ISSN 1349-127X (Online)

ISSN 1349-1261 (Paper)

http://www.math.kobe-u.ac.jp/raj

Risa/Asir Journal

(New title: Journal of Free Mathematical Software)

Volume 1 (2006)

Risa/Asir Journal (Journal of Free Mathematical Software)
Volume 1 (2006). pp 1--10.

Sendra-Winkler の有理曲線パラメトライズ・アルゴリズムの高速化とその実装

藤堂秀平, 神戸大学自然科学研究科

2005 年 2 月 2 日. 改訂版 2005 年 9 月 10 日

Title: An Acceleration method of the Parametrization Algorithm of Rational Curves of Sendra-Winkler and its Implementation
Author: Shuhei Todo
Submission: February 2, 2005. Revised: September 10, 2005.
English Summary: A parametrization algorithm of rational curves is given by Sendra and Winkler [1], [2]. We will give an acceleration method for Sendra-Winkler's algorithm. The main point of the acceleration is a heuristic method to find fixed points to obtain a rational map, by which the given curve is reduced to a quadratic curve. We implement our method in Risa/Asir and examine timing data. Our implementation is published in the asir-contrib project [3].

1 はじめに

代数的閉体 K 上で定義された既約射影代数曲線 $C : F(x, y, z) = 0 \quad (n = deg(F))$ の種数 $g(C)$ を

$$g(C) = \frac{1}{2}(n-1)(n-2) - \frac{1}{2}\sum_{p \in N} r_p(r_p - 1).$$

で定義する。ただし、r_p は点 p における重複度を表し、和は隣接点もこめてとらなければならない ([1])。$g(C)$ は 0 以上の整数である。代数曲線 C は種数 $g(C) = 0$ のとき、またそのときに限り K 上の多項式 $p(t), q(t), r(t)$ によって $(x : y : z) = (p(t) : q(t) : r(t)), t \in K$ とパラメーター表示され、$(x : y : z) \in C$ と $t \in K$ が有限個の点を除いて一対一に対応することが知られている。

種数 0 の曲線、すなわち有理曲線を、有理関数によってパラメトライズする問題は、曲線上の有理点を求める問題とも関連して 19 世紀後半に Max Noether[6]、Hilbert-Hurwitz[7]、Poincare[8] 等によって取り扱われた。1990

年代、Sendra -Winkler[1][2] は体 K が計算機上で取り扱える場合（例えば K が代数的数全体のなす体 $\overline{\mathbb{Q}}$ のとき）、与えられた既約斉次多項式 $F(x,y,z)$ から対応する代数曲線の種数を計算し、もしそれが 0 ならば、この曲線をパラメトライズする多項式の組 $p(t),q(t),r(t)$ を実際に計算するための実用的なアルゴリズムを考案した。

著者はこのアルゴリズムを Risa/Asir 上に実装し実験を行った結果、このアルゴリズムをそのまま使うだけでは、次数が 6 程度の曲線に対しても相当の計算時間がかかることが分かった。ボトルネックは曲線の特異点の座標を計算する部分と随伴曲線から適当な双有理写像をつくる部分にあるが、後者に関しては、Sendra-Winkler の一般論を用いなくても、いくつかのケースはヒューリスティックな方法で計算時間を短くすることができた。この方法は外見上はかなり限られた状況の下でしか適用できないように見える。しかし、４次程度の多項式を分母分子に持つ 2 つの有理関数からパラメーターを消去して、ランダムに作った有理曲線や、いくつかの論文に登場する有名な有理曲線にこれらの方法を適用したところ、ほとんどの場合この方法は有効であることを見た。以下の議論では、$K=\overline{\mathbb{Q}}$ とし、取り扱う既約曲線 $C:F(x,y,z)=0$ は有理数体 $\mathbb{Q}$ 上定義されたものを考える。

2　ある条件が満たされる場合の高速アルゴリズム

$F(x,y,z)$ を n 次既約斉次多項式とする。m 次の斉次多項式 $A(x,y,z)\in K[x,y,z]$ によって定義される曲線 $C':A(x,y,z)=0$ が既約曲線 $C:F(x,y,z)=0$ 上の重複度 r の点（隣接点も含めて）を少なくとも重複度 $r-1$ に持つとき、曲線 C' あるいはその定義多項式を曲線 C の m 次の adjoint という ([1])。$g(C)=0$ のとき、以下の事実が成立する。

命題 1 ([2])

1. $n-1$ 個の $n-2$ 次の *adjoint*、$A_0(x,y,z),A_1(x,y,z),\ldots,A_{n-2}(x,y,z)$ が存在して、$n-2$ 次の *adjoint* 全体は

$$c_0A_0(x,y,z)+c_1A_1(x,y,z)+\ldots+c_{n-2}A_{n-2}(x,y,z),\ (c_i\in K).$$

   と表される。ここで各 $A_i(x,y,z)$ は $F(x,y,z)$ から有理的に計算できる $\mathbb{Q}$ 上の斉次多項式である。同様にして $n-1$ 次の *adjoint* が定義でき、$2n-1$ 個の基底が存在する。

2. 曲線 C 上の特異点ではない、異なる $e(\leq n-2)$ 個の点を通る $n-2$ 次の *adjoint* 全体は

$$c_0B_0(x,y,z)+\ldots+c_{n-2-e}B_{n-2-e}(x,y,z),\ (c_i\in K).$$

と表される。e 個の固定点が

$$\{(p(t):q(t):r(t));\lambda(t)=0\} \quad (p(t),q(t),r(t),\lambda(t)\in\mathbb{Q}[t],\lambda(t)\text{は既約}) \tag{1}$$

というタイプの点の和集合によって表されているとき、$B_j(x,y,z)$ は $\mathbb{Q}$ 上の斉次多項式で、$A_i(x,y,z)$ から有理的に計算できる。

ここではこの命題を証明する代わりに、$F(x,y,z)=(x^2+y^2)^3-4x^2y^2z^2$ によって定義される曲線 C を例にとって、C の 4 次の adjoint を計算してみよう。

そのためにはまず曲線 C の特異点 $S(C)$ を計算しなければならない。方程式 $F_x=F_y=F_z=0$ を終結式（または Gröbner 基底）を利用して解けば、

$$S(C)=\{(0:0:1)\}\cup\{(t:1:0);t^2+1=0\}$$

が出せる。次に、それぞれの特異点を適当な射影変換で原点にうつした後、二次変換で特異点を分解する ([4],Chap.3,Section7)。原点 $(0:0:1)$ は重複度 4 の通常でない特異点で、二次変換による分解で重複度 2 の通常特異点 (node) が二つ現れる。二つ目のものは重複度 2 の通常でない特異点 (cusp) で、二次変換で分解すると通常点になる。したがって、隣接点も含めれば、C は重複度が 4,2,2,2,2 の合計 5 つの特異点を持ち、$g(C)=0$ である。曲線 $G(x,y,z)=0$ が $(0:0:1)$ を分解することによって出てくる二つの node を通るということは、実は $G(x,y,z)=0$ が原点で x 軸、y 軸に接するということと同値である。よって、4 次の一般多項式

$$G(x,y,z)=\sum_{i+j+k=4}c_{ijk}x^iy^jz^k$$

が C の adjoint になるための条件は

$$\begin{gathered}G_{xx}(0,0,1)=G_{xy}(0,0,1)=G_{xz}(0,0,1)\\=G_{yy}(0,0,1)=G_{yz}(0,0,1)=G_{zz}(0,0,1)\\c_{103}=c_{013}=0,\\t^2+1\mid G(t,1,0)\end{gathered}$$

である。これらの関係式を用いて $G(x,y,z)$ を書き直せば、C の adjoint の一般形

$$G(x,y,z)=c_0(x^4+x^2y^2)+c_1(x^3y+xy^3)+c_2(y^4-x^4)+c_3x^2yz+c_4xy^2z$$

を得る。

Sendra-Winkler の方法 ([2]) により有理曲線をパラメトライズするために我々が必要とするのは、固定点の個数 $e=n-3$（あるいは $e=n-4$）の場

合の $B_0(x,y,z)$,
$B_1(x,y,z)$ (あるいは $B_0(x,y,z), B_1(x,y,z), B_2(x,y,z)$) である。最初の目標は、これらの多項式を効率よく計算することである。B_0, B_1 あるいは B_0, B_1, B_2 から曲線のパラメーター表示を得る方法は次節で見る。$n-1$ 次の adjoint に対しても、$e = 2n-3$, または $2n-4$ として、同様の多項式系が得られ、これらが有用になることもある（ただし、次数が 1 大きくなる分だけ計算量も増える)。以下の場合、固定点 (1) が直ちに得られるので、A_i から B_j を求めることが非常に容易になる。($n = 3, 4$ のときは A_i がすでに求めるべき多項式 B_j になっていることに注意。)

方法 1　(場合分けによる方法)

1. 重複度 3 の有理点 p がある場合。この場合 p を通る直線 L を適当に選べば L と C は p 以外で $n-3$ 個の交点を持つ。これらの点を固定点にすれば $n-2$ 次の adjoint の基底 $A_i(i = 0, 1, \ldots, n-2)$ から B_0, B_1（$n-2$ 次）を得る。

2. 重複度 4 の有理点がある場合。あるいは、さらに一般に $n-r \mid n-4$ となるような重複度 r の有理点 p がある場合。p を通る直線 L を適当に選べば L と C は p 以外で $n-r$ 個の交点を持つ。この操作を $(n-4)/(n-r)$ 回繰り返せば $n-4$ 個の点を得る。これらの点を固定点にすれば $n-2$ 次の adjoint の基底 $A_i(i = 0, 1, \ldots, n-2)$ から B_0, B_1, B_2（$n-2$ 次）を得る。

3. $n-r \mid 2n-3$ となるような重複度 r の有理点 p がある場合。p を通る直線 L を適当に選べば L と C は p 以外で $n-r$ 個の交点を持つ。この操作を $(2n-3)/(n-r)$ 回繰り返せば $2n-3$ 個の点が得られ、これらの点を固定点にすれば $n-1$ 次の adjoint の基底 $A_i(i = 0, 1, \ldots, 2n-2)$ から B_0, B_1（$n-1$ 次）を得る。

4. $n-r \mid 2n-4$ となるような重複度 r の有理点がある場合。3. と同様、$n-1$ 次の adjoint の基底 $A_i(i = 0, 1, \ldots, 2n-2)$ から B_0, B_1, B_2（$n-1$ 次）を得る。

上の 4 つは非常に特殊なケースに見えるが、4 次程度の多項式を分母分子に持つ 2 つの有理関数からパラメーターを消去して、ランダムに作った有理曲線や、いくつかの論文に登場する有名な有理曲線のほとんどは、これらのうちのどれかに該当することを実験的に確かめた。先ほどの例の場合は、原点が重複度 4 の有理点になっているので、直線 L を $y-x=0$ として、固定点 $\{(t/2 : 0 : 1); t^2-2=0\}$ が得られる。この二点を通る adjoint の一般形は

$$c_0(-x^4+x^3y-x^2y^2+xy^3)+c_1(-x^4+y^4)+c_2(xy^2z-x^2yz)$$

である。

例

$$
\begin{aligned}
F_1 &= (4y^2+4z^2)x^4+8z^3x^3+8z^2y^2x^2-8z^5x+4z^4y^2-4z^6\\
F_2 &= (2y^2+4zy+4z^2)x^3+(-4z^2y+4z^3)x^2+(6z^2y^2-4z^3y-4z^4)x\\
&\quad +4z^4y-4z^5\\
F_3 &= (8y^5-120z^3y^2-360z^4y-240z^5)x^2\\
&\quad +(-14zy^5-41z^2y^4-118z^3y^3-210z^4y^2-139z^5y-13z^6)x\\
&\quad +3z^2y^5+9z^3y^4+13z^4y^3\\
F_4 &= (y^4+8zy^3-10z^2y^2-8z^3y-7z^4)x^5\\
&\quad +(-4zy^4+34z^2y^3-18z^3y^2-18z^4y-10z^5)x^4\\
&\quad +(-17z^2y^4+56z^3y^3+3z^4y^2-26z^5y-12z^6)x^3\\
&\quad +(-25z^3y^4+36z^4y^3+29z^5y^2-8z^6y-8z^7)x^2\\
&\quad +(-17z^4y^4+3z^5y^3+19z^6y^2+8z^7y)x-5z^5y^4-6z^6y^3-2z^7y^2\\
F_5 &= 9x^5+4yx^4+(9y^2+6z^2)x^3\\
&\quad +(4y^3+13/3z^2y)x^2+(9/4y^4+13/4z^2y^2+z^4)x+y^5+2z^2y^3+z^4y\\
F_6 &= (x^2+y^2)^3-4x^2y^2z^2\\
F_7 &= 64x^8-128z^2x^6+80z^4x^4-16z^6x^2+16z^2y^6-24z^4y^4+9z^6y^2\\
&\quad (\text{リサージュ曲線 } x=\sin(3\theta), y=\sin(4\theta))\\
F_8 &= 256x^{10}-640z^2x^8+560z^4x^6-200z^6x^4+25z^8x^2+4z^6y^4-4z^8y^2\\
&\quad (\text{リサージュ曲線 } x=\sin(2\theta), y=\sin(5\theta))\\
F_9 &= (12y^3+20zy^2+10z^2y+2z^3)x^3\\
&\quad +(12zy^3+2z^2y^2-3z^3y-z^4)x^2-2z^3y^2x+z^3y^3
\end{aligned}
$$

上記場合分けによる方法と SW の一般論の計算時間の比較 (Sec)

F_i	方法 1 の場合分け番号	方法 1	SW 一般論
F_1	2	0.21	1.121
F_2	1	0.14	1 時間以上
F_3	1	0.781	1 時間以上
F_4	2	1.732	1 時間以上
F_5	適用外	—-	2.423
F_6	2	0.241	1.142
F_7	4	11.07	1 時間以上
F_8	4	12.82	1 時間以上
F_9	1	0.311	0.921

上記タイミングデータは次の環境での値である.

```
CPU: Intel(R) Pentium(R) III CPU family
   1133MHz (1129.43-MHz 686-class CPU)
real memory  = 2147418112 (2097088K bytes)
```

3　双有理変換に関する計算

$C : F(x,y,z) = 0$ を種数 0 の既約代数曲線とする。固定点の個数を $e = n-4$ として B_0, B_1, B_2 が得られる場合、曲線 C をパラメトライズするために、次のような有理変換

$$\varphi : \mathbb{P}^2(K) \ni (x : y : z) \longmapsto (B_0(x,y,z) : B_1(x,y,z) : B_2(x,y,z)) \in \mathbb{P}^2(K)$$

を定義する。$B_0(x,y,z) = B_1(x,y,z) = B_2(x,y,z) = 0$ を満たす点は φ の不確定点である。C 上にある不確定点（有限個）を取り除いたものを $\dot{C}$ と書く。このとき次の良く知られた事実が成立する。

命題 2

1. $\varphi(\dot{C})$ は $\mathbb{Q}$ 上定義された二次曲線から有限個の点を除いた集合になる。
2. φ は $\dot{C}$ から $\varphi(\dot{C})$ への単射である。
3. $\varphi(\dot{C})$ 上定義された有理変換 ψ で、$\psi \circ \varphi$ が $\dot{C}$ 上で恒等写像となるものが存在する。

