\})$ とみなして行う. すなわち, 一変数多項式環 $K[y]$ を, 有理数体上の多変数多項式の因数分解における, 整数環の類似とみなすわけである. こうすることにより, 1 変数の分解から出発した場合に生ずる大量のニセ因子による困難を避けることができる. あとで示すように, Hensel 構成自体も, 整数上の場合の類似の方法により行うことができる.

3.3 2 変数の因数分解

x, y が決まったら, $f_a(x, y) = f(x, y, a)$ が, 無平方になるように Z に代入する値のベクトル $a = (a_1, \ldots, a_{n-2}) \in K^{n-2}$ を選び, f_a を因数分解する. ここで, f_a の x に関する主係数 (これは y の多項式) の定数項が 0 でなく, かつ $f_a|_{y=0}$ が無平方であるよう, 必要があれば $y \to y + c$ という平行移動を行う.

3.4 $K[y]$ 上での Hensel 構成 (前処理)

$f_a(x, y)$ の因数分解の結果より, 因子を 2 組にわけ $f_a(x, y) = g_0(x, y)h_0(x, y)$ とした上で, $K[y]$ 上で Hensel 構成を行う. この際, 問題となるのが g_0, h_0 の x に関する主係数の決め方である. いわゆる主係数問題を回避するために, 真の因子の主係数となるべく近いものをあらかじめ固定しておくのがよい. 少なくとも, それは, f の, x に関する主係数 $\mathrm{lc}_x(f)$ の因子ではあるが, $\mathrm{lc}_x(f)$ そのものをとることは一般に overestimate である. 有理数体上の場合, P. S. Wang により主係数の決定方法が提案されているが, ここでは次のように見積もる:

1. $\mathrm{lc}_x(f) = \prod u_i^{n_i}$ と因数分解する ($u_i \in K[y, Z]$: 既約).
2. 各 i に対し, $u_i(a) \in K[y]$ が $\mathrm{lc}_x(g_0)$ を割り切る回数を数える. それを m_i としたとき, $\mathrm{lc}_g = \prod u_i^{m_i}$ とする. 同様に h_0 に対しても lc_h を求める.

   もし, $\mathrm{lc}_x(g_0) \not| \mathrm{lc}_g(a)$ または $\mathrm{lc}_x(h_0) \not| \mathrm{lc}_h(a)$ または, $\mathrm{lc}_x(f) \not| \mathrm{lc}_g \cdot \mathrm{lc}_h$ ならば, それは, f_a の因子の組合せが正しくないことを意味するので, g_0, h_0 をとり直す.
3. $g_0 \leftarrow \mathrm{lc}_g(a)/\mathrm{lc}_x(g_0) \cdot g_0$ の主係数を lc_g で置き換えたもの

   $h_0 \leftarrow \mathrm{lc}_h(a)/\mathrm{lc}_x(h_0) \cdot h_0$ の主係数を lc_h で置き換えたもの

   $f \leftarrow \mathrm{lc}_g \cdot \mathrm{lc}_h/\mathrm{lc}_x(f) \cdot f$

   とする. この時, $f = g_0 h_0$ となっている.

この段階で, もし g_0 が真の因子の射影となっていれば, Hensel 構成により, f の因子に持ち上るはずであり, $\mathrm{lc}_x(g_0)$ は既に, 真の因子の主係数に等しくなっている. この場合, h_0 も同様の性質を満たす.

