数理解析研究所講究録
第1568巻 2007年 149-155



# 多変数の解析的因数分解の実装について*

岩見 真希
MAKI IWAMI
大阪経済法科大学　教養部
FACULTY OF LIBERAL ARTS AND SCIENCES, OSAKA UNIVERSITY OF ECONOMICS AND LAW†

## 1　はじめに

これまで，多変数の解析的因数分解について取り組んできたが，アルゴリズムは提示していたものの，実際に実験に使用してきたのは逐次計算であった．今回，アルゴリズムとしてきちんと書き下し，Mathematicaで実装を行うことで，これまで曖昧に思われてきた部分を実証し，今後，いくつかのアルゴリズムの比較，応用を模索する土台づくりにしたいと考えている．

解析的因数分解とは，展開点を固定した形式的べき級数環での因数分解のことで，2変数に関しては，Abhyankarの近似根を求めるアイディア [1] をもとにKuoが解析的因数分解のために展開基底法 [5] を提案，後にMcCallumが実装 [6] を行った．近年，Sasakiの拡張Hensel構成を用いた方法 [9] がある．3変数以上（以下，本稿では多変数とよぶ）では，筆者が2変数の各アルゴリズムをそれぞれ [2] および [3] で多変数化してきたものがある．さらに，これらの過程を経た成果である拡張Hensel構成と展開基底法を融合させた解析的因数分解法が [4] である．本稿では，これら [2, 3, 4] の実装について報告する．

例として，次のような$f$を考える．

$$\begin{aligned} f &\stackrel{def}{=} (x^3-y-z+(y+3z)^2)^2-((y+2z)^8+(y+2z)^{10}) \\ &= (x^3-y-z+(y+3z)^2+\sqrt{(y+2z)^8+(y+2z)^{10}})(x^3-y-z+(y+3z)^2-\sqrt{(y+2z)^8+(y+2z)^{10}}) \end{aligned}$$

$f$のニュートン多項式は，$(x^3-(y+z))^2$ で，多項式環では互いに素に分解できない．$f$は，多項式環では既約であるが，解析的因数分解では，原点を展開点としたときの形式的べき級数環での因数分解を求めるのが目的である．2つの因子をそれぞれ原点で展開してみる．

$$\begin{aligned} &x^3-y-z+(y+3z)^2+\sqrt{(y+2z)^8+(y+2z)^{10}} \\ &= x^3-y+y^2+y^4+y^6/2-z+6yz+8y^3z+6y^5z+9z^2+24y^2z^2+30y^4z^2+32yz^3+80y^3z^3+16z^4 \\ &\quad +120y^2z^4+96yz^5+\cdots \\ &x^3-y-z+(y+3z)^2-\sqrt{(y+2z)^8+(y+2z)^{10}} \\ &= x^3-y+y^2-y^4-y^6/2-z+6yz-8y^3z-6y^5z+9z^2-24y^2z^2-30y^4z^2-32yz^3-80y^3z^3-16z^4 \\ &\quad -120y^2z^4-96yz^5-\cdots \end{aligned}$$

この例では原点でのTaylor展開で済むが，いつもそうとは限らない．ここでは，多変数展開基底を用いた解析的因数分解のアルゴリズムおよびそのMathematicaへの実装について報告する．

*本研究の一部は，日本学術振興会 科学研究費補助金の助成を受けて行われた．
†maki@keiho-u.ac.jp

## 2 多変数展開基底アルゴリズム

ニュートン線の異符号傾きと展開基底の重みを対応づけて，多変数展開基底法におけるニュートン多項式とニュートン線の異符号傾きは，"拡張 Hensel 構成 [7, 8] における手法であるアルゴリズム 1 を読み込んで計算するアルゴリズム 2"，あるいは改良して，"ニュートン線の異符号傾きのみを計算するアルゴリズム 3 を読み込んで，ニュートン線を統一的に水平になるように変換（拡張 Hensel 構成 [7] のものを多変数展開基底法に適用し，両者を融合させた変換）してから計算するアルゴリズム 4"により計算できる．