したがって、問題は二次曲線をパラメトライズする問題に帰着される。

これらの事実を証明している文献が見当たらないので、ここに証明を載せておく。まず 1, 2 を証明しよう。1 は『拡張定理』および『閉包定理』([5], Chap.3 および p.376-383) から従う（あるいは [4], p142, Theorem6.2 も参照)。2 を証明するために、$\dot{C}$ 上に $\varphi(p_1) = \varphi(p_2)$ を満たす異なる二点 $p_i = (x_i : y_i : z_i) \quad (i = 1, 2)$ があると仮定する。C 上に固定点、p_1, p_2 以外から一点 $p_0 = (x_0 : y_0 : z_0)$ をとり、$\lambda_0 B_0(x_j, y_j, z_j) + \lambda_1 B_1(x_j, y_j, z_j) + \lambda_2 B_2(x_j, y_j, z_j) = 0 \quad (j = 0, 1)$ を満たすように、すべてが 0 ではない $\lambda_0, \lambda_1, \lambda_2$ を選ぶ。このとき C と $\lambda_0 B_0(x,y,z) + \lambda_1 B_1(x,y,z) + \lambda_2 B_2(x,y,z) = 0$ は p_0, p_1, p_2 を交点にもち、交点数は少なくとも

$$\sum_{p \in N} r_p(r_p - 1) + (n-4) + 3 = (n-1)(n-2) + (n-2) + 1 = n(n-2) + 1 > n(n-2)$$

となって、Bezout の定理に矛盾する。故に φ は単射。

3 でいう ψ は、次の構成的証明によって具体的に計算できる: 曲線 C の次数を n とし、$m = n+2$ とおく。$3(m+1)-1$ 個の未定係数 $a_0, \ldots, a_m, b_0, \ldots, b_m, c_0, \ldots, c_{m-1}$

を含む斉次多項式

$$\begin{aligned} H(x,y,z) &= \{(a_0B_1^m+\ldots+a_{m-1}B_1B_0^{m-1}+a_mB_0^m)B_0 \\ &\quad +(b_0B_1^m+\ldots+b_{m-1}B_1B_0^{m-1}+b_mB_0^m)B_2\}z \\ &\quad +(c_0B_1^m+\ldots+c_{m-1}B_1B_0^{m-1}+B_0^m)B_0x \end{aligned} \tag{2}$$

を作る。曲線 C と曲線 $H(x,y,z)=0$ は、係数 a_i,b_i,c_i が何であっても、C の特異点と $n-4$ 個の固定点において合計

$$(m+1)\{\sum_{p\in N} r_p(r_p-1)+(n-4)\}=(m+1)\{(n-1)(n-2)+(n-4)\} \tag{3}$$

の交点数をもつ（[4],p.110 にある交点数の定義、また特に p.113, Theorem 5.9 参照)。C 上に固定点以外から、$3(m+1)-2$ 個の点を選び、曲線 $H(x,y,z)=0$ がこれらの点を通るように係数 a_i,b_i,c_i を定めることができる。このとき (3) と合わせると、2 曲線の交点数は少なくとも

$$\begin{aligned} & (m+1)\{(n-1)(n-2)+(n-4)\}+3(m+1)-2 \\ = \ & (m+1)\{(n-1)(n-2)+n-1\}-2 \\ = \ & (m+1)\{n(n-2)+1\}-2 \\ = \ & (m+1)n(n-2)+n+1 \end{aligned}$$

になる。一方

$$deg(F)deg(H)=n\{(m+1)(n-2)+1\}=(m+1)n(n-2)+n$$

であるから、Bezout の定理より $H(x,y,z)$ は恒等的に 0 である。有理式

$$\frac{x}{z}=-\frac{(a_0u^m+\ldots+a_{m-1}uw^{m-1}+a_mw^m)w+(b_0u^m+\ldots+b_{m-1}uw^{m-1}+b_mw^m)v}{(c_0u^m+\ldots+c_{m-1}uw^{m-1}+w^m)w}$$

と、(2) の右辺の最後の x を y に置き換えた多項式から同様の方法で得られる有理式

$$\frac{y}{z}=-\frac{(a'_0u^m+\ldots+a'_{m-1}uw^{m-1}+a'_mw^m)w+(b'_0u^m+\ldots+b'_{m-1}uw^{m-1}+b'_mw^m)v}{(c'_0u^m+\ldots+c'_{m-1}uw^{m-1}+w^m)w}$$

から、求めるべき有理変換

$$\psi:\varphi(\dot{C})\ni(u:v:w)\longmapsto(x:y:z)\in\mathbb{P}^2(K)$$

が得られる。

特異点と固定点以外から選ぶ $3(m+1)-2$ 個の点は次のようにしてとるとよい:
直線 L をランダムにとり、C と L の交点を (1) の形の和集合で表す。$H(x,y,z)$

が $\{(p(t):q(t):r(t));\lambda(t)=0\}$ を零点にもつための条件は、$H(p(t),q(t),r(t))$ を $\lambda(t)$ で割った余り $R(t;a_i,b_i,c_i)$ が 0 になることであり、R の t の冪の係数より a_i,b_i,c_i に関する $\mathbb{Q}$ 上の連立一次方程式ができる。適当な数だけ直線をとってこの操作を続ければ、目的の a_i,b_i,c_i を求めることができる（それらは $\mathbb{Q}$ の元である）。もちろん、$\{(p(t):q(t):r(t));\lambda(t)=0\}$ が固定点であるときは、これらを除外する。

ψ は Gröbner 基底を用いても計算できる。そのためには、$\mathbb{Q}[x,y,u,v,k]$ のイデアル $< B_0(x,y,1)u-B_1(x,y,1), B_0(x,y,1)v-B_2(x,y,1), kB_0(x,y,1)-1 >$ から x,y,k を消去したイデアルの Gröbner 基底を計算すればよい。

これら二つの方法での計算時間を比べてみると、次の表になる。

未定係数法 vs. Gröbner 基底 (Sec)

F_i	未定係数法	Gröbner 基底
F_1	0.541	0.01
F_4	4.897	1 時間以上
F_6	0.301	0.03

B_i が簡単なときは両者に大差はないが、B_i が少し長い式になると Gröbner 基底による計算では膨大な時間がかかってしまい、未定係数法のほうが格段に有効である。

固定点の個数が $e=n-3$ で、$B_0(x,y,z),B_1(x,y,z)$ が得られる場合、問題はより簡単になる。$B_0(x,y,z)+tB_1(x,y,z)=0$ と $F(x,y,z)=0$ は、固定点以外にちょうど一つ、パラメーター t に依存する交点 $p_t=(x(t):y(t):z(t))$ をもち、t の値を適当に選ぶことによって、p_t が任意の点（有限個を除く）になるようにできる。$x(t),y(t),z(t)\in\mathbb{Q}[t]$ となることが証明され、曲線 C が $(x:y:z)=(x(t):y(t):z(t))$, $t=B_0(x,y,z)/B_1(x,y,z)$ とパラメトライズされる。$B_0(x,y,z),B_1(x,y,z)\in\mathbb{Q}[x,y,z]$ が有理的に計算できる場合は、以上のようにして曲線 C をパラメトライズするのである。通常、多項式 $x(t),y(t),z(t)$ を計算するのに終結式が用いられる（終結式による方法）が、これらの多項式を上記の未定係数法によっても求めることができる:

$$\begin{aligned} H(x,y,z) &= (a_0B_1^n+a_1B_1^{n-1}B_0+\ldots+a_nB_0^n)z \\ &\quad +(b_0B_1^n+b_1B_1^{n-1}B_0+\ldots+b_nB_0^n)x \end{aligned} \tag{4}$$

とおき、曲線 $H(x,y,z)=0$ が曲線 C の特異点と $n-3$ 個の固定点以外の $2n-1$ 個の点を通るように、係数 a_i,b_i に全てが 0 でない値を与える。この時、2 曲線 $F(x,y,z)=0, H(x,y,z)=0$ の交点数の合計は、少なくとも

$$\begin{aligned} & n[(n-1)(n-2)+(n-3)]+2(n+1)-1 \\ = \;& n[(n-1)(n-2)+(n-3)+2]+1 \\ = \;& n[n(n-2)+1]+1 \end{aligned}$$

になる。これは

$$deg(F)deg(H) = n[n(n-2)+1]$$

よりも大きい。これから、上の議論と全く同様にして

$$\frac{x}{z} = -\frac{a_0 + a_1(-t) + \cdots + a_n(-t)^n}{b_0 + b_1(-t) + \cdots + b_n(-t)^n}$$

を得る ($t = -B_1(x,y,z)/B_0(x,y,z)$)。これと、(4) の右辺の最後の x を y に変えて、同様の方法で得られる有理式

$$\frac{y}{z} = -\frac{a'_0 + a'_1(-t) + \cdots + a'_n(-t)^n}{b'_0 + b'_1(-t) + \cdots + b'_n(-t)^n}$$

を合わせれば、曲線 C をパラメトライズできる。

以上、$n-2$ 次の adjoint から得られる多項式系を用いたが、$n-1$ 次の adjoint から得られる多項式系を用いても全く同様の議論が行える。

未定係数法 vs. 終結式 (Sec)

F_i	未定係数法	終結式
F_2	0.1	0.01
F_3	0.331	0.35
F_9	0.14	0.06

上の表のように、大抵の場合、未定係数法は終結式にわずかに負ける。それでも Gröbner 基底による計算より計算時間はずっと短い。

最後に、前節の例で出した曲線 $C : (x^2+y^2)^3 - 4x^2y^2z^2 = 0$ をパラメトライズしてみよう。

$$B_0 = -x^4 + x^3y - x^2y^2 + xy^3, \quad B_1 = -x^4 + y^4, \quad B_2 = xy^2z - x^2yz$$

であった。有理変換 φ によって C は、二次曲線 $-2u^2 + 2uv - v^2 + 4w^2 = 0$ に変換される。この計算方法は [2] にある。この二次曲線は有理点 $(0:2:1)$ をもつので、$\mathbb{Q}$ 上の有理関数でパラメトライズできる。結局、C は

$$\begin{aligned}
&(x:y:z)\\
&= (-16t^5 - 40t^4 - 32t^3 - 8t^2 :\\
&\quad 16t^4 + 32t^3 + 20t^2 + 4t :\\
&\quad 8t^6 + 24t^5 + 36t^4 + 32t^3 + 18t^2 + 6t + 1),
\end{aligned}$$

$$t = \frac{-x^4 + yx^3 - y^2x^2 + y^3x}{x^4 - 2zyx^2 + 2zy^2x - y^4}$$

というパラメーター表示をもつ。

4 実装

我々の実装は、Risa/Asir Contrib の代数曲線論用パッケージ上 ([3]) で試すことができる。有理曲線のパラメトライゼーションを取り扱ったパッケージとしては、RISC-Linz で開発された CASA がある。CASA は Maple 用パッケージとして配布されている。我々のパッケージが CASA から改良された点は次の通り。

1. CASA の実装では、非常に大きな係数をもつ有理関数が返ってくる。われわれのものは、2 節で述べた方法によって、多くの例に対して係数が比較的小さい。したがって計算時間も短縮されている。
2. $\mathbb{Q}$ 上の有理関数によってパラメトライズできる場合は、必ずこのような関数を返す。したがって曲線上の有理点を求めるのにも使える。CASA では、こうはいかない。

参考文献

[1] J.F.S̃endra, F.W̃inkler, Symbolic Parametrization of Curves, Journal of Symbolic Computation **12**, (1991), 607-631.

[2] J.F.S̃endra, F.W̃inkler, Parametrization of Algebraic Curves over Optimal Field Extensions, Journal of Symbolic Computation **23**, (1997), 191-207.

[3] OpenXM; http://www.openxm.org,
`OpenXM/src/asir-contrib/packages/src/todo_parametrize`,
`OpenXM/src/asir-contrib/packages/doc/todo_parametrize_ja.tex`

[4] R.J.W̃alker, *Algebraic Curves*, Princeton University Press, 1950.

[5] D. コックス, J. リトル, D. オシー, グレブナー基底と代数多様体入門（上）, シュプリンガーフェアラーク東京, 2000.

[6] Max Noether, Rationale Ausführung der Operationen in der Theorie der algebraischen Funktionen, Mathematiche Annallen **23**, (1884), 311-358.

[7] D.H̃ilbert, A.H̃urwitz, Über die Diophantischen Gleichungen vom Geschlecht Null, Acta Mathematica **14**,(1890),217-224.

[8] M.H.P̃oincaré, Sur les propriétés arithmétiques des courbes algébriques, Journal de Mathematique pure et applique (5^esérie), tome VII, (1901), 161-233.

Risa/Asir Journal (Journal of Free Mathematical Software)
Volume 1 (2006). pp 11--52.

Mora の割り算アルゴリズムとそれを用いた local b 関数の計算アルゴリズムの Risa/Asir 上での実装

中山洋将

平成 18 年 2 月 11 日

1 Introduction

次のように記号を定めておく。

$$x = (x_1, \cdots, x_n), \partial = (\partial_1, \cdots, \partial_n), s = (s)$$

$$D[s] = \left\{ \sum_{k \in \mathbb{N}, \beta \in \mathbb{N}^n} a_{k,\beta}(x) s^k \partial^\beta \mid a_{k,\beta}(x) \in \mathbb{C}[x] \right\}$$

$$\mathbb{C}[x]_{\langle x \rangle} = \left\{ \frac{f(x)}{g(x)} \mid f(x), g(x) \in \mathbb{C}[x], g(0) \neq 0 \right\}$$

$$\mathcal{D}_{alg}[s] = \left\{ \sum_{k \in \mathbb{N}, \beta \in \mathbb{N}^n} a_{k,\beta}(x) s^k \partial^\beta \mid a_{k,\beta}(x) \in \mathbb{C}[x]_{\langle x \rangle} \right\}$$

$$\widehat{\mathcal{D}}[s] = \left\{ \sum_{k \in \mathbb{N}, \beta \in \mathbb{N}^n} a_{k,\beta}(x) s^k \partial^\beta \mid a_{k,\beta}(x) \in \mathbb{C}[[x]] \right\}$$

多項式 $f \in \mathbb{C}[x]$ について、f の global b 関数と、f の原点での local b 関数の定義と性質を簡単に説明する。

f の global b 関数とは、$Pf^{s+1} = \tilde{b}(s) f^s$ となる微分作用素 $P \in D[s]$ が存在する、最小次数で monic な s の多項式 $\tilde{b}(s) \in \mathbb{C}[s]$ のことである。

f の原点での local b 関数とは、$Pf^{s+1} = b(s) f^s$ となる微分作用素 $P \in \mathcal{D}_{alg}[s]$ が存在する、最小次数で monic な s の多項式 $b(s) \in \mathbb{C}[s]$ のことである。簡単な例を挙げる。

Example 1.1. ($f = x_1(x_1 + x_2 + 1)$ の b 関数)

global b 関数は、$b(s) = (s+1)^2$ であって、実際次のような式が成り立つ。

$$(-\partial_2^2 + \partial_1 \partial_2) f^{s+1} = (s+1)^2 f^s$$

local b 関数は、$b(s) = s + 1$ であって、次の式が成り立つ。

$$\frac{1}{1 + 2x_1 + x_2} \partial_1 f^{s+1} = (s+1) f^s$$

b 関数は最初に佐藤幹夫により定義され、それとは独立に Bernstein によっても定義され、global b 関数の存在性が証明された ([17], [18])。

global b 関数は local b 関数で割りきれる ($D[s] \subset \mathcal{D}_{alg}[s]$ だから)。このことは、あとの local b 関数の計算アルゴリズムで用いられる。原点において f が非特異であれば、 f の原点での local b 関数は $s+1$ になる (簡単な計算から)。また、b 関数の根は負の有理数であることが知られている (柏原 [19])。