3.5 $K[y]$ 上での Hensel 構成

前項により, $f = g_0h_0$ において, g_0 が正しい因子の射影ならば, $\mathrm{lc}_x(g_0)$ は既に真の因子の主係数に等しい. ここでは, g_0, h_0 から $K[y]$ 上の Hensel 構成により, $f = g_kh_k \bmod I^{k+1}$, ただし $I =< z_1 - a_1, \ldots, z_{n-2} - a_{n-2} >$, となる g_k, h_k を EZ 法により計算する. まず, $z_i \to z_i + a_i$ なる平行移動により, $I =< z_1, \ldots, z_{n-2} >$ としておく. 通常の EZ 法では, 係数に分数が現れるのを避けるため, 因子の係数の大きさの評価から定められる, ある大きな素数巾 p^l を法として $\mathbf{Z}/(p^l)$ 上で計算する. ここでは, f の y に関する次数を越える整数 d に対し, $K[y]/(y^d)$ での演算を導入することで, $K[y]$ での商体での演算を避ける. すなわち, $ug_0(a) + vh_0(a) = 1 \bmod y^d$ となる $u, v \in K[y]$ を計算しておき, Hensel 構成の係数の計算は u, v を用いて $\bmod y^d$ で行うのである. このとき, $f = g_kh_k \bmod (I^{k+1}, y^d)$ だが, d が十分大きいため, g_0 が真の因子の射影ならば, Hensel 構成の一意性により, 十分大きい k に対し $f = g_kh_k$ となる. u, v の計算は, $g_0(a)|_{y=0}$, $h_0(a)_{y=0}$ が互いに素であることを利用して, やはり Hensel 構成により計算できる.

$K[y]$ 上の Hensel 構成は次のように行う.

1. $f - g_{k-1}h_{k-1} = \sum_t F_t t \bmod (I^k, y^d)$ と書く. ここで $t \in I^k$ は単項式, $F_t \in K[y][x]$.
2. $G_th_0 + H_tg_0 = F_t \bmod y^d$ となる $G_t, H_t \in K[y][x]$ を計算する. これは u, v を使って作れる.
3. $g_{k+1} \leftarrow g_k + \sum_t G_t t$, $h_{k+1} \leftarrow h_k + \sum_t H_t t$ とすれば $f = g_{k+1}h_{k+1} \bmod (I^{k+1}, y^d)$.

この操作において, $\sum_t G_t t$ または $\sum_t H_t t$ が 0 となった場合に g_k または h_k で f を割ってみることで, 次数の上限まで Hensel 構成せずに, 真の因子を検出することができる.

4 実装について

4.1 有限体の表現について

ここで述べた各アルゴリズムにおいては, 係数体の位数が十分大きい必要がある. [2] で述べたように, 代入する点の数が不足する場合のために, 代数拡大しても計算効率が落ちないような, 原始根を用いた表現を実装し, その有効性を示した. 欠点としては, この表現において, 実用的な位数が 2^{16} 程度に限られることであった. 今回, その改良として, 標数が 2^{14} 以下の場合には, 原始根表現を許し, それ以上の場合には, 通常の表現をとることにした. これは, 後者では実用上十分に代入する点が得られるからである. これにより, 位数が 2^{29} 程度までの素体上で, 多変数多項式の因数分解が行えるようになった.

4.2 係数環としての $R[y]/(y^d)$ について

Hensel 構成においては, $R[y]/(y^d)$ を, 多項式の係数環として扱う必要がある. このため, 新たな数の型として, $R[y]/(y^d)$ を表す型を定義した. 多変数多項式は, Hensel 構成の最初で, この型の係数を持つ多項式に変換される. あらかじめ d をセットしておくことにより, 演算は自動的に $\bmod y^d$ されるため, 通常の多項式演算の呼び出しを行うだけで, $K[y]/(y^d)$ 係数の多項式演算が実行できる.

4.3 タイミングデータ

OpenXM_contrib2/asir2000/lib/fctrdata の, 因数分解テスト用多変数多項式 Wang[1], ..., Wang[15] を用いて計算時間を計測した. マシン は Athlon 1900+ で, Maple7 と比較を行った. Maple における実装は, Bernardin によるものであり, やはり Kernel で諸演算をサポートしているので, アンフェアではないだろうと考える. Maple, Asir のタイミングデータをそれぞれ表 1, 表 2 に示した. 単位はミリ秒である. 表で, N は 1 分以上待っても結果が得られないもの, F は何らかの内部エラーにより計算が完了しなかったものを示す. 実験で用いた多項式に対しては, Asir 上での今回の実装は全ての場合に正しい結果を返し, かつ計算時間についても一部 (Wang[8]) を除いて Asir 上での今回の実装が良好な結果を示した.