**アルゴリズム 1 (ニュートン多項式とニュートン線の異符号傾き)**

INPUT : 主変数 $x$, 従変数に関する全次数変数 $t$, 多項式 $f$ に $t$ を導入した $tf$
OUTPUT : $tf$ のニュートン多項式を第 1 要素，ニュートン線の異符号傾きを第 2 要素とするリスト

1. $n \leftarrow$ ($tf$ の $x$ の次数)， $d0 \leftarrow$ ($x^n$の係数部の $t$ に関する order)
2. $slist \leftarrow$ ( $\left((x^i$の係数部の $t$ に関する order$) - d0\right)/(n-i)$, $(i = 0, \cdots, n-1)$ のリスト )
3. $slope \leftarrow$ ($slist$ の最小値 (ニュートン線の傾き $\times (-1)$ に相当))
4. ニュートン線上の項をすべて足し合わせた式と，$slope$ のリストを出力する．

**アルゴリズム 2 (多変数展開基底法でのニュートン多項式とニュートン線の異符号傾き)**

INPUT : 主変数 $G[s]$, $G$ 進展開した $tf$, 展開基底の要素である従変数 $G[i]$ とその重み $w[i]$ の対 $\{G[i], w[i]\}$ $(i = -1, 0, \cdots, s-1)$ のリスト．初期値は $G[-1] = t$, $G[0] = x$, $w[-1] = 1$.
OUTPUT : $tf$ の $G[s]$ を主変数とみたときの従変数に関する重みつきニュートン多項式を第 1 要素，ニュートン線の異符号傾きを第 2 要素とするリスト

1. 次で $tf$ を変数変換したものを $ntf$ とおく．
   $G[i] \leftarrow (nt^{w[i]}) * G[i]$ $(i = -1, 0, \cdots, s-1)$
2. アルゴリズム 1 により，$ntf$ の主変数 $G[s]$，全次数変数 $nt$ に関するニュートン多項式と傾き $slope$ を計算する．
3. 2.で計算した結果を，1.の変数変換の逆変換 $(nt \leftarrow 1)$ したものを第 1 要素，傾き $slope$ を第 2 要素として出力する．

**アルゴリズム 3 (ニュートン線の異符号傾きのみ)**

INPUT : 主変数 $x$, 従変数に関する全次数変数 $t$, 多項式 $f$ に $t$ を導入した $tf$
OUTPUT : ニュートン線の異符号傾き

1. $n \leftarrow$ ($tf$ の $x$ の次数)， $d0 \leftarrow$ ($x^n$の係数部の $t$ に関する order)
2. $slist \leftarrow$ ( $\left((x^i$の係数部の $t$ に関する order$) - d0\right)/(n-i)$, $(i = 0, \cdots, n-1)$ のリスト )
3. $slope \leftarrow$ ($slist$ の最小値 (ニュートン線の傾き $\times (-1)$ に相当)) を出力する

**アルゴリズム 4 (多変数展開基底法でのニュートン多項式とニュートン線の異符号傾き (アルゴリズム 3 利用))**

INPUT : 主変数 $G[s]$, $G$ 進展開した $tf$, 展開基底の要素である従変数 $G[i]$ とその重み $w[i]$ の対 $\{G[i], w[i]\}$ $(i = -1, 0, \cdots, s-1)$ のリスト．初期値は $G[-1] = t$, $G[0] = x$, $w[-1] = 1$.
OUTPUT : $tf$ の $G[s]$ を主変数とみたときの従変数に関する重みつきニュートン多項式を第 1 要素，ニュートン線の異符号傾きを第 2 要素とするリスト

1. 次で $tf$ を変数変換したものを $ntf$ とおく.
   $G[i] \leftarrow (nt^{w[i]}) * G[i]$ $(i = -1, 0, \cdots, s-1)$

2. アルゴリズム 3 により, $ntf$ の主変数 $G[s]$ と全次数変数 $nt$ に関するニュートン線の異符号傾き（のみ）を計算し, $ws \leftarrow w[s]$ とおく.

3. $G[s] \leftarrow (nt^{ws}) * G[s]$, $\alpha \leftarrow (ntf$ の $nt$ に関する order), $ntf \leftarrow ntf/t^{\alpha}$

4. $nt \leftarrow 0$ で変換した結果をリストの第 1 要素, 異符号傾き $ws$ を第 2 要素として出力する.