多項式の b 関数を求めるアルゴリズムがいくつか知られている。多項式の global b 関数を求めるアルゴリズムは、(大阿久 [2], [11], [12]) 等で与えられている。また大阿久自身による、数式処理ソフト Kan/sm1 ([21]) における、それらアルゴリズムの実装 `bfunction.sm1` がある。global b 関数を求める効率的な方法は、(野呂 [10]) で与えられている。また野呂自身による、数式処理ソフト Risa/Asir ([22]) における、それらアルゴリズムを実装したライブラリ `bfct` がある。

大阿久は local b 関数を計算するアルゴリズムを与えた ([2], [11], [12], [13])。このアルゴリズムでは、多項式環や微分作用素環におけるグレブナ基底計算が使われている。

さて、最近新たに、D 上の Mora の割り算アルゴリズムが見つけられた ([5], [6])。これは、$\mathcal{D}_{alg}$ での割り算を与えている。そこでその割り算アルゴリズムを用いて、今までとは別の local b 関数を計算するアルゴリズムを得た。またそれを Risa/Asir 上に実装し、様々な計算を行った。この論文では、そのアルゴリズムの概略を復習し、実装の詳細とその結果について述べる。アルゴリズムの数学的な考察については [9] で述べている。

2 知られている b 関数の計算アルゴリズム

2.1 Global b 関数の計算アルゴリズム

global b 関数は次のようなアルゴリズムで計算できる ([2], [11], [12])。

Algorithm 2.1. (多項式 f の global b 関数の計算)

1. f^s の $D[s]$ における零化イデアル $\mathrm{Ann}_{D[s]}f^s$ の生成元 G の計算 (すなわち、 $P \cdot f^s = 0$ なる微分作用素 $P \in D[s]$ 全体の計算)
2. J を $G \cup \{f\}$ の生成する $D[s]$ のイデアルとして、$J \cap \mathbb{C}[s]$ の生成元の計算（この生成元が f の global b 関数である）

(1.) の $\mathrm{Ann}_{D[s]}f^s$ を計算するには、D_{n+1} におけるグレブナ基底を用いた方法が知られている ([2], [11], [12])。(2.) の $J \cap \mathbb{C}[s]$ を計算するには、次の 2 つの方法がある。

2a. J の s 以外の変数を消去する項順序 $<$ (例えば、$x_i, \partial_i > s$ なる項順序) についてのグレブナ基底を計算する。

2b. まず J のある項順序 $<$ についてのグレブナ基底 G を計算する。$\mathrm{NF}(s^i, G, <)$ (s^i の G についての正規形) を計算し、$a_l\mathrm{NF}(s^l, G, <) + \cdots + a_0\mathrm{NF}(1, G, <) = 0$ となるような最小の l と未定係数 $a_l, \cdots, a_0 \in \mathbb{C}$ を求める。この時、$a_l s^l + \cdots + a_0$ が $J \cap \mathbb{C}[s]$ の生成元となる。([10])

このように $D[s]$ におけるグレブナ基底計算を使い、global b 関数を求めることができる。簡単な計算例を示す。

Example 2.2. (global b 関数の計算)

$f = x^2 + y^2$ の global b 関数の計算について。まず、零化イデアル $\mathrm{Ann}_{D[s]}f^s$ の生成系を計算した結果は次のようになる。

$$\{x\partial_x + y\partial_y - 2s, y\partial_x - x\partial_y\}$$

この生成系と f のなす $D[s]$ のイデアル I について、 $I \cap \mathbb{C}[s]$ の計算を行う。グレブナ基底計算を用いた消去法 (上記 (2a.)) を用いてこの計算を行う。$x, y, \partial_x, \partial_y$ を消去するような項順序について、I のグレブナ基底 G を計算すれば、

$$\left\{-s^2-2s-1, (-s-1)y, (-s-1)x, y\partial_x - x\partial_y, x\partial_x + y\partial_y - 2s, x^2+y^2, -y\partial_x^2 - y\partial_y^2 + 2s\partial_y\right\}$$

となり、 $I \cap \mathbb{C}[s]$ の生成元は $G \cap \mathbb{C}[s] = \left\{-s^2-2s-1\right\}$ となる。結局、 global b 関数は $(s+1)^2$

実は、$f = \sum_{i=1}^n x_i^2$ の global b 関数は $(s+1)(s+\frac{n}{2})$ になることが知られており、定義式左辺の微分作用素として $P = \sum_{i=1}^n \partial_i^2$ がとれる。

2.2 Local b 関数の計算アルゴリズム

local b 関数を計算する大阿久のアルゴリズムについて簡単に述べる ([11], [12])。多項式環 $K[x,s]$ と微分作用素環 $D[s], D_{n+1}[y]$ におけるグレブナ基底計算が用いられている。

Algorithm 2.3. (local b 関数計算アルゴリズム 1 [11], [12])
y をパラメータとしておく。

1. $I = D_{n+1}[y] \cdot \left\{t - yf, \partial_i + y\frac{\partial f}{\partial x_i}\partial_t (i = 1, \cdots, n)\right\}$ とおく。I に対して、y の消去順序についてのグレブナ基底で、w 斉次なもの G を計算する。ただし、重みベクトル w は次のようなものである。

x_1	$\cdots$	x_n	t	y	ξ_1	$\cdots$	ξ_n	∂_t
0	$\cdots$	0	-1	1	0	$\cdots$	0	1

2. $\psi(G) = \{\psi(P(1)) \mid P(y) \in G\}$ とおく。ただし、$P \in D_{n+1}, \mathrm{ord}_w(P) = m$ ($\mathrm{ord}_w(P)$ とは P の w についての階数、すなわち P の項で w の重みづけに関して最大の値) に対して $\psi(P)$ とは

$$\psi(P)(s) = \psi(P)(t\partial_t) = \begin{cases} \mathrm{in}_w(t^m P) & (m \geq 0) \\ \mathrm{in}_w(\partial_t^{-m} P) & (m < 0) \end{cases}$$

で定義されるようなもの。(ただし、$\mathrm{in}_w(P)$ とは P の w についての主部、すなわち P において w の重みづけに関して最大の部分をとってきたもの)

3. $D[s] \cdot \psi(G)$ に対して、∂_i の消去順序についてのグレブナ基底 G_1 を計算する。$J = K[x,s] \cdot (G_1 \cap K[x,s])$ とおく。

4. local b 関数 $b(-s-1)$ は $K[[x]][s]J \cap K[s]$ の monic な生成元である。また、global b 関数 $\tilde{b}(-s-1)$ は $J \cap K[s]$ の monic な生成元である。あとは、$K[[x]][s]J \cap K[s]$ の生成元の計算さえできればよい。

5. $K[[x]][s]J \cap K[s]$ の生成元の計算について。J の生成元を $\{f_1(x,s), \cdots, f_k(x,s)\}$ とおく。
   (a) $J(0) = K[s] \cdot \{f_1(0,s), \cdots, f_k(0,s)\}$ の生成元 $f_0(s)$ の計算。
   (b) $f_0(s)$ を $\mathbb{Q}$ 上で因数分解。$f_0(s) = g_1(s)^{l_1} \cdots g_d(s)^{l_d}$
   (c) 各 $i = 1, \cdots, d$ に対して、イデアル商 $J : g_i(s)^l$ $(l = l_i, l_i+1, \cdots)$ を計算して、$g_i(s)$ の倍元でない $a_i(x,s)$ を含む $J : g_i(s)^l$ の内で l が最小のものをとり、それを l_i' と表す。
   (d) $b(s) = g_1(s)^{l_1'} \cdots g_d(s)^{l_d'}$ が生成元となる。

また、零化イデアル $\mathrm{Ann}_{D[s]}f^s$ の生成元を用いて、次のようにも計算できる。

Algorithm 2.4. (local b 関数計算アルゴリズム 2 [12])

1. 零化イデアル $Ann_{D[s]}f^s$ $D[s]$ の生成元と f の生成するの $D[s]$ イデアルを J とおく。
2. J に対して ∂_i の消去順序についてのグレブナ基底を G とする。$J' = K[x,s]\cdot(G\cap K[x,s])$ とおく。この時、local b 関数 $b(s)$ は $K[[x]][s]J\cap K[s]$ の monic な生成元である。この計算については、先のアルゴリズム 2.3 のものをそのまま用いればよい。

今回、計算時間の比較のためにこれらのアルゴリズムも Risa/Asir において実装した。

3 Local b 関数の計算アルゴリズムその１

$D[y]$ における Mora の割り算アルゴリズム (Algorithm 3.5) を紹介し、それを用いた local b 関数の計算アルゴリズム (Algorithm 3.13) を導く。

3.1 $D[y]$ 上の Mora の割り算アルゴリズム

今まで $D[s]$ で考えてきたが、あとの都合上、パラメータ変数を s から y にかえて、$D[y]$ を考える。またパラメータ変数 s はある斉次化のために用いるものとしておく。

以下の内容は、[5],[6] に書かれている微分作用素の Mora の割り算アルゴリズムの内容を少し変形したものである。(具体的には、斉次化微分作用素上の Mora の割り算アルゴリズムであったものを普通の微分作用素上の Mora の割り算アルゴリズムにした。また、パラメータ y がついている場合を考えた。)

この $D[y]$ 上の Mora の割り算アルゴリズムを用いることにより、$\mathcal{D}_{alg}[y]$ のグレブナ基底の計算を行うことができ、$D[y]$ の元が $\mathcal{D}_{alg}[y]$ のあるイデアルに属するかどうかの判定を行うことができる。

$D[y]$ 上で次の単項式順序 $<_1$ についての割り算を行いたい。

x_1	$\cdots$	x_n	y	ξ_1	$\cdots$	ξ_n
0	$\cdots$	0	1	1	$\cdots$	1
-1	$\cdots$	-1	0	0	$\cdots$	0

(適当な項順序 $<'$)

$<_1$ は項順序でないので（$x_1,\cdots,x_n$ について局所順序である）、普通の割り算アルゴリズムでは一般に無限回つづいてしまう場合がある。そこで用いられるのが、Mora の割り算アルゴリズムである。このアルゴリズムでは、新たな変数 s を使って、元に $(-1,1)$-斉次化を行う。$<_1$ に付随して決まる $D[y][s]$ 上の項順序 $<_1^s$ を使い、割り算を行う。最後に結果を非斉次化して（s に 1 を代入して）、$<_1$ についての割り算の結果が得られるというものである。

最初に $(-1,1)$-斉次化を定義する。これは x_i の重みを -1 とし、y の重みと ξ_i の重みを 1 として、重みが -1 である変数 s を各項に付けることによって全ての項を最小重みに合わすものである。

Definition 3.1. ($(-1,1)$-斉次化)

$P\in D[y]$ は、$P=\sum a_{\alpha\beta\gamma}x^\alpha\partial^\beta y^\gamma$ $(\alpha\in(\mathbb{Z}_{\geq 0})^n,\beta\in(\mathbb{Z}_{\geq 0})^n,\gamma\in\mathbb{Z}_{\geq 0})$ と表される。この時、

$$m=\min\{|\beta|-|\alpha|+|\gamma| \mid a_{\alpha\beta\gamma}\neq 0\}\quad (\text{ただし } |v| \text{ はベクトル } v \text{ の各成分の総和を表す})$$

とおいて、P の $(-1,1)$ -斉次化 $P^{(s)}$ を

$$P^{(s)} = \sum a_{\alpha\beta\gamma} x^{\alpha} \partial^{\beta} y^{\gamma} s^{-|\alpha|+|\beta|+|\gamma|-m}$$

と定義する。

また、$P \in D[y][s]$ について、$P = \sum a_{\alpha\beta\gamma\nu} x^{\alpha} \partial^{\beta} y^{\gamma} s^{\nu}$ と表されているとする。$a_{\alpha\beta\gamma\nu} \neq 0$ であるすべての $(\alpha, \beta, \gamma, \nu)$ について、$d = -|\alpha| + |\beta| + |\gamma| - |\nu|$ が成り立つ時、P は d 次の $(-1,1)$-斉次であるという。

$<_1$ とある関係が成り立つような $D[y][s]$ 上の項順序 $<_1^s$ を次のように定義する。

Definition 3.2. ($<_1$ に付随する項順序)
$D[y][s]$ 上の項順序 $<_1^s$ を次のように定める。

x_1	$\cdots$	x_n	y	s	ξ_1	$\cdots$	ξ_n
1	$\cdots$	1	0	1	0	$\cdots$	0
0	$\cdots$	0	1	0	1	$\cdots$	1
-1	$\cdots$	-1	0	0	0	$\cdots$	0

(適当な項順序 $<'$)

この順序は、x と s の全次数比較の次に $<_1$ の比較を行うものである。最初に x と s の全次数比較が来るので、うまく項順序になっている。

$(-1,1)$-斉次元について、$<_1$ の leading monomial と $<_1^s$ の leading monomial には、次のような関係が成り立つ。

Lemma 3.3. ($<_1$ と $<_1^s$ の LM)
$P \in D[y][s]$ として、P は $(-1,1)$-斉次元とする。この時、

$$\mathrm{LM}_{<_1}(P|_{s=1}) = (\mathrm{LM}_{<_1^s}(P))|_{s=1}$$

が成り立つ。

また、$P, Q \in D[y][s]$ について、P, Q ともに 同じ次数の $(-1,1)$-斉次元であるとする。この時、

$$\mathrm{LM}_{<_1^s}(P) <_1^s \mathrm{LM}_{<_1^s}(Q) \Leftrightarrow \mathrm{LM}_{<_1}(P|_{s=1}) <_1 \mathrm{LM}_{<_1}(Q|_{s=1})$$

が成り立つ。

$P, Q \in D[y][s]$ が $(-1,1)$-斉次元であり、P が Q で簡約できる場合 (即ち、$\mathrm{LM}_{<_1^s}(P)$ が $\mathrm{LM}_{<_1^s}(Q)$ で割り切れる場合) を考える。すると簡約結果 R も P と同じ次数の $(-1,1)$-斉次元であり、$\mathrm{LM}_{<_1^s}(R) <_1^s \mathrm{LM}_{<_1^s}(P)$ が成り立つが、先の補題から、$\mathrm{LM}_{<_1}(R|_{s=1}) <_1 \mathrm{LM}_{<_1}(P|_{s=1})$ が成り立つ。この事を利用して、$D[y]$ 上の Mora の割り算アルゴリズムが導かれる。

Theorem 3.4. ($D[y]$ の Mora の割り算アルゴリズム, [5] [6])
$P, P_1, \cdots, P_m \in D[y]$ が与えられている。この時、$a(x) \in \mathbb{C}[x], Q_1, \cdots, Q_m \in D[y], R \in D[y]$ が存在して、次のことが成り立つ。

$$a(x)P = Q_1P_1 + \cdots + Q_mP_m + R$$
$$a(0) \neq 0 \text{ (すなわち、} a(x) \text{ は単元)}$$
$$Q_i \neq 0 \Rightarrow \mathrm{LM}_{<_1}(Q_iP_i) \leq_1 \mathrm{LM}_{<_1}(P)$$
$$R \neq 0 \Rightarrow \mathrm{LM}_{<_1}(R) \text{ は } \mathrm{LM}_{<_1}(P_i) \text{ のいづれでも割り切れない}$$

また、この $a(x), Q_1, \cdots, Q_m, R$ を計算するアルゴリズムは次のように与えられる。

Algorithm 3.5. (Mora の割り算アルゴリズム, [5],[6])