p	1	2	3	4	5	6	7	8
2	N	F	F	F	N	N	10	1000
3	70	100	70	N	400	N	10	20
5	N	50	80	3500	200	400	10	600
7	80	100	100	250	600	500	20	1000

	9	10	11	12	13	14	15
2	10	F	N	5	6	8	F
3	60	4000	N	4	7	140	20
5	100	200	F	5	6	30	400
7	F	600	14000	5	160	40	600

p	1	2	3	4	5	6	7	8
547	200	200	100	300	1000	1200	20	6000
32003	200	200	200	400	1000	1000	20	4200
99981793	500	600	500	3000	3000	4500	20	N

p	9	10	11	12	13	14	15
547	F	900	3300	5	200	100	400
32003	F	1800	4900	6	300	100	400
99981793	F	2600	11000	6	900	500	1400

表 1: Timing data (Maple)

5 おわりに

今回の実装は, Asir (Version 20030307 以降) 上で, sffctr(Poly) として利用できる. 例えば, 多項式 $x^3+y^3+z^3+xyz$ を $GF(2^{10})$ 上で因数分解するには次のようにすればよい.

```
[0] setmod_ff(2,10);
[2,x^10+x^3+1,x]
[1] sffctr(x^3+y^3+z^3+x*y*z);
[[@_0,1],[@_0*x+@_341*y+@_682*z,1],[@_0*x+@_0*y+@_0*z,1],
[@_0*x+@_682*y+@_341*z,1]]
```

p	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2(5)	3	3	4	10	20	50	1	10	0.3	30	70	0.6	1	2	1
3(5)	3	2	5	3	3	100	2	1	3	40	200	0.1	0.5	20	1
5(2)	4	3	4	20	60	400	2	400	5	10	200	1	1	4	10
7(2)	4	4	5	30	100	100	4	1800	200	20	600	1	7	5	10

p	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
547	4	4	5	30	50	200	20	2000	200	40	300	1	6	6	10
32003	4	4	5	40	70	200	4	2000	200	40	200	1	7	6	30
99981793	4	4	5	30	30	200	4	4000	200	40	300	1	8	8	10

表 2: Timing data (Asir)

この関数を用いて, 小位数有限体上でのイデアルの極小素因子計算 (radical の素イデアル分解) を試験的に実装した. これはまもなく公開予定である. 最終的には準素分解の実装を予定しているが, 標数に関係する部分はほぼ実装が完了したと考えられる. また, これまで有限素体上での一変数多項式の因数分解のみを対象としていた `modfctr(Poly,Mod)` が, `sffctr()` を内部的に用いることで, 多変数多項式も因数分解できるようになった.

`sffctr()` 自体, 改良すべき点はまだ多くある. その主なものをあげておく.

- 基礎体の自動的拡大

  体の位数が足りない場合は, 現状ではユーザが体を設定しなおす必要があるが, これを, 自動的に基礎体を拡大するようにする.

- 無平方分解の改良

  体の標数が十分大きい場合には, 無平方分解において, 微分が消えるような場合は起こらないので, 無平方分解を標数 0 と同様の Hensel 構成で行うようにする.

- hard-to-factor 多項式への対応

  素イデアル分解において, ニセ因子を多く含む多項式が実際に現れた. このような場合にも効率よい因数分解を行うために, 2 変数の因数分解において, [2] で述べた, 多項式時間アルゴリズムを自動的に選択して実行するようにする.

参 考 文 献

[1] Bernardin, L. (1997). On square-free factorization of multivariate polynomials over a finite field. *Theoret. Comput. Sci.* **187**, 105–116.

[2] M. Noro and K. Yokoyama (2002). Yet Another Practical Implementation of Polynomial Factorization over Finite Fields. Proceedings of ISSAC2002, ACM Press, 200–206.