**アルゴリズム 5 ($G$ 進展開の計算)**

INPUT : 全次数変数 $t$ を導入した多変数多項式 $tf$,
展開基底のリスト $\{G[-1](=t), G[0](=x), \cdots, G[s]\}$, 主変数 $G[s]$
OUTPUT : $tf$ の $G$ 進展開 $\sum_{j=0}^{m} c_j G[-1]^{e[j,-1]} \cdots G[s-1]^{e[j,s-1]} G[s]^j$

1. 初期値の設定 :
   $baserow \leftarrow (G[s], G[s-1], \cdots, G[1]$ の順で並べたリスト)
   (* $x$ に関する多項式として表現されている. $G[-1](=t)$, $G[0](=x)$ はここには入れない *)
   $result \leftarrow 0$, (* ここに結果を足しこめていく *)
   $rem \leftarrow tf$

2. $baserow$ の要素数が 1 以上であれば以下を実行する.
   $rem \leftarrow (rem + result)$, $result \leftarrow 0$,
   $pb \leftarrow (baserow$ の 1 番目の要素), $m \leftarrow (tf$ における $pb$ の重複度),
   For [ $k = 0$, $k < m$, $k++$,
   $result \leftarrow \left(result + pb^{(m-k)} * (rem$ を $pb^{(m-k)}$でわった商$)\right)$,
   $rem \leftarrow \left(rem$ を $pb^{(m-k)}$でわった剰余$\right)$,
   $baserow$ から 1 番目の要素を除く ]

3. $result \leftarrow (result + rem)$ (* 最後に剰余の分を足す *)

4. $result$ を出力する.

**アルゴリズム 6 (多変数展開基底で互いに素な初期因子を求める)**

INPUT : 多変数多項式 $f$, 主変数 $x$ (* ニュートン多項式が互いに素な因子に分解できない場合に有効 *)
OUTPUT : $f$ の互いに素な初期因子 (* これを *Lifting* すればよい *)

1. $tf \leftarrow (f$ に従変数に関する全次数変数 $t$ を導入)

2. アルゴリズム 1 により, $tf$ の主変数 $x$, 全次数変数 $t$ に関する計算を行い, 次のようにおく.
   $newpoly \leftarrow$ (ニュートン多項式), $tmp \leftarrow$ (ニュートン線の異符号傾き)
   (* $tmp$ は3.で展開基底の重みとして用いる *)

3. 展開基底の初期設定 : $base \leftarrow \{t, x\}$;
   重みの初期設定 : $weights \leftarrow \{\{t, 1\}, \{x, tmp\}\}$ (* 展開基底と対にしたリストで定義する *)

4. $newpoly$ を展開基底の要素を変数とした多項式環で因数分解する. 互いに素な因子に分解できれば, その分解結果を出力するために 6. へ. 互いに素な因子に分解できず, 重複していなければアルゴリズム 7 にわたす. 重複していれば, 重複既約部分を展開基底の要素として $base$ に追加する.

5. $gadicpoly \leftarrow$ (アルゴリズム 5 により, $tf$ を展開基底 $base$ で $G$ 進展開した結果),
$s \leftarrow$ ($base$ の要素数 $-2$)　　(* 初期値の $t$ と $x$ の数は含めない *)
アルゴリズム 4 により $gadicpoly$ を $G[s]$ を主変数とみて重み $weights$ で計算を行い, 次のようにおく.
$newpoly \leftarrow$ (ニュートン多項式),　$tmp \leftarrow$ (ニュートン線の異符号傾き),
$weights$ に $tmp$ を $G[s]$ の重みとして追加し, $newpoly$ を因数分解し, 4.へ

6. $G$ 進展開で表現されている各因子を, $x$ で表現されている $base$ の各要素を用いて変換し, 全次数変数も $t \leftarrow 1$ に変換して戻してから出力する.