MoraDivision$(P, \{P_1, \cdots, P_m\}, <_1)$

input $P, P_1, \cdots, P_m \in D[y]$

output $a(x) \in \mathbb{C}[x], Q_1, \cdots, Q_m, R \in D[y]$ で定理 3.4 の条件を満たす。

$\mathcal{G} \leftarrow [P_1^{(s)}, \cdots, P_m^{(s)}]$
$R \leftarrow P^{(s)}$
$A \leftarrow 1$
$Q \leftarrow [0, \cdots, 0]$　(Q は m 個の要素を持つ配列)
if $(R \neq 0)$
　$\mathcal{F} \leftarrow \left\{P' \in \mathcal{G} \mid \exists l \in \mathbb{Z}_{\geq 0} \text{ s.t. } \mathrm{LM}_{<_1^s}(P') | \mathrm{LM}_{<_1^s}(s^l R)\right\}$
else
　$\mathcal{F} \leftarrow \phi$
$\mathcal{H} \leftarrow []$
while $(\mathcal{F} \neq \phi)\{$
　$P' \leftarrow (\mathcal{F}$ の中で l が最小の元 ($\mathcal{G}$ の i 番目の元としておく))
　if $(l > 0)\{$
　　$\mathcal{G} \leftarrow$ append $(\mathcal{G}, [R])$
　　$\mathcal{H} \leftarrow$ append $(\mathcal{H}, [[A, Q]])$
　$\}$
　$R \leftarrow (s^l R$ を P' で簡約した結果)
　$U \leftarrow ($上の簡約で P' に掛けた単項式)
　if $(i \leq m)$ {
　　$Q[i] \leftarrow Q[i] + U$
　} else if $(i > m)$ {
　　$[A', Q'] \leftarrow \mathcal{H}[i - m]$
　　$A \leftarrow A - UA'$
　　for $(j \leftarrow 1; j \leq m; j \leftarrow j + 1)$
　　　$Q[j] \leftarrow Q[j] - UQ'[j]$
　}
　if $(s|R)$ {
　　$\nu \leftarrow (s^k | R$ なる最大の $k)$
　　$R \leftarrow R/s^\nu$
　}
　$\mathcal{F} \leftarrow \left\{P' \in \mathcal{G} \mid \exists l \in \mathbb{Z}_{\geq 0} \text{ s.t. } \mathrm{LM}_{<_1^s}(P') | \mathrm{LM}_{<_1^s}(s^l R)\right\}$
$\}$
for $(j \leftarrow 1; j \leq m; j \leftarrow j + 1)$
　$Q[j] \leftarrow Q[j]|_{s=1}$
$R \leftarrow R|_{s=1}$
$a \leftarrow A|_{s=1}$
return $[a, Q, R]$

このアルゴリズムで集合 $\mathcal{G}$ を reducer set と呼ぶ。初期状態では $\mathcal{G} = \left\{P_1^{(s)}, \cdots, P_m^{(s)}\right\}$ であるが、s^l をかけてから reduce が行われる場合には、$\mathcal{G}$ に余りが追加される。この部分が通常の割り算と異なる。

この割り算アルゴリズムの計算例を３つ挙げよう。

Example 3.6. (Mora の割り算アルゴリズム、典型的な例)

x を $1+x$ で割り算することを考える。これは、別に Mora の割り算アルゴリズムを使わなくとも、$1+x$ が $\mathcal{D}_{alg}[y]$ において単元であるから、

$$x = \frac{1}{1+x} \cdot (1+x) + 0 \quad (\text{すなわち、} (1+x) \cdot x = (1+x) + 0)$$

となることがわかる。アルゴリズムの説明のために、Mora の割り算アルゴリズムを使った場合の計算過程を見る。

$1+x$ を $(-1,1)-$ 斉次化すると $s+x$ になる。Mora の割り算アルゴリズムを行った過程は、次のようになる。

$$\underline{x} \xrightarrow[s+x]{s\cdot,x} -x^2 \xrightarrow[\underline{x}]{-x} 0$$

矢印の下に書かれているのは reducer で、矢印の上に書かれているのは s をかけるかどうかと reducer にかける微分作用素である。下線が引かれているものは reducer set $\mathcal{G}$ に追加された微分作用素を表す。この各簡約過程を式で表せば、

$$\begin{aligned} s \cdot x &= x \cdot (s+x) - x^2 \\ -x^2 &= -x \cdot x + 0 \end{aligned}$$

これらの式を用いて、最終的に

$$(s+x) \cdot x = x \cdot (s+x) + 0$$

が得られ、この式を非斉次化して

$$(1+x) \cdot x = x \cdot (1+x) + 0$$

この場合、$D[y]$ の Mora の割り算アルゴリズムを使わなくとも、$\mathbb{C}[x]$ の Mora の割り算アルゴリズムでも計算できる。$D[y]$ の Mora の割り算アルゴリズムは、$\mathbb{C}[x]$ の Mora の割り算アルゴリズムを含んでいる。

Example 3.7. (Mora の割り算アルゴリズム、典型的な例、微分作用素を含む場合)

∂ を $1+x$ で割り算することを考える。上の例と同様に $1+x$ は $\mathcal{D}_{alg}[y]$ の単元であるから、割り算の余りは 0 であることがわかる。

Mora の割り算アルゴリズムの計算過程を見る。$1+x$ を $(-1,1)-$ 斉次化すると $s+x$ となる。

$$\underline{\partial} \xrightarrow[s+x]{s\cdot,\partial} -x\partial - 1 \xrightarrow[\underline{\partial}]{-x} \underline{-1} \xrightarrow[s+x]{s\cdot,-1} x \xrightarrow[\underline{-1}]{-x} 0$$

この簡約過程を式で表せば、

$$\begin{aligned} s \cdot \partial &= \partial \cdot (s+x) - x\partial - 1 \\ -x\partial - 1 &= -x \cdot \partial - 1 \\ s \cdot (-1) &= (-1) \cdot (s+x) + x \\ x &= (-x) \cdot (-1) + 0 \end{aligned}$$

これらの式から、

$$(s+x)^2 \cdot \partial = (s\partial + x\partial - 1) \cdot (s+x) + 0$$

よって、この結果を非斉次化して、

$$(1+x)^2\cdot\partial\ =(\partial+x\partial-1)\cdot(1+x)+0$$

を得る。

ここまでは、簡単な手計算でもできるような例を見てきたが、Mora の割り算アルゴリズムには入力によっては計算が非常に困難な場合がある。

Example 3.8. (Mora の割り算アルゴリズム、困難な例)
まず簡単な計算例として、$P=x_1x_2\partial_1\partial_2$ を $P_1=x_1\partial_1+x_1x_2\partial_2, P_2=x_2\partial_2+x_1x_2\partial_1$ で割ることを考える。([5], [6] からの計算例)
この Mora の割り算の計算過程は次の様になる。まずそれぞれの元を斉次化して、$P^{(s)}=x_1x_2\partial_1\partial_2, P_1^{(s)}=sx_1\partial_1+x_1x_2\partial_2, P_2^{(s)}=sx_2\partial_2+x_1x_2\partial_1$

$$P_3=x_1x_2\partial_1\partial_2\xrightarrow[P_1^{(s)}]{s\cdot,x_2\partial_2}P_4=-x_1x_2^2\partial_2^2-x_1x_2\partial_2\xrightarrow[P_2^{(s)}]{s\cdot,-x_1x_2\partial_2}x_1^2x_2^2\partial_1\partial_2+x_1^2x_2\partial_1\xrightarrow[P^{(s)}]{x_1x_2}$$

$$P_5=x_1^2x_2\partial_1\xrightarrow[P_1^{(s)}]{s\cdot,-x_1x_2}P_6=-x_1^2x_2^2\partial_2\xrightarrow[P_2^{(s)}]{s\cdot,x_1^2x_2}x_1^3x_2^2\partial_1\xrightarrow[P_5]{x_1x_2}0$$

計算結果は

$$(1-x_1x_2)^2P=(x_2\partial_2-x_1x_2^2\partial_2+x_1x_2)P_1+(-x_1x_2\partial_2+x_1^2x_2^2\partial_2-x_1^2x_2)P_2+0$$

となる。
一方、割られる元 P に適当な単元をかけて P_1,P_2 で割ることを考える。例えば、$(1+x_1)^n\cdot P$ を P_1,P_2 で割る時、計算時間を計測してやれば次の様になり、n が大きくなればなるほど計算が大変になっていく。

n	計算時間 (秒)	簡約回数
50	0.90 + 0.22	2762
100	11.9 + 1.1	10512
200	173.2 + 14.5	41012
300	823.9 + 61.1	91512

3.2 $D[y]$ 上の Mora の割り算アルゴリズム 2(他の単項式順序について)

前の section では、$<_1$ という単項式順序についての割り算を行うアルゴリズムを考えたが、他の単項式順序についても同様に計算することが可能である。$<_1$ についての Mora の割り算アルゴリズムの計算が大変な場合に、他の単項式順序だとうまくいく場合もある。このような計算効率の向上のために、他の単項式順序について Mora の割り算アルゴリズムを考えよう。
次のような単項式順序 $<_2$、

x_1	$\cdots$	x_n	y	ξ_1	$\cdots$	ξ_n
0	$\cdots$	0	0	1	$\cdots$	1
0	$\cdots$	0	1	0	$\cdots$	0
-1	$\cdots$	-1	0	0	$\cdots$	0

(適当な項順序 $<'$)

について割り算を行う。$<_1$ との違いは、y の大小のみである。$<_2$ は $\xi_1, \cdots \xi_n > y$ だから、$\xi_1, \cdots, \xi_n$ についての消去順序になっている。

この時、$(-1, 1)$-斉次化のやり方が、$<_1$ と異なってくる。$<_1$ の場合は、次の様な重みベクトル v_1 について、変数 s を使って斉次化している。

x_1	$\cdots$	x_n	y	s	ξ_1	$\cdots$	ξ_n
-1	$\cdots$	-1	1	-1	1	$\cdots$	1

$<_2$ の場合には、次の様な重みベクトル v_2 について、変数 s を使って斉次化する。

x_1	$\cdots$	x_n	y	s	ξ_1	$\cdots$	ξ_n
-1	$\cdots$	-1	0	-1	1	$\cdots$	1

$<_2$ に付随する $D[y][s]$ 上の項順序 $<_2^s$ は次のようになる。

x_1	$\cdots$	x_n	y	s	ξ_1	$\cdots$	ξ_n
1	$\cdots$	1	0	1	0	$\cdots$	0
0	$\cdots$	0	0	0	1	$\cdots$	1
0	$\cdots$	0	1	0	0	$\cdots$	0
-1	$\cdots$	-1	0	0	0	$\cdots$	0

(適当な項順序 $<'$)

$<_2$ についても、$<_1$ と同じように 補題 3.3、定理 3.4 が成り立って、アルゴリズム 3.5 が適用できる。

次の単項式順序 $<_3$

x_1	$\cdots$	x_n	y	ξ_1	$\cdots$	ξ_n
0	$\cdots$	0	1	0	$\cdots$	0
0	$\cdots$	0	0	1	$\cdots$	1
-1	$\cdots$	-1	0	0	$\cdots$	0

(適当な項順序 $<'$)

についても割り算を行うことができる。$(-1, 1)$-斉次化の重みベクトル v_3 は、

x_1	$\cdots$	x_n	y	s	ξ_1	$\cdots$	ξ_n
-1	$\cdots$	-1	0	-1	1	$\cdots$	1

であって、$<_3$ に付随する項順序 $<_3^s$ は、

x_1	$\cdots$	x_n	y	s	ξ_1	$\cdots$	ξ_n
0	$\cdots$	0	1	0	0	$\cdots$	0
1	$\cdots$	1	0	1	0	$\cdots$	0
0	$\cdots$	0	0	0	1	$\cdots$	1
-1	$\cdots$	-1	0	0	0	$\cdots$	0

(適当な項順序 $<'$)

とする。このようにすると、$<_3$ について Mora の割り算アルゴリズムが実行できる。

3.3 $D[y]$ 上の Mora の割り算アルゴリズムの実装と効率化

理論的な根拠はないが実験した結果から、効率化について次のような戦略が考えられる。

- どの reducer を用いるかの選択。アルゴリズム 3.5 において、$\mathcal{G}$ 中から 1 つの reducer P' を選んでいるが、それは複数の候補がある場合がある。その選択方法により計算効率が劇的に変化することがある。

  Example 3.9. （reducer 選択の例)

  reducer の選択方法として、sugar degree が最小のものを選択する方法 (min sugar) と最大のものを選択する方法 (max sugar) を比較してみる。ここで sugar degree とは全次数のことである。Buchberger algorithm において、この sugar が小さい S 多項式のペアから選んでいくと計算効率がよいことが知られている ([20])。

  $P = (1+x_1)^{50}x_1x_2\partial_1\partial_2$ を $P_1 = x_1\partial_1 + x_1x_2\partial_2, P_2 = x_2\partial_2 + x_1x_2\partial_1$ で $<_1$ について Mora の割り算を行う。

  この時、計算結果は次のようになった。

reducer 選択の方法	計算時間 (秒)	簡約回数	$\mathcal{G}$ の要素数
min sugar	0.20	1062	6
max sugar	99.3 + 9.9	2989	21

  min sugar で計算した方が、計算結果 (P にかかる単元 A や商 Q) が項の数や係数ともに max sugar に比べて小さくなる。

- modular 計算で代用。係数が膨れて計算が困難になる場合にこれを用いる。ただし、正確な計算結果を返しているとはいえないので、あくまでも結果の目安として用いる。

- $<_1$ 以外の単項式順序の採用する。$<_1$ では計算ができない場合でも、$<_2, <_3$ なら計算ができる場合がある。勿論逆の場合もある。

3.4 $\mathcal{D}_{alg}[y]$ 上のグレブナ基底計算

$D[y]$ の元で生成される $\mathcal{D}_{alg}[y], \widehat{D}[y]$ 上のイデアル $\mathcal{I}$ についてグレブナ基底を計算する方法は、次の 2 つがある。

1. $D[y]$ における Mora の割り算アルゴリズムを用いた Buchberger アルゴリズムを適用する。
2. 生成元に対して $(-1,1)$ 斉次化を行い、項順序 $<_1^s$ についてグレブナ基底を求め、得られた基底を非斉次化する。

経験的には、後者の斉次化を使った計算方法の方が速い場合が多い。また、後者で求めた基底の数が、前者で求めた基底の数よりも大きくなる傾向がある。具体的に例を挙げる。

Example 3.10. (Mora の割り算を使ったグレブナ基底計算、斉次化を経由したグレブナ基底計算)
自明な計算例であるが、次のようなイデアル $\mathcal{I}$ のグレブナ基底計算を考える。

$$\mathcal{I} = \left\{P_1 = \partial, P_2 = 1 + x^3\right\}$$

これは、$P_1 = 1 + x^3$ が $\mathcal{D}_{alg}[y]$ では単元であるから、$\mathcal{I} = \mathcal{D}_{alg}[y]$ である。
Mora の割り算を用いたグレブナ基底計算では、$\mathrm{sp}_{<_1}(P_1, P_2)$ を $\{P_1, P_2\}$ で $<_1$ について割った余りは明らかに 0 だから、$\{P_1, P_2\}$ がグレブナ基底である。

斉次化を経由したグレブナ基底計算では、$\mathcal{I}^{(s)} = \left\{P_1^{(s)} = \partial, P_2^{(s)} = s^3 + x^3\right\}$ について項順序 $<_1^s$ についてグレブナ基底計算をすれば、

$$\left\{P_1^{(s)} = \partial, P_2^{(s)} = s^3 + x^3, -3x^2, 6x, -6\right\}$$

なるグレブナ基底が得られる。両方とも極小グレブナ基底の形にしてやれば $\{1\}$ となる。

計算時間などについては、 local b 関数の計算時間の計測とともに、再び Section 5 で取り上げる。

3.5 Local b 関数の計算

$\mathcal{I}$ を $\mathrm{Ann}_{D[s]}f^s$ と f の生成する $\mathcal{D}_{alg}[s]$ のイデアルとする。$\mathcal{I} \cap \mathbb{C}[s]$ の生成元が local b 関数になる。これは $\mathcal{D}_{alg}[s]$ のイデアルの消去法の問題であるが、多項式環や微分作用素環の場合のようなグレブナ基底による消去法では計算ができない。それは、local な変数 $x_1, \cdots, x_n(< 1)$ を消去し、global な変数 $s(> 1)$ を残さなければならないので、$x_1, \cdots, x_n < 1 < s$ となり、消去順序をつくれないことによる。

この生成元を求めるための一つの方法を与えよう。別の方法については次の section で与える。$\mathcal{I}$ の $\mathcal{D}_{alg}[s]$ のグレブナ基底を G とする。この G をもちいた $\mathcal{D}_{alg}[s]$ での割り算で、余りが 0 になるかどうかで $\mathcal{I}$ に元が入るかどうかを判定することができる。local b 関数 $b(s)$ は global b 関数 $\tilde{b}(s)$ の約元なので、$\tilde{b}(s)$ の各約元について $\mathcal{I}$ に入るかどうか判定して、そのうちで最小次数の元が local b 関数 $b(s)$ になる。

Algorithm 3.11. (local b 関数の計算 （総当たり法))
Input : $f \in \mathbb{C}[x]$
Output : f の local b 関数
$H \leftarrow \mathrm{Ann}_{D_n[s]}f^s$ の生成系
$\mathcal{I} \leftarrow (H \cup \{f\}$ の生成する $\mathcal{D}_{alg}[s]$ のイデアル)
$G \leftarrow \mathcal{I}$ のグレブナ基底
$BF \leftarrow f$ の global b 関数
$L \leftarrow BF$ の約元の集合
$LF \leftarrow L$ の元の内で G で Mora の割り算を行なって、余りが 0 のもの
return LF 内で最小次数の元

Example 3.12. ($f = x_1^2(x_1+1)^3$ の local b 関数の計算)
$f = x_1^2(x_1+1)^3$ の global b 関数 $\tilde{b}(s) = \frac{1}{18}(s+1)(2s+1)(3s+1)(3s+2)$ である。$\mathrm{Ann}_{D[s]}f^s$ の生成系は $\left\{g_1 = 2s + 5sx_1 - x_1\partial_1 - x_1^2\partial_1^2\right\}$ となる。J を $\mathcal{D}_{alg}[s] \cdot \{f, g_1\}$ として、J のグレブナ基底 G は $\{f, g_1\}$ となる。$\tilde{b}(s)$ の約元の内で、G で Mora の割り算を行い余りが 0 となる元は、

- $\frac{1}{18}(s+1)(2s+1)(3s+1)(3s+2)$
- $\frac{1}{6}(s+1)(2s+1)(3s+1)$
- $\frac{1}{6}(s+1)(2s+1)(3s+2)$
- $\frac{1}{2}(s+1)(2s+1)$

だから、この中の最小次数の元は $\frac{1}{2}(s+1)(2s+1)$ でこれが f の local b 関数となる。特に、$\frac{1}{2}(s+1)(2s+1)$ を G で割った計算結果から、

$$\frac{1}{2}(s+1)(2s+1)f^s = \frac{1}{2}\cdot\frac{1}{(x_1+1)^2(5x_1+2)^3}\cdot(4\partial_1+14x_1\partial_1+10x_1^2\partial_1-12-15x_1)\cdot\partial_1\cdot f^{s+1}$$

がわかる。

ここで問題となってくるのは、global b 関数の次数が大きくなるほど約元の数も莫大になるということである。しかし、実は全ての約元に対してイデアルに入るかどうかの判定をしなくてもよい。

$$g(s)\in\mathcal{I}\cap\mathbb{C}[s] \Leftrightarrow \exists(g(s)\text{ の約元})\in\mathcal{I}\cap\mathbb{C}[s]$$

だから、$g(s)\notin\mathcal{I}$ なることが確定したら、$g(s)$ の約元については調べる必要がないことがわかる。このことを利用した方法を次に説明しよう。

global b 関数が

$$\tilde{b}(s)=b_1(s)^{e_1}b_2(s)^{e_2}\cdots b_l(s)^{e_l}\quad(b_i(s)=s+a_i\quad a_i\in\mathbb{Q}_{>0})$$

と表されるとせよ。

$$f_i=\tilde{b}(s)/b_i(s)$$

とおく。$\mathcal{J}$ のグレブナ基底 G と Mora の割り算の剰余を用いて、つぎのようにイデアル所属判定ができたとする。

$$f_{i_1}(s),\cdots,f_{i_m}(s)\in\mathcal{I}\quad f_{j_1},\cdots,f_{j_{l-m}}\notin\mathcal{I}$$

もし、f_i の内で、$\mathcal{I}$ に入るものがなければ、この時点で $\mathcal{I}\cap\mathbb{C}[s]$ の生成元は $\tilde{b}(s)$ と確定する。それ以外の場合、すなわち $m>0$ の場合を考える。ここで、$g(s)=\gcd(f_{i_1}(s),\cdots,f_{i_m}(s))$ とせよ。$g(s)\in\mathcal{I}$ である。上の $\tilde{b}(s)$ の部分を $g(s)$ に置き換えて、同じ操作を終わるまで繰り返せばよい。これをまとめれば、次のようになる。

Algorithm 3.13. (local-b 関数の計算 (総当たり法改良版))

localb-rr(f)

input $f\in\mathbb{C}[x]$

output f の local b 関数

$H\leftarrow \mathrm{Ann}_{D_n[s]}f^s$ の生成系

$\mathcal{I}\leftarrow(H\cup\{f\}$ の生成する $\mathcal{D}_{alg}[s]$ のイデアル $)$

$G\leftarrow\mathcal{I}$ のグレブナ基底

$BF\leftarrow f$ の global b 関数

$LF\leftarrow BF$

while (true) {

  $LF=b_1(s)^{e_1}\cdots b_l(s)^{e_l}\quad(b_i(s)=s+a_i,a_i\in\mathbb{Q}_{>0})$ と表されたとする。(LF の因数分解による)

  $f_i\leftarrow LF/b_i(s)\quad(1\le i\le l,$ ただし $b_i(s)|LF$ が成り立つ i に限る)

  f_i が $\mathcal{I}$ に入るかどうかの判定を行う。

  (f_i を G で Mora の割り算アルゴリズムにより割って、余りが 0 かどうかで判定する。)

  if $(f_i\notin\mathcal{I}(1\le\forall i\le l))$

    return LF

  $f_{i_1},\cdots,f_{i_m}\in\mathcal{I}$ であったとする。

  $LF\leftarrow\gcd(f_{i_1},\cdots,f_{i_m})$

}

Example 3.14. ($f=(x-1)^3+(y+1)^2$ の local b 関数)

$f=(x-1)^3+(y+1)^2$ の global b 関数は $\tilde{b}(s)=(s+1)(s+\frac{5}{6})(s+\frac{7}{5})$ である。また、この場合 f は原点で非特異なので、計算するまでもなく local b 関数は $s+1$ とわかるのであるが、上の方法で計算してみる。

$\mathrm{Ann}_{D[s]}f^s$ の生成系は $\{g_1=-6s-2\partial_x+3\partial_y+2x\partial_x+3y\partial_y, g_2=2\partial_x-3\partial_y+2y\partial_x+6x\partial_y-3x^2\partial_y\}$ となる。$\mathcal{I}$ を $\mathcal{D}_{alg}[s]\cdot\{f,g_1,g_2\}$ として、$\mathcal{I}$ のグレブナ基底 G は $\{f,g_1,g_2\}$ となる。

$f \xrightarrow{G} g$ と書くと、f を G で Mora の割り算アルゴリズムを用いて割った余りが g であることを表すとする。

次は自明であるが、計算してみると確かに

$$\tilde{b}(s)=(s+1)(s+\frac{5}{6})(s+\frac{7}{5}) \xrightarrow{G} 0$$

$\tilde{b}(s)$ の因子の指数を一つ下げた約元について

$$f_1=(s+1)(s+\frac{5}{6}) \xrightarrow{G} 0$$
$$f_2=(s+1)(s+\frac{7}{5}) \xrightarrow{G} 0$$
$$f_3=(s+\frac{5}{6})(s+\frac{7}{5}) \xrightarrow{G} \text{non-zero}$$

$(s+1)=\gcd(f_1,f_2)$ であり、

$$s+1 \xrightarrow{G} 0$$

となって、結局 local b 関数 $b(s)=s+1$ となることがわかる。

4 Local b 関数の計算アルゴリズムその 2

$\widehat{\mathcal{D}}[s]$ における割り算定理 (Theorem 4.5) とその近似割り算アルゴリズム (Algorithm 4.6) を紹介し、それを用いた local b 関数計算アルゴリズム (Algorithm 4.10) を導く。これらアルゴリズムの正しさの証明については [9] で述べている。

この Section で使う記号の説明をしておく。$\mathrm{LE}_<(f)$ は f の先頭項の指数ベクトルを表す。$\mathrm{Exps}(f)$ は f の各項の指数ベクトルを全て集めた集合を表す。$\mathrm{Mono}(A)$ $(A=\{\alpha_1,\cdots,\alpha_n \mid \alpha_i\in(\mathbb{Z}_{\geq 0})^n\})$ は指数ベクトルの集合 A のモノイデアルを表す。すなわち、$\mathrm{Mono}(A)=\cup_{i=1}^n(\alpha_i+(\mathbb{Z}_{\geq 0})^n)$ である。微分作用素 P、重みベクトル e について、$\mathrm{ord}_e(P)$ は P の e による階数を表す。すなわち、P の各項について e で重み付けを行ったときの重みの最大値のことである。$\mathrm{in}_e(P)$ は P の e による主部を表す。すなわち、P において e で重み付けした時の重みが最大の項を足し合わせたものである。

4.1 $\mathbb{C}[[x]]$ の割り算定理とその近似アルゴリズム

$\widehat{\mathcal{D}}$ の近似割り算アルゴリズムを与えるために、その準備として $\mathbb{C}[[x]]$ の割り算である Weierstrass-Hironaka の割り算 (WH 割り算) について復習する。また $\mathbb{C}[[x]]$ の近似割り算アルゴリズムについて述べる。

巾級数環についての近似割り算アルゴリズムや近似グレブナ基底については、[15] で詳しく述べられている。(そこでは M-reduction, M-Gröbner basis と呼ばれている。)

$\{x^\alpha\}$ $(\alpha \in (\mathbb{Z}_{\geq 0})^n)$ 上の全順序 $<_r$ を次の行列で定義されるものとする。

$$\begin{array}{ccc} x_1 & \cdots & x_n \\ \hline -1 & \cdots & -1 \end{array}$$

(適当な項順序 $<$)

すなわち、$<_r$ は $\mathbb{C}[x]$ 上の逆次数順序である。$<_r$ は項順序でないため、普通の多項式の割り算アルゴリズムでは割り算が有限回で終わるとは限らないことに注意する。

次の補題は、$f \in \mathbb{C}[[x]]$ を ある単項式の集合で順序 $<_r$ について割る時、商と余りが存在することを述べたものである。

Lemma 4.1. (単項式による完全簡約)

任意の単項式の集合 $\left\{x^{\alpha(1)}, \cdots, x^{\alpha(s)}\right\}$ $(\alpha(i) \in (\mathbb{Z}_{\geq 0})^n)$ と $f \in \mathbb{C}[[x]]$ に対して、ある $q_1, \cdots, q_s, r \in \mathbb{C}[[x]]$ が存在して次のことが成り立つ。

$$\begin{aligned} & f = q_1 x^{\alpha(1)} + \cdots + q_s x^{\alpha(s)} + r \\ & q_i \neq 0 \text{ ならば } \mathrm{LM}_{<_r}(q_i x^{\alpha(i)}) \leq_r \mathrm{LM}_{<_r}(f) \\ & r \neq 0 \text{ ならば } \mathrm{Exps}(r) \cap \mathrm{Mono}(\{\alpha(1), \cdots, \alpha(s)\}) = \phi \end{aligned}$$

この補題の計算を用いることにより、次の割り算定理が得られる。

Theorem 4.2. (Weierstrass-Hironaka の割り算定理 ,[3])

$f \in \mathbb{C}[[x]], f \neq 0$ と $G = \{g_1, \cdots, g_s\} \subset \mathbb{C}[[x]]$ に対して、ある $q_1, \cdots, q_s, r \in \mathbb{C}[[x]]$ が存在して次の条件を満たす。

$$\begin{aligned} & f = q_1 g_1 + \cdots + q_s g_s + r \\ & q_i \neq 0 \text{ ならば } \mathrm{LM}_{<_r}(q_i g_i) \leq_r \mathrm{LM}_{<_r}(f) \\ & \mathrm{Exps}(r) \cap \mathrm{Mono}(\{\mathrm{LE}_{<_r}(g_1), \cdots, \mathrm{LE}_{<_r}(g_s)\}) = \phi \end{aligned}$$

多項式環における Mora の割り算アルゴリズムによる余りは r は、先頭項のみが reducer でそれ以上簡約できないという条件だけで、それより後の項については何もいっていない。だから Mora の割り算アルゴリズムによる商と余りは、上の WH 割り算の商と余りの条件を満たさない。余りを完全簡約するには、一般に無限回の計算を必要とする。そのため、ある次数で計算打ち切ることにより有限回の計算でおさめる。ある次数までは正確な商と余りが得られるような近似割り算が得られる。

Algorithm 4.3. (WH 割り算の近似)

Input $f \in \mathbb{C}[x], G = \{g_1, \cdots, g_s\} \subset \mathbb{C}[x], N \in \mathbb{Z}_{>0}$

Output $\overline{Q'_1}, \cdots, \overline{Q'_s}, \overline{R'}, \overline{F} \in \mathbb{C}[x]$ で次を満たす。

$f = \overline{Q'_1} g_1 + \cdots + \overline{Q'_s} g_s + \overline{R'} + \overline{F}$

$\overline{Q'_i} \neq 0$ ならば $\mathrm{LM}_{<_r}(\overline{Q'_i} g_i) \leq_r \mathrm{LM}_{<_r}(f)$

$\mathrm{Exps}(\overline{R'}) \cap \mathrm{Mono}(\{\mathrm{LE}_{<_r}(g_1), \cdots, \mathrm{LE}_{<_r}(g_s)\}) = \phi$

$\overline{Q'_i}$ の全次数が $N - |\mathrm{LE}_{<_r}(g_i)| - 1$ 次の項 までは正確

$\overline{R'}$ の全次数が $N - 1$ 次の項までは正確

WH-approximate-division(f, G)

$\overline{F} \leftarrow f, \overline{Q'_i} \leftarrow 0, \overline{R'} \leftarrow 0, \beta \leftarrow 0$

while ($\overline{F} \neq 0$ and $|\beta| < N$) {

$[Q, R] \leftarrow$ mono-div$(\overline{F}, G)$ (Lemma 4.1 の操作)

$\beta \leftarrow \max_{<_r}(\{\mathrm{LE}_{<_r}(Q_1 g_1), \cdots, \mathrm{LE}_{<_r}(Q_s g_s), \mathrm{LE}_{<_r}(R)\})$
$\overline{Q_i'} \leftarrow \overline{Q_i'} + Q_i$
$\overline{R'} \leftarrow \overline{R'} + R$
$\overline{F} \leftarrow -\sum_{i=1}^{s} Q_i \mathrm{Rest}_{<_r}(g_i)$
}
return $[\overline{Q'}, \overline{R'}, \overline{F}]$