**アルゴリズム 7 (pseudoinitialform への変換)**

INPUT　: $tf$ の $G[s]$ を主変数とするニュートン多項式, ニュートン線の異符号傾き $slope$,
展開基底 $base = \{G[-1], G[0], \cdots, G[s]\}$, ただし $G[-1] = t, G[0] = x, w[-1] = 1$,
$weights = \{\{G[-1], w[-1]\}, \{G[0], w[0]\}, \cdots \{G[s], w[s]\}\}$.

OUTPUT　: $tf$ のニュートン多項式と各項の $weight$ を同じくする $pseudoinitialform$
(の予定であるが, 現時点でのプログラミングでは, $pseudoinitialform$ にするために用いる $monomial\Delta$ の生成まで完了しており, 今後, $G[s]^d$ と $\Delta$ の同次式への変換を行う)

1. ($G[s]$ の $x$ に関する次数) $* w[s] = N/d$　($d$ と $N$ は互いに素な正の整数) から $d$ と $N$ を計算する

2. $d * w[s] = h[-1] * w[-1] + h[0] * w[0] + \cdots + h[s-1] * w[s-1]$,
$h[-1] > 0$, $0 \leq h[i] < d[i+1]$(= ($G[i+1]$ の $x$ の次数)/($G[i]$ の $x$ の次数)), $(0 \leq i \leq s-1)$
なる正の整数 $h[-1], \cdots, h[s-1]$ を計算する

3. $\Delta \leftarrow G[-1]^{h[-1]} * G[0]^{h[0]} * \cdots * G[s-1]^{h[s-1]}$ とおく

4. 入力したニュートン多項式を $G[s]^d$ と $\Delta$ の同次多項式で表現した $pseudoinitialform$ を出力する.

他に, Lifting と最終的に解析的因子を返すアルゴリズムを追加する必要がある.

## 3 Mathematica での実装

- $x$ を主変数とみたときの $f$ のニュートン多項式 (第 1 要素) と, ニュートン線の傾きの異符号 (第 2 要素) を求める関数 newtonpolynomial　(アルゴリズム 1, 2)

```
newtonpolynomial[tf_, x_, t_] := Module[{n, dlist, slope},
    (* tf は f に従変数の全次数変数 t を導入したもの,  n は x の次数 *)
    n = Exponent[tf, x];
    (* x の次数に関して降べきの順で並べ, 係数部 t の order のリストを dlist とおく *)
    dlist = Reverse[Map[Min[Exponent[#, t, List]] &, CoefficientList[tf, x]]];
    (* t 座標の変化量 *)
    dlist = Map[If[NumberQ[#], # - First[dlist], Infinity] &, Rest[dlist]];
    dlist = MapIndexed[(#1/First[#2]) &, dlist];
    slope = Min[dlist];
    (* ニュートン多角形の右下端点から x^i(i=0,…,n-1) の各 order を与える項に対応する点
       までの線分のうち異符号傾き最小のものを slope とおき, そのときの線分がニュートン線 *)
    dlist = Flatten[Position[dlist, Min[dlist]]];
```

```
    (* もとのxの次数になおす *)
    dlist = Prepend[n + 1 - dlist, n + 1];
    (* xとtに関するニュートン多項式と，異符号傾きを返す *)
    Return[{Plus@@Map[If[FreeQ[#, t], #,
            Coefficient[#, t^Exponent[#, t, Min]]*t^Exponent[#, t, Min]] &,
            DeleteCases[(CoefficientList[tf, x]*Table[x^i, {i, 0, n}])[[dlist]], 0]],
            slope}]]

newtonpolynomial[tf_, Gs_, weights_List] := Module[{ntf, nt},
    (* 展開基底に関する重みつき全次数変数ntの導入 *)
    ntf = tf /. MapThread[Rule[#1, nt^(#2)*#1] &, Transpose[weights]];
    Return[newtonpolynomial[ntf, Gs, nt] /. nt -> 1]]
```