この近似割り算アルゴリズムは、$\widehat{\mathcal{D}}$ における近似割り算アルゴリズム (Algorithm 4.6) で用いられる。

Example 4.4. (WH 近似割り算の計算例)
$f = x, G = 1 + x, N = 5$ の場合 (WH-approximate-division$(x, \{1 + x\}, 5)$) の計算過程とその結果

mono-div	Q	R	Q'	R'	F	β
			0	0	x	0
$x = x \cdot 1 + 0$	x	0	x	0	$-x \cdot x$	$\mathrm{LE}(x)$
$-x^2 = -x^2 \cdot 1 + 0$	$-x^2$	0	$x - x^2$	0	$-(-x^2) \cdot x$	$\mathrm{LE}(-x^2)$
$x^3 = x^3 \cdot 1 + 0$	x^3	0	$x - x^2 + x^3$	0	$-x^3 \cdot x$	$\mathrm{LE}(x^3)$
$-x^4 = -x^4 \cdot 1 + 0$	$-x^4$	0	$x - x^2 + x^3 - x^4$	0	$-(-x^4) \cdot x$	$\mathrm{LE}(-x^4)$
$x^5 = x^5 \cdot 1 + 0$	x^5	0	$x - x^2 + x^3 - x^4 + x^5$	0	$-x^5 \cdot x$	$\mathrm{LE}(x^5)$

結果は

$$x = (x - x^2 + x^3 - x^4 + x^5) \cdot (1 + x) - x^6$$

となる。

また、この場合の正確な WH 割り算の結果は手計算から、

$$\begin{aligned} x &= \frac{x}{1+x} \cdot (1 + x) + 0 \\ &= (x - x^2 + x^3 - x^4 + x^5 - \cdots) \cdot (1 + x) + 0 \end{aligned}$$

4.2 $\widehat{\mathcal{D}}[y]$ の割り算定理とその近似アルゴリズム

Castro は [4] において、$\widehat{\mathcal{D}}$ の割り算定理を与えた。WH 割り算の場合と同様に、余りを完全簡約するには一般に無限回の簡約を必要とする。この節では、この割り算定理により求められる商と余りを与えられた次数まで正確に求める近似割り算アルゴリズムを導く。$y = (y)$ を 1 変数のパラメータとしておく。

$<_1$ は section 3.1 で考えた、$\left\{x^\alpha \xi^\beta y^\gamma\right\}$ 上の単項式順序とする。$\left\{x^\alpha \xi^\beta y^\gamma\right\}$ 上に新たな順序 $<_r$ を定義する。$<_r$ は次の行列で表される。

x_1	$\cdots$	x_n	y	ξ_1	$\cdots$	ξ_n
-1	$\cdots$	-1	-1	-1	$\cdots$	-1
(	$<_1$ で用いている項順序 $<'$					)

また、重みベクトル e を次のように定義する。

x_1	$\cdots$	x_n	y	ξ_1	$\cdots$	ξ_n
0	$\cdots$	0	1	1	$\cdots$	1

ここで $<_r$ について次の事が成り立つことに注意する。$|\beta|+|\gamma| = |\beta'|+|\gamma'|$ の時 (すなわち e についての階数が等しい時), $x^\alpha \xi^\beta y^\gamma <_1 x^{\alpha'} \xi^{\beta'} y^{\gamma'} \Leftrightarrow x^\alpha \xi^\beta y^\gamma <_r x^{\alpha'} \xi^{\beta'} y^{\gamma'}$ である。だから、$\mathrm{LM}_{<_1}(P) = \mathrm{LM}_{<_r}(\mathrm{in}_e(P))$ が成り立つ。

$D[y]$ での割り算定理に相当するものを $\widehat{\mathcal{D}}[y]$ でも考える。ただし、$D[y]$ では割り算は有限回で終了するアルゴリズムとして与えられたが、$\widehat{\mathcal{D}}[y]$ では無限回の操作が必要な場合が出てくる。これは $<_1$ が項順序でないことによる。

Theorem 4.5. ($\widehat{\mathcal{D}}[y]$ における割り算定理 ,[4])
$P, P_1, \cdots, P_s \in \widehat{\mathcal{D}}[y]$ に対して、次の条件を満すような $Q_1, \cdots, Q_s, R \in \widehat{\mathcal{D}}[y]$ が存在する。

$$P = Q_1 P_1 + \cdots + Q_s P_s + R \tag{1}$$

$$\mathrm{Exps}(R) \cap \mathrm{Mono}(\{\mathrm{LE}_{<_1}(P_1), \cdots, \mathrm{LE}_{<_1}(P_s)\}) = \phi \tag{2}$$

$$Q_k \neq 0 \text{ ならば、} \mathrm{LM}_{<_1}(Q_k P_k) \leq \mathrm{LM}_{<_1}(P) \tag{3}$$

そこで、割る元、割られる元を $D[y]$ の元として、商、余りが与えられた次数まで正確に求めることのできる近似割り算アルゴリズムを考える。WH 割り算の近似アルゴリズム (Algorithm 4.3) を用いて、これを実現する。

Algorithm 4.6. ($\widehat{\mathcal{D}}[y]$ の割り算の近似, [9])
input $P, P_1, \cdots, P_s \in D[y], N \in \mathbb{Z}_{>0}$
output $\overline{Q_1}, \cdots, \overline{Q_s}, \overline{R} \in D[y]$ s.t.
$P = \overline{Q_1} P_1 + \cdots + \overline{Q_s} P_s + \overline{R}$
$\overline{Q_i} \neq 0$ ならば $\mathrm{LM}_{<_1}(\overline{Q_i} P_i) \leq \mathrm{LM}_{<_1}(P)$
$\left\{\alpha \mid \alpha \in \mathrm{Exps}(\overline{R}), |\alpha| < N\right\} \cap \mathrm{Mono}(\{\mathrm{LE}_{<_1}(P_1), \cdots, \mathrm{LE}_{<_1}(P_s)\}) = \phi$
$\overline{Q_i}$ の全次数が $N - |\mathrm{LE}_{<_1}(P_i)|$ 次未満の部分は正確である。
$\overline{R}$ の全次数が N 次未満の部分は正確である。

D-approximate-division$(P, \{P_1, \cdots, P_s\}, N)$
$\overline{Q_1} \leftarrow 0, \cdots, \overline{Q_s} \leftarrow 0, \overline{R} \leftarrow P$
$m_0 \leftarrow \mathrm{ord}_e(\overline{R})$
for $(k \leftarrow 0; k \leq m_0; k \leftarrow k+1)$
  $M_k \leftarrow \max(\{|\mathrm{LE}_{<_1}(P_i)| + 2(\max(k - \mathrm{ord}_e(P_i), 0) \mid 1 \leq i \leq s\})$
Bound $\leftarrow N + \sum_{i=0}^{m_0} M_i$
for $(k \leftarrow m_0; k \geq 0; k \leftarrow k-1)$ {
  $\overline{r} \leftarrow$ ($\overline{R}$ のうちの階数 k で全次数が Bound 未満の部分の全表象)
  $[\overline{q_i'}, \overline{r'}] \leftarrow$ WH-approximate-division$(\overline{r}, \{\mathrm{in}_e(P_1), \cdots, \mathrm{in}_e(P_s)\}, <_r, \text{Bound})$
  $\overline{Q_i'} \leftarrow$ ($\overline{q_i'}$ の ξ を ∂ にかえたもの)
  $\overline{R} \leftarrow \overline{R} - \sum \overline{Q_i'} P_i$
  $\overline{Q_i} \leftarrow \overline{Q_i} + \overline{Q_i'}$
  Bound $\leftarrow$ Bound $- M_k$
}
return $[\overline{Q}, \overline{R}]$

Example 4.7. (D-approximate-division の計算例)
$P = \partial^2, P_1 = (1+x)\partial + x$ として、近似割り算アルゴリズム D-approximate-division$(P, \{P_1\}, 5)$ の計算過程を見る。

この計算の正確な結果は、

$$\partial^2 = (\frac{1}{1+x}\partial - \frac{1}{1+x})((1+x)\partial + x) + \frac{x-1}{1+x}$$

であり、商は

$$\frac{1}{1+x}\partial - \frac{1}{1+x} = (1 - x + x^2 - x^3 + x^4 - \cdots)(\partial - 1)$$

となって、余りは

$$\frac{x-1}{1+x} = -1 + 2x - 2x^2 + 2x^3 - 2x^4 + 2x^5 - \cdots$$

となることが手計算からわかる。

D-approximate-division$(P, \{P_1\}, 5)$ の計算過程を見よう。

$\overline{R} \leftarrow P, m_0 \leftarrow \mathrm{ord}_e(R) = 2$

$M_0 \leftarrow 1, M_1 \leftarrow 1, M_2 \leftarrow 3$

Bound $\leftarrow 5 + (1 + 1 + 3) = 10$

$\overline{r} \leftarrow (\overline{R}$ の階数 2 で全次数 Bound 次未満の部分$) = \xi^2$

$[\overline{q'_1}, \overline{r'}] \leftarrow$ WH-approximate-division($\overline{r}, \mathrm{in}_e(P_1) = (1+x)\xi, <_r$,Bound)

($\xi^2 = (1 - x + x^2 - \cdots + x^8)\xi \cdot (1+x)\xi + -x^9\xi^2$ となり、$\overline{q'_1} = (1 - x + x^2 - \cdots + x^8)\xi, \overline{r'} = -x^9\xi^2$ となる。)

$\overline{Q'_1} \leftarrow (1 - x + x^2 - \cdots + x^8)\partial$

$\overline{R} \leftarrow \overline{R} - \overline{Q'_1}P_1 = -x^9\partial^2 - \partial - x^9\partial - 1 + x - x^2 + \cdots + x^7 - x^8$

Bound $\leftarrow 10 - 3 = 7$

$\overline{r} \leftarrow (\overline{R}$ の階数 1 で全次数 Bound 次未満の部分$) = -\xi$

$[\overline{q'_1}, \overline{r'}] \leftarrow$ WH-approximate-division($\overline{r}, \mathrm{in}_e(P_1) = (1+x)\xi, <_r$, Bound)

($-\xi = -(1-x+x^2-x^3+\cdots+x^6)\cdot(1+x)\xi + x^7\xi$ となり、$\overline{q'_1} = -(1-x+x^2-x^3+\cdots+x^6), \overline{r'} = x^7\xi$ となる。)

$\overline{Q'_1} \leftarrow -(1 - x + x^2 - x^3 + \cdots + x^6)$

$\overline{R} \leftarrow \overline{R} - \overline{Q'_1}P_1 = -x^9\xi + x^7\xi - x^9\xi - 1 + 2x - 2x^2 + 2x^3 - \cdots + 2x^7 - x^8$

Bound $\leftarrow 7 - 1 = 6$

$\overline{r} \leftarrow (\overline{R}$ の階数 0 で全次数 Bound 次未満の部分$) = -1 + 2x - 2x^2 + 2x^3 - 2x^4 + 2x^5$

$[\overline{q'_1}, \overline{r'}] \leftarrow$ WH-approximate-division($\overline{r}, \mathrm{in}_e(P_1) = (1+x)\xi, <_r$, Bound)

($\overline{r} = 0 \cdot (1+x)\xi + \overline{r}$ となり、$\overline{q'_1} = 0, \overline{r'} = \overline{r}$ となる。)

$\overline{Q'_1} \leftarrow 0$

$\overline{R} \leftarrow \overline{R} = -x^9\partial^2 - \partial - x^9\partial - 1 + x - x^2 + \cdots + x^7 - x^8$

計算結果は、

$$\begin{aligned}\partial^2 = \big\{(1 - x + x^2 + \cdots + x^8)\partial - (1 - x + x^2 - \cdots + x^6)\big\} \cdot ((1+x)\partial + x) + \\ - x^9\partial^2 - x^9\partial + x^7\partial - 1 + 2x - 2x^2 + 2x^3 - 2x^4 + \cdots + 2x^7 - x^8\end{aligned}$$

となり、余りの正確な部分は

$$-1 + 2x - 2x^2 + 2x^3 - 2x^4$$

となって、確かに正しい結果と一致する。

4.3 $\mathcal{I} \cap \mathbb{C}[s]$ の計算アルゴリズム

$\widehat{\mathcal{D}}[s]$ のイデアル $\mathcal{I}$ について、$\mathcal{I} \cap \mathbb{C}[s]$ の生成元を計算する方法を考える。P を グレブナ基底 G で割り算 (Theorem 4.5) した余りを正規形と呼び、$\mathrm{NF}(P, G, <_1)$ で表す。

$g(s) = a_l s^l + a_{l-1} s^{l-1} + \cdots + a_0 \in \mathcal{I}$ について、次のことが成り立つ。G を $\mathcal{I}$ のグレブナ基底としておく。global b 関数の計算の場合 (Algorithm 2.1(2a.)) と同様にして、

$$\begin{aligned} g(s) \in \mathcal{I} &\Leftrightarrow \mathrm{NF}(g(s), G, <_1) = 0 \\ &\Leftrightarrow a_l \mathrm{NF}(s^l, G, <_1) + a_{l-1}\mathrm{NF}(s^{l-1}, G, <_1) + \cdots + a_0 \mathrm{NF}(1, G, <_1) = 0 \end{aligned}$$

だから、あらかじめ $\mathrm{NF}(s^i, G, <_1)$ を計算しておき、

$$a_l \mathrm{NF}(s^l, G, <_1) + a_{l-1}\mathrm{NF}(s^{l-1}, G, <_1) + \cdots + a_0 \mathrm{NF}(1, G, <_1) = 0$$

が成り立つような最小の l と $a_l, \cdots, a_0 \in \mathbb{C}$ を計算してやれば、$\mathcal{I} \cap \mathbb{C}[s]$ の生成元 $a_l s^l + \cdots + a_0$ が得られるわけである。

しかし、今 $\widehat{\mathcal{D}}[s]$ で考えているので、正規形 $\mathrm{NF}(s^i, G, <_1)$ は一般には無限個の項を持ち、アルゴリズミックに求められない。そこで正規形の近似を導入する。

Definition 4.8. (正規形の近似)

$P \in \widehat{\mathcal{D}}[s]$ と $\widehat{\mathcal{D}}[s]$ のあるイデアル $\mathcal{I}$ のグレブナ基底 G について、アルゴリズム 4.6 の近似割り算アルゴリズムで P を G で全次数 $N-1$ 次まで割ることにより得られた余り $\overline{R}$ を、P の G による $<_1$ についての $N-1$ 次までの正規形の近似と呼び、$\mathrm{NF}(P, G, <_1, N)$ で表す。$\mathrm{NF}(P, G, <_1)$ と $\mathrm{NF}(P, G, <_1, N)$ は、全次数が $N-1$ 次まで一致する。

正規形の近似を用いた $\mathcal{I} \cap \mathbb{C}[s]$ の生成元の計算法について説明する。ここで問題となるのは、

$$a_l \mathrm{NF}(s^l, G, <_1, N) + a_{l-1}\mathrm{NF}(s^{l-1}, G, <_1, N) + \cdots + a_0 \mathrm{NF}(1, G, <_1, N) = 0$$

が成り立っても、近似であるから、即 $a_l s^l + a_{l-1} s^{l-1} + \cdots + a_0 \in \mathcal{I}$ とは言えないことである。

$D[s]$ のある元 P が $\mathcal{I}$ に入るかどうかを判定するのに、Mora の割り算アルゴリズム (3.4) を用いればよい。余りが 0 なら $P \in \mathcal{I}$ であり、そうでなければ $P \notin \mathcal{I}$ である。

$\mathcal{I} \cap \mathbb{C}[s]$ の生成元を計算するアルゴリズムについて述べる。準備として、次のような線形空間を考える。

$$L_{d,N} = \left\{ (a_0, \cdots, a_d) \in \mathbb{C}^{d+1} \mid a_d \mathrm{NF}(s^d, G, <_1, N) + \cdots + a_0 \mathrm{NF}(1, G, <_1, N) = 0 \right\}$$

$$L_d = \left\{ (a_0, \cdots, a_d) \in \mathbb{C}^{d+1} \mid a_d \mathrm{NF}(s^d, G, <_1) + \cdots + a_0 \mathrm{NF}(1, G, <_1) = 0 \right\}$$

$\mathcal{I}$ の $<_1$ についてのグレブナ基底を G として、d を十分大きな自然数 (d は $\mathcal{I} \cap \mathbb{C}[s]$ の生成元の次数以上（ただし、この次数は現時点ではわからない) であればよい。) とし、N を適当な自然数としておく。この集合について、次のような包含関係があることに注意する。

$$\begin{array}{ccccccccc} L_{d,N} & \supset & L_{d,N+1} & \supset & L_{d,N+2} & \supset & \cdots & \supset & L_d \\ \cup & & \cup & & \cup & & & & \cup \\ L_{d-1,N} & \supset & L_{d-1,N+1} & \supset & L_{d-1,N+2} & \supset & \cdots & \supset & L_{d-1} \\ \cup & & \cup & & \cup & & & & \cup \\ \vdots & & \vdots & & \vdots & & & & \vdots \end{array}$$

$\mathcal{I} \cap \mathbb{C}[s]$ の生成元は、 $L_i \neq \{0\}, L_{i-1} = \{0\}$ なる i について、L_i の元 $(a_0, \cdots, a_i)$ からつくられる $a_i s^i + \cdots + a_0$ である。このような元を得るために、次のような手順で計算を行う。

Algorithm 4.9. ($\mathcal{I} \cap \mathbb{C}[s]$ の生成元の計算, [9])

1. $L_{d,N}, L_{d-1,N}, \cdots$ と見ていき、 $L_{i,N} \neq \{0\}, L_{i-1,N} = \{0\}$ なる i を探す。(上の包含関係から、$L_{i-1} = L_{i-2} = \cdots = \{0\}$ が確定する。)
2. $(a_i, \cdots, a_0) \in L_{i,N}$ をとってきて、$g(s) = a_i s^i + \cdots + a_0$ なる多項式を作る。
3. $g(s)$ を G で $<_1$ について Mora の割り算を行い、余りを R とする。$R = 0$ の場合、$g(s) \in \mathcal{I}$ となり、 $(a_0, \cdots, a_i) \in L_i$ で、$g(s)$ は確かに $\mathcal{I} \cap \mathbb{C}[s]$ の生成元になる。$R \neq 0$ の場合、 N を 1 増やして 1. に戻る。

N の初期値 は任意の自然数で良い。実装では、$N = d + 1$ をとって計算している。

この手順が有限回で終了するのは、

$$\exists N_0 \in \mathbb{N}, \forall n \geq N_0 \text{ s.t. } L_{j,n} = L_j \quad (j = 0, \cdots, d)$$

が成り立つからである。(最長の場合でも N が N_0 になれば計算は止まる。)

4.4 local b 関数の計算

Algorithm 4.10. (正規形の近似を用いた local b 関数の計算アルゴリズム, [9])
$f \in \mathbb{C}[x]$ の原点における local b 関数を計算する方法は次のようなものである。

1. $\mathrm{Ann}_{\widehat{\mathcal{D}}[s]} f^s$ の生成元の H の計算
2. H と f の生成する $\widehat{\mathcal{D}}[s]$ のイデアルを $\mathcal{I}$ とおく。この時、$\mathcal{I} \cap \mathbb{C}[s]$ の生成元が local b 関数 $b(s)$ となる。

1. の $\mathrm{Ann}_{\widehat{\mathcal{D}}[s]} f^s$ の生成元は、$\mathrm{Ann}_{D[s]} f^s$ の生成元をとればよい。
2. の計算については、アルゴリズム 4.9 を用いればよい。この時、local b 関数の次数は global b 関数の次数を越えることはないから、global b 関数をあらかじめ求めておき、その次数以下で $\mathcal{I} \cap \mathbb{C}[s]$ の生成元を探せばよい。

実際に計算例を挙げる。

Example 4.11. ($f = x^2(y+1)^2 z^2$ の原点での local b 関数)
まず結果から述べると、global b 関数は $(s+1)^3(s+1/2)^3$ であり、local b 関数は $(s+1)^2(s+1/2)^2$ である。
$\mathrm{Ann}_{\widehat{\mathcal{D}}[s]} f^s$ の生成元 H は、

$$\{P_1 = -2s + z\partial_z, P_2 = x\partial_x - z\partial_z, P_3 = -\partial_y - y\partial_y + z\partial_z\}$$

$\mathcal{J} = \widehat{\mathcal{D}}[s] \cdot (H \cup \{f\})$ の $<_1$ についてのグレブナ基底 G は、

$$\{f, P_1, P_2, P_3, P_4 = -z^4\partial_z^2 - 2yz^4\partial_z^2 - y^2z^4\partial_z^2 - 4z^3\partial_z - 8yz^3\partial_z - 4y^2z^3\partial_z - 2z^2 - 4yz^2 - 2y^2z^2,$$
$$P_5 = xz^3\partial_z + 2xyz^3\partial_z + xy^2z^3\partial_z + 2xz^2 + 4xyz^2 + 2xy^2z^2\}$$

$d=6, N=7$ として、$\mathrm{NF}(s^i, G, <_1, N)$ $(i=0,\cdots,d)$ を計算すれば、

$$\begin{aligned}
&\mathrm{NF}(1, G, <_1, 7) = 1\\
&\mathrm{NF}(s, G, <_1, 7) = 1/2 z\partial_z\\
&\mathrm{NF}(s^2, G, <_1, 7) = 1/4 z^2\partial_z^2 + 1/4 z\partial_z\\
&\mathrm{NF}(s^3, G, <_1, 7) = 1/8 z^3\partial_z^3 + 3/8 z^2\partial_z^2 + 1/8 z\partial_z\\
&\mathrm{NF}(s^4, G, <_1, 7) = -3/8 z^3\partial_z^3 - 31/16 z^2\partial_z^2 - 31/16 z\partial_z - 1/4\\
&\mathrm{NF}(s^5, G, <_1, 7) = 23/32 z^3\partial_z^3 + 135/32 z^2\partial_z^2 + 157/32 z\partial_z + 3/4\\
&\mathrm{NF}(s^6, G, <_1, 7) = -9/8 z^3\partial_z^3 - 447/64 z^2\partial_z^2 - 555/64 z\partial_z - 23/16
\end{aligned}$$

ここで、

$$\begin{aligned}
L_{d,N} &= \left\{(a_0,\cdots,a_d) \in \mathbb{C}^d \mid a_d\mathrm{NF}(s^d, G, <_1, N) + \cdots + a_0\mathrm{NF}(1, G, <_1, N) = 0\right\}\\
L_d &= \left\{(a_0,\cdots,a_d) \in \mathbb{C}^d \mid a_d\mathrm{NF}(s^d, G, <_1) + \cdots + a_0\mathrm{NF}(1, G, <_1) = 0\right\}
\end{aligned}$$

とおく。上で得た正規形の近似から、$L_{6,7}, L_{5,7}, \cdots$ と順に計算していくと、

$$\begin{aligned}
L_{6,7} =& \{c_0(1/4, 3/2, 13/4, 3, 1, 0, 0) + c_1(-3/4, -17/4, -33/4, -23/4, 0, 1, 0)\\
&+ c_2(23/16, 63/8, 231/16, 9, 0, 0, 1) \mid c_i \in \mathbb{C}\}\\
L_{5,7} =& \{c_0(1/4, 3/2, 13/4, 3, 1, 0, 0) + c_1(-3/4, -17/4, -33/4, -23/4, 0, 1, 0) \mid c_i \in \mathbb{C}\}\\
L_{4,7} =& \{c_0(1/4, 3/2, 13/4, 3, 1, 0, 0) \mid c_i \in \mathbb{C}\}\\
L_{3,7} =& \{0\}
\end{aligned}$$

ここまでで、$L_3 = L_2 = \cdots = L_0 = \{0\}$ は確定した。すなわち、local b 関数の次数は 3 より大きいことがわかる。$(1/4, 3/2, 13/4, 3, 1, 0, 0) \in L_{4,7}$ から、$g(s) = s^4 + 3s^3 + 13/4 s^2 + 3/2 s + 1/4$ が local b 関数の候補である。この $g(s)$ を G で $<_1$ について Mora の割り算を行えば、余りが 0 になることがわかる。よって、$g(s) \in \mathcal{J}$ となり、$g(s)$ は $\mathcal{J}$ に含まれる s について最小次数の元であることが保証される。$g(s)$ は local b 関数である。

5 Timing data(local b 関数の計算アルゴリズムの比較)

f の local b 関数を計算するアルゴリズムは、次の 3 つの部分に分けられる。

1. $\mathrm{Ann}_{\widehat{\mathcal{D}}[s]} f^s$ の生成元計算（$\mathrm{Ann}_{D[s]} f^s$ の生成元の計算）
2. $\mathcal{I} = \mathrm{Ann}_{\widehat{\mathcal{D}}[s]} f^s + \widehat{\mathcal{D}}[s] f$ のグレブナ基底計算
3. $\mathcal{I} \cap \mathbb{C}[s]$ の生成元計算

1. の計算は、大阿久による $\mathrm{Ann}_{D[s]} f^s$ の生成元を計算するアルゴリズムを用いる ([12], [2] 参照)。表では、$\mathcal{I}$ の部分である。

3. の計算方法は、次の 2 つの方法がある。

3a. 総当たり法 (Algorithm 3.13 の一部)

3b. 正規形の近似を用いた方法 (Algorithm 4.9)

表では、(3a),(3b) はそれぞれ localb-rr (round-robin), localb-nf (normal form) の部分に対応する。

2. の計算方法は、次の 2 つの方法があった。(Section 3.4)

2a. $D[s]$ における Mora の割り算アルゴリズムを用いて、Buchberger アルゴリズムを適用

2b. ある重みについて斉次化を行い、ある項順序についてのグレブナ基底を求め、非斉次化する

この 2 つの計算方法のどちらを採用するかによって、計算時間がかなり違う場合がある。

まず、GB 計算に (2a.) (Mora の割り算をもちいた GB 計算) を使った場合の計算時間について。計算には次の machine を使った。CPU : Athlon MP 1800+ (1533MHz) × 2, Memory : 3GB, OS : FreeBSD 4.8

f	$\mathcal{I}$	GB	GB の数	localb-rr	localb-nf	deg	locdeg
$x+y^2+z^2$	0.00504	0.0190	5	0.00958	0.0176	1	1
$x^2+y^2+z^2$	0.00524	0.00157	5	0.0127	0.0302	2	2
$x^3+y^2+z^2$	0.00637	0.0298	6	0.0312	0.0760	3	3
$x^4+y^2+z^2$	0.00631	0.0317	6	0.0711	0.153	4	4
$x^5+y^2+z^2$	0.00624	0.0306	6	0.142	0.552	5	5
$x^6+y^2+z^2$	0.00619	0.0291	6	0.291	1.53 + 0.322	6	6
$x^7+y^2+z^2$	0.00610	0.0281 + 0.0170	6	0.542 + 0.198	2.77 + 1.13	7	7
$x^3+xy^2+z^2$	0.0260	0.172	10	0.849	0.208	4	4
$x^4+xy^2+z^2$	0.0566	0.140	10	0.446	0.819 + 0.335	6	6
$x^5+xy^2+z^2$	0.0903	0.194	11	0.410	0.850	6	6
$x^3+y^4+z^2$	0.00858	0.118	9	0.461	1.04	6	6
$x^3+xy^3+z^2$	0.0110	0.236 + 0.337	11	2.68 + 0.338	2.90 + 0.680	8	8
$x^3+y^5+z^2$	0.00979	0.123	9	5.39 + 1.02	4.23 + 0.689	9	9

つぎに、GB 計算に (2b.) (斉次化経由の GB 計算) を使ったものについて。

f	$\mathcal{I}$	GB	GB の数	localb-rr	localb-nf	deg	locdeg
$x+y^2+z^2$	0.00504	0.00494	11	0.0176	0.0305 + 0.0160	1	1
$x^2+y^2+z^2$	0.00524	0.00224	5	0.0134	0.0319	2	2
$x^3+y^2+z^2$	0.00637	0.0694 + 0.0192	26	0.0154 + 0.037	0.303 + 0.0727	3	3
$x^4+y^2+z^2$	0.00631	1.41 + 0.27	92	1.97 + 0.653	2.93 + 0.838	4	4
$x^5+y^2+z^2$	0.00624	7.23 + 1.51	176	8.80 + 3.80	23.1 + 5.12	5	5
$x^6+y^2+z^2$	0.00619	33.0 + 3.25	322	35.7 + 17.3	128.3 + 26.9	6	6
$x^7+y^2+z^2$	0.00610	137.5 + 24.1	559	124.5 + 67.07	423.1 + 134.5	7	7
$x^3+xy^2+z^2$	0.0260	0.107	32	0.258	0.577 + 0.337	4	4
$x^4+xy^2+z^2$	0.0566	2.29 + 0.338	138	4.22 + 1.04	7.07 + 4.86	6	6
$x^5+xy^2+z^2$	0.0903	32.6 + 5.47	446	21.9 + 6.97	35.0 + 10.7	6	6
$x^3+y^4+z^2$	0.00858	1.45	110	4.66 + 1.05	7.17 + 1.72	6	6
$x^3+xy^3+z^2$	0.0110	1.09	101	7.55 + 1.71	16.4 + 2.06	8	8
$x^3+y^5+z^2$	0.00979	18.3 + 2.82	319	39.1 + 9.27	54.4 + 12.9	9	9

これらの例は、 Mora の割り算を用いた GB 計算の方が有利な例である。また、 GB 計算の部分だけでなく、その得られた基底で local b 関数を計算する部分でも速く計算ができている。斉次化による GB 計算では、得られた基底の数が大きくなっていることが観察される。

もう少し一般の計算例で見てみる。これらの計算例は [12] を参考にした。

まずは (2a.)(Mora 割り算を用いた GB 計算) の方について。

f	$\mathcal{I}$	GB	GB の数	localb-rr	localb-nf	deg	locdeg
$x^5+x^3y^3+y^5$	0.70					7	7
$x^4+x^2y^2+y^4$	0.0061	0.0138	4	0.0980	0.280	6	6
$x^3+x^2y^2+y^3$	0.031	0.0748	5			7	7
$x^3+y^3+z^3$	0.0062	0.00162	5	0.0696	0.231	5	5
$x^6+y^4+z^3$	0.0134	0.052	7	307.1 + 28.4		18	18
$x^3+y^2z^2$	0.029	0.0462	15	0.165	0.460	7	7
$(x^3-y^2z^2)^2$	0.102	7.49 + 1.67	26	48.1 + 8.82	17.3 + 3.42	14	14
$x^5-y^2z^2$	0.013	0.0827	9	1.31 + 0.330	2.88 + 0.331	13	13
$x^5-y^2z^3$	0.0088	0.0265	6	14.3 + 2.67	11.0 + 2.03	17	17
x^3+y^3-3xyz	0.019	0.473	12	0.210	0.644	5	5
$x^3+y^3+z^3-3xyz$	0.013	0.140	8	0.0304	0.0.962	3	3
$y(x^5-y^2z^2)$	0.014	16.7 + 3.43	31	125.1 + 25.8	49.3 + 10.5	18	18
$y(x^3-y^2z^2)$	0.644	2.65 + 0.668	22	1.61 + 0.335	3.18 + 0.673	10	10
$y((y+1)x^3-y^2z^2)$	0.284					11	10