- ニュートン線の異符号傾きのみ出力する関数 newtonslope (**アルゴリズム 3**), 展開基底の重みを利用してニュートン線を統一的に水平に変換し，$x$ を主変数とする $f$ のニュートン多項式（第 1 要素），ニュートン線の異符号傾き（第 2 要素）を求める関数 newtonpolynomialhorizontal (**アルゴリズム 4**)

```
newtonslope[tf_, x_, t_] := Module[{n, dlist, slope},
    (* tfはfに従変数の全次数変数tを導入したもの，nはxの次数 *)
    n = Exponent[tf, x];
    (* xの次数に関して降べきの順で並べ，係数部tのorderのリストをdlistとおく *)
    dlist = Reverse[Map[Min[Exponent[#, t, List]] &, CoefficientList[tf, x]]];
    (* t座標の変化量 *)
    dlist = Map[If[NumberQ[#], # - First[dlist], Infinity] &, Rest[dlist]];
    dlist = MapIndexed[(#1/First[#2]) &, dlist];
    slope = Min[dlist]; Return[slope];]

newtonpolynomialhorizontal[tf_, Gs_, weights_List] := Module[{ntf, nt, ws, a, ws},
      (* 展開基底に対する重みつき全次数変数ntの導入 *)
      ntf = tf /. MapThread[Rule[#1, nt^(#2)*#1] &, Transpose[weights]];
      ws = newtonslope[ntf, Gs, nt];
      a = ntf /. {Gs -> Gs *(nt^{ws})};
      Return[{Cancel[(a/nt^Exponent[a, nt, Min])] /. nt -> 0, ws}]];
```

- $f$ を展開基底 $base := \{t, x, G[1], \cdots, G[s]\}$ で $G$ 進展開する関数 gadicpolynomial (**アルゴリズム 5**)

```
gadicpolynomial[tf_, base_List, G_] := Module[{baserow, rem, result, pb, a, x},
    (* G[1], … , G[s] をもとのxとtの表現になおす *)
    x = base[[2]];
    baserow = Reverse[Drop[base, 2]]
              //. MapThread[Rule, {Table[G[i - 2], {i, 1, Length[base]}], base}];
    (* tfを展開基底で表現するための初期値を設定 *)
    result = 0;
    rem = tf;
```

```
(* ループを開始するのでresult, remの初期値を設定 *)
While[Length[baserow] > 0, rem += result; result = 0;
  (* 新しい主変数のxを用いた元の表現をpbとする *)
  pb = First[baserow];
  (* pbの重複度をmとする *)
  m = Exponent[tf, x]/Exponent[pb, x];
  For[k = 0, k < m, k++,
      result += G[Length[baserow]]^(m-k)*PolynomialQuotient[rem, pb^(m-k), x];
      rem = PolynomialRemainder[rem, pb^(m-k), x];];
  baserow = Rest[baserow];];
result = result + rem; Return[result];]
```

- pseudoinitial form に書き換える関数 pseudoinitialform (**アルゴリズム 7**)
  $G[s]^d$ と monomial $\Delta(= G[-1]^{h[-1]} \cdots G[s-1]^{h[s-1]})$ の同次多項式である pseudoinitial form にすることで初期因子をうまく分解できればよい．ここでは $h[-1], \cdots, h[s-1]$ を得るところまで実装した．

```
<< Algebra`InequalitySolve`
(* パッケージを読み込む *)
pseudoinitialform[tf_, slope_, base_, weights_] := Module[{dd, d, s, w, h},
    s = Length[base] - 2;
    (* 展開基底の要素数sを求める *)
    Table[w[i - 2] = weights[[i, 2]], {i, 1, s + 2}];
    (* 展開基底の重みに添字をつけて取り出しておく *)
    dd = Denominator[Cancel[Exponent[Last[base], x]*w[s]]];
    (* (G[s]のxに関する次数)*w[s] = N/d なるdを求める *)
    Table[d[i + 1] = Exponent[base[[i+3]], x]/Exponent[base[[i+2]], x], {i,0, s-1}];
    (* d[i + 1] = (G[i + 1]のxの次数)/(G[i]のxの次数)  の設定 *)
    Reduce[InequalitySolve[Flatten[{h[-1] > 0,
     Flatten[{Table[0 <= h[i], {i,0,s-1}], Table[h[i] < d[i+1], {i,0,s-1}]}]}],
     Table[h[i], {i, -1, s - 1}]] && dd*w[s] == Sum[h[i]*w[i], {i, -1, s - 1}],
     Table[h[i], {i, -1, s - 1}], Integers]]
```

- 初期因子が重複した $f$ を受け取り，多変数展開基底とそれらの Weight，初期因子の因数分解結果を返す関数 multivariateexpansionbase (**アルゴリズム 6**)

```
multivariateexpansionbase[f_, x_] := Module[{newpoly, subvariables, t, tf, result,
    weights, G, base, gadicpoly, tmp},
    subvariables = DeleteCases[Variables[f], x];  (* 従変数を取り出す *)
    tf = f /. Map[Rule[#, t*#] &, subvariables];  (* 全次数変数tの導入 *)
    {newpoly, tmp} = newtonpolynomial[tf, x, t];
    (* ニュートン多項式をnewpoly, その異符号傾きをtmpとおく *)
    weights = {{t, 1}, {x, tmp}};
    (* 展開基底とその重みを対にしたリストでweightを定義し，初期設定を行う *)
    base = {t, x};  (* 展開基底の初期設定を行う *)
```

```
    result = FactorList[newpoly];
(* 以下, newpoly が互いに素な因子の積に分解できればその分解結果を返し, 重複していて互いに
   素な因子の積に分解できなければその重複既約部分を新しい展開基底として追加する *)
    While[result[[2, 2]] >= 2,
     If[Length[result] == 2, AppendTo[base, result[[2, 1]]], Return[Factor[newpoly] //.
       MapThread[Rule, {Table[G[i - 2], {i, 1, Length[base]}], base}] /. t -> 1]];
     gadicpoly = [tf, base, G];
     s = Length[base] - 2;
     {newpoly, tmp} = newtonpolynomial[gadicpoly, G[s], weights];
     AppendTo[weights, {G[s], tmp}];
   (* tf を G 進展開し, そのときのニュートン線の異符号傾きを, 追加された展開基底の新要素
     (新たな主変数) の重みとする *)
     result = FactorList[newpoly];];
   (* もし newpoly が重複していれば, 展開基底 base に追加してくりかえす *)
    result = result //. MapThread[Rule, {Table[G[i - 2], {i, 1, Length[base]}], base}];
    result = result /. t -> 1;
    result = Times @@ Map[(Power @@ #) &, result]; Return[result];]
```

## 参 考 文 献


[1] S. S. Abhyankar: Irreducibility Criterion for Germs of Analytic Functions of Two Complex Variables. Advances in Math., 74, pp. 190–257 (1989).

[2] M. Iwami: Analytic Factorization of the Multivariate Polynomial. Proc. of the 6th International Workshop on Computer Algebra in Scientific Computing. pp.213–225 (2003).

[3] M. Iwami: Extension of Expansion Base Algorithm to Multivariate Analytic Factorizaion. Proc. of the 7th International Workshop on Computer Algebra in Scientific Computing. pp.269–281 (2004).

[4] M. Iwami: Extension of expansion base algorithm for multivariate analytic factorization including the case of singular leading coefficient. ACM SIGSAM Bulletin, Volume 39, Issue 4, COLUMN: Timely communications, pp.122–126 (2005).

[5] T. C. Kuo: Generalized Newton-Puiseux Theory and Hensel's Lemma in $C[[x, y]]$. Can. J. Math., Vol.XLI, No. 6, pp. 1101–1116 (1989).

[6] S. McCallum: On Testing a Bivariate Polynomial for Analytic Reducibility. J. Symb. Comput. 24, pp. 509–535 (1997).

[7] T. Sasaki and D. Inaba: Hensel Construction of $F(x, u_1, \ldots, u_l)$, $l \geq 2$, at a Singular Point and Its Applications. SIGSAM Bulletin, Vol. 34, pp.9–17 (2000).

[8] T. Sasaki and F. Kako: Solving Multivariate Algebraic Equation by Hensel Construction. Japan J. Indus. Appl. Math., 16, 257–285 (1999).

[9] T. Sasaki: Properties of Extended Hensel Factors and Application to Approximate Factorization. Preprint, Univ. Tsukuba (2000).