次に (2b.)(斉次化経由の GB 計算) の方について。

f	$\mathcal{I}$	GB	GB の数	localb-rr	localb-nf	deg	locdeg
$x^5+x^3y^3+y^5$	0.70	0.052		2.585 + 0.712		7	7
$x^4+x^2y^2+y^4$	0.0061	0.0019	4	0.0869	0.226	6	6
$x^3+x^2y^2+y^3$	0.031	0.018	19	0.1304 + 0.0170		7	7
$x^3+y^3+z^3$	0.0062	0.0025	5	0.0714 + 0.0173	0.226	5	5
$x^6+y^4+z^3$	0.0134					18	18
$x^3+y^2z^2$	0.029	0.016	21	0.257 + 0.0265	0.372	7	7
$(x^3-y^2z^2)^2$	0.102	0.0932	49	45.93 + 3.621	12.7 + 3.63	14	14
$x^5-y^2z^2$	0.013	0.092	48	6.042 + 0.389	5.09 + 0.756	13	13
$x^5-y^2z^3$	0.0088	0.0028	6	13.96 + 0.933	12.6 + 4.60	17	17
x^3+y^3-3xyz	0.019	0.020	12	0.161 + 0.0265	0.48	5	5
$x^3+y^3+z^3-3xyz$	0.013	0.012	8	0.0286	0.0937	3	3
$y(x^5-y^2z^2)$	0.014	0.58	99	73.6 + 5.80	30.6 + 19.7	18	18
$y(x^3-y^2z^2)$	0.644	0.15	54	4.138 + 0.280	3.22	10	10
$y((y+1)x^3-y^2z^2)$	0.284	3.6 + 0.14	80	441.3 + 42.82		11	10

こちらの場合は、(2a.) も (2b.) あまり大差はないが、概して (2b.) の方が速い。一般的には (2b.) の方が速いことが経験的にわかる。

既存の local b 関数を求めるアルゴリズムとして、多項式環におけるイデアル商を用いる方法があった (Algorithm 2.3)。計算時間を比較すると次のようになった。

localb_quo がイデアル商を用いる方法で、localb_rr, localb_nf は前の表と同じである。

f	localb_quo	localb_rr	localb_nf	locdeg
$x+y^2+z^2$	0.00612	0.0336	0.0416	1
$x^2+y^2+z^2$	0.0119	0.0195	0.0370	2
$x^3+y^2+z^2$	0.0226	0.0674	0.112	3
$x^4+y^2+z^2$	0.0384	0.109	0.191	4
$x^5+y^2+z^2$	0.0612	0.179	0.589	5
$x^6+y^2+z^2$	0.129	0.326	1.89	6
$x^7+y^2+z^2$	0.247	0.791	3.95	7
$x^3+xy^2+z^2$	0.0767	1.05	0.406	4
$x^4+xy^2+z^2$	0.184	0.643	1.35	6
$x^5+xy^2+z^2$	0.247	0.791	3.95	6
$x^3+y^4+z^2$	0.0767	1.05	0.406	7
$x^3+xy^3+z^2$	0.184	0.643	1.35	8
$x^3+y^5+z^2$	0.173	0.694	1.13	9
$x^5+x^3y^3+y^5$	7.06			7
$x^4+x^2y^2+y^4$	0.155	0.118	0.300	6
$x^3+x^2y^2+y^3$	0.0932			7
$x^3+y^3+z^3$	0.112	0.0774	0.239	5
$x^6+y^4+z^3$	23.1	336		18
$x^3+y^2z^2$	0.144	0.24	0.535	7
$(x^3-y^2z^2)^2$	3.99	66.2	30.0	14
$x^5-y^2z^2$	0.817	1.72	3.31	13
$x^5-y^2z^3$	3.53	17.0	13.1	17
x^3+y^3-3xyz	0.0946	0.702	1.14	5
$x^3+y^3+z^3-3xyz$	0.0207	0.183	0.249	3
$y(x^5-y^2z^2)$	6.10	171	79.9	18
$y(x^3-y^2z^2)$	0.711	5.91	7.83	10
$y((y+1)x^3-y^2z^2)$	10.2			10

以上の例のほどんどでは、既存のイデアル商を用いて計算する方法の方が速くなる。localb_quo では途中から計算が多項式環 $\mathbb{C}[x,s]$ での計算に帰着されるのに対し、localb_rr, localb_nf では最後まで $\widehat{\mathcal{D}}[s]$ で計算を行っていることによると思われる。しかしながら、我々の方法は全ステップにおいて局所的な計算になっており、局所的な計算法（主に Mora の割り算アルゴリズムと $\widehat{\mathcal{D}}[s]$ の近似割り算）の効率化をはかることができれば、遠い特異点に邪魔されないようなアルゴリズムが得られることを期待している。

6 ライブラリ localb.rr の関数の説明

6.1 使い方

ライブラリ localb.rr の使い方を説明するために、幾つかの実行例を挙げよう。まずは、 Asir を起動し、ライブラリ localb.rr を読み込む。この時、localb.rr 関数中に ord 関数を使っているので、起動時に -norc オプションをつけて起動させる。このようにするのは、ord を複数回呼ぶとデータに不整合が生じる恐れがあるからである (詳細については、Asir のマニュアルを参照) ライブラリ中に様々な大域変数が定義されており、後で関数の引数として使う。たとえば、VL3 は変数リスト、A3 は $<_1$ に対応する順序を表す行列である。

```
nakayama@orange2(91)=> asir -norc
This is Risa/Asir, Version 20050726 (Kobe Distribution).
Copyright (C) 1994-2000, all rights reserved, FUJITSU LABORATORIES LIMITED.
Copyright 2000-2005, Risa/Asir committers, http://www.openxm.org/.
GC 6.5 copyright 1988-2005, H-J. Boehm, A. J. Demers, Xerox, SGI, HP.
PARI 2.0.17, copyright 1989-1999, C. Batut, K. Belabas, D. Bernardi,
   H. Cohen and M. Olivier.
[0] load("localb.rr");
1
[679] VL3;
[x,y,z,ss,s,dx,dy,dz,dss,ds]
[680] A3;
[ 0 0 0 1 0 1 1 1 0 0 ]
[ -1 -1 -1 0 0 0 0 0 0 0 ]
[ 0 0 0 1 0 0 0 0 0 0 ]
[ 0 0 0 0 0 1 0 0 0 0 ]
[ 0 0 0 0 0 0 1 0 0 0 ]
[ 0 0 0 0 0 0 0 1 0 0 ]
[ 1 0 0 0 0 0 0 0 0 0 ]
[ 0 1 0 0 0 0 0 0 0 0 ]
[ 0 0 1 0 0 0 0 0 0 0 ]
[ 0 0 0 1 0 0 0 0 0 0 ]
[ 0 0 0 0 0 0 0 0 0 0 ]
[681]
```

次に Mora の割り算アルゴリズム (Algorithm 3.5) を実行させてみよう。関数 weyl_mora を用いることで Mora の割り算アルゴリズムを実行する。

```
[2018] weyl_mora(x*y*dx*dy,[x*dx+x*y*dy,y*dy+x*y*dx],VL2,A2S,V2);
[0,x^2*y^2-2*x*y+1,[(-x*y^2+y)*dy+x*y,(x^2*y^2-x*y)*dy-x^2*y]]
```

上の例では、$xy\partial_x\partial_y$ を $\{x\partial_x + xy\partial_y, y + \partial_y + xy\partial_x\}$ で割った時の結果を返す。あとの引数は変数リスト VL2、順序 $<_1^s$ を表す行列 A2S、(-1,1)-斉次化のためのベクトル V2 である。入力するデータが x, y の 2 変数であるので、VL2，A2S，V2 なる 2 変数用のものを使う。返ってきた結果は、余りが 0、

左辺につく単元 $1-2xy+x^2y^2$、商が残りのリストを表している。割り算の結果は次の通りになる。

$$(1-2xy+x^2y^2)xy\partial_x\partial_y = \{(-xy^2+y)\partial_y+xy\}(x\partial_x+xy\partial_y)+\{(x^2y^2-xy)\partial_y-x^2y\}(y\partial_y+xy\partial_x)+0$$

$\mathcal{D}_{alg}[s]$ でのグレブナ基底計算の例を挙げよう。関数 mora_gr を用いることで、Mora 割り算アルゴリズムを使ったグレブナ基底計算を実行できる。

```
[2075] J;
[x*y,-y*dy+x*dx,x*dx-ss]
[2076] mora_gr(J,VL2,A2,A2S,V2);
Count :  --> criterion の適用回数
criB:0
criF:1
criM:0
[x*y,-y*dy+x*dx,x*dx-ss,y^2*dy+y]
```

この結果から、$\{xy, y\partial_y + x\partial_x, x\partial_x - s\}$ の生成する $\mathcal{D}_{alg}[s]$ のイデアルのグレブナ基底は、

$$\left\{xy, -y\partial_y + x\partial_y, x\partial_x - s, y^2\partial_y + y\right\}$$

であることがわかる。また、斉次化経由のグレブナ基底計算は関数 tangentcone_gr を用いて計算することができる。mora_gr で計算できないようなものでも tangentcone_gr を用いれば計算できることがある。逆のこともある。

```
[2026] J=jf(x^3+x*y+y^3);
[y^3+x*y+x^3,(y+3*x^2)*dy+(-3*y^2-x)*dx,(-y^3-x*y-x^3)*dx+(y+3*x^2)*ss,
(-9*x*y^2+2*y)*dy+(-3*y^2-9*x^2*y+x)*dx+(27*x*y-3)*ss,
(-3*y^3-2*x*y)*dy+(-3*x*y^2-x^2)*dx+(9*y^2+3*x)*ss]
0.01sec(0.008774sec)
[2027] tangentcone_gr(J, VL2, A2S, V2);
[y^3+x*y+x^3,(y+3*x^2)*dy+(-3*y^2-x)*dx,(-y^3-x*y-x^3)*dx+(y+3*x^2)*ss,
(-9*x*y^2+2*y)*dy+(-3*y^2-9*x^2*y+x)*dx+(27*x*y-3)*ss,
(-3*y^3-2*x*y)*dy+(-3*x*y^2-x^2)*dx+(9*y^2+3*x)*ss,
(-y^2-3*x^2*y)*dy+(2*y^3-x^3)*dx-y-3*x^2,
(-y^4+2*x^3*y)*dy+(-2*x*y^3+x^4)*dx-3*y^3+3*x^3,
(-6*y^3+3*x^3)*ss-6*y^3+3*x^3,-9*y^3*ss-9*y^3,
(27*x^2*y^2*ss-9*y^3)*dy+(-18*y^4+9*x^3*y)*ss*dx+(-18*y^2+27*x^2*y)*ss-27*y^2,
(9*x*y^3-18*x^4)*ss*dy+(27*x^2*y^2*ss-9*x^3)*dx+(27*x*y^2-18*x^2)*ss-27*x^2,
(-9*y^5-9*x^3*y^2)*ss+9*x*y^3,(-9*x^2*y^3-9*x^5)*ss+9*x^3*y,
(27*x*y^4+27*x^4*y)*ss-27*x^2*y^2,
(27*x^2*y^3*ss-9*y^4)*dy+(-27*y^5*ss+9*x*y^3)*dx,
(81*y^5*ss-54*x*y^3)*dy+(81*x*y^4*ss-27*x^2*y^2)*dx+
(378*y^4+135*x^3*y)*ss-216*x*y^2]
0.01sec(0.01984sec)
[2028] mora_gr(J, VL2, A2, A2S, V2);
-----> 計算がずっとつづいてしまう。
```

上の例だと mora_gr では、Mora の割り算部分で計算が困難なものがでてきて、計算ができない。しかし tangentcone_gr を使えば計算ができる。

local b 関数を計算するには次のように行う。まずは総当たり法 (Algorithm 3.13) で計算してみる。それには関数 localb_rr を用いる。

```
[2011] localb_rr(x*(x+1)*(y+1),VL2, A2, A2S, V2);
J :
+x+x^2+x*y+x^2*y
-ss+dy+y*dy
+dy-x*dx+2*x*dy+y*dy-x^2*dx+2*x*y*dy
Gr(J) :  --> J のグレブナ基底計算 (tangentcone_gr2 を使う)
+x
-ss+dy+y*dy
+dy+y*dy+1
+y*dy^2+y^2*dy^2+2*y*dy
+y^2*ss*dy^2-dy^2+2*y*ss*dy-2*y*dy^2-2*dy
compute global-b. --> global b 関数計算 (bfct を使う)
ok : the remainder of global-b is 0.
0th challenge
Pick :ss^2+2*ss+1
length(GenL):1
ss+1-->zero --> ss+1 をグレブナ基底で Mora 割り算
InL :1,NotInL :0
1th challenge
Pick :ss+1
length(GenL):1
1-->non-zero --> 1 をグレブナ基底で Mora 割り算
InL :0,NotInL :1
ss+1
[2012]
```

この結果から、$x(x+1)(y+1)$ の local b 関数は $s+1$ であることがわかる。この関数の中では、$\mathcal{D}_{alg}[s]$ におけるグレブナ基底計算、Mora の割り算アルゴリズム、global b 関数の計算等が行われる。

次に正規形の近似を用いた方法 (Algorithm 4.10) で計算させてみる。それには関数 localb_nf を用いる。

```
[2470] localb_nf(x*(x+1)*(y+1), VL2, A2, A2S, V2, W2, 3, 10);
J :
+x+x^2+x*y+x^2*y
-ss+dy+y*dy
+dy-x*dx+2*x*dy+y*dy-x^2*dx+2*x*y*dy
Gr(J) :  <-- J のグレブナ基底
+x
-ss+dy+y*dy
+dy+y*dy+1
+y*dy^2+y^2*dy^2+2*y*dy
+y^2*ss*dy^2-dy^2+2*y*ss*dy-2*y*dy^2-2*dy
NF :
+1 <-- 1 の Gr(J) による 10 次までの正規形の近似
-1 <-- ss の Gr(J) による 10 次までの正規形の近似
+1 <-- ss^2 の Gr(J) による 10 次までの正規形の近似
-1 <-- ss^3 の Gr(J) による 10 次までの正規形の近似
sol_space(4) <-- L_{4,10} の計算
[ 0 0 0 0 ]
[ 1 1 0 0 ]
[ -1 0 1 0 ]
[ 1 0 0 1 ]
sol_space(3) <-- L_{3,10} の計算
[ 0 0 0 ]
[ 1 1 0 ]
[ -1 0 1 ]
sol_space(2) <-- L_{2,10} の計算
[ 0 0 ]
[ 1 1 ]
sol_space(1) <-- L_{1,10} の計算
[ 0 ]
BSum: <-- local b 関数の候補
ss+1
weyl_mora(BSum, G) <-- Mora の割り算をすることによる最終確認
0
ss+1
[2471]
```

このようにして $x(x+1)(y+1)$ の local b 関数は $s+1$ であることがわかる。この関数 localb_nf の最後の引数でどの全次数まで正規形の近似を求めるかを指定し、最後より 1 つ前の引数は s のどの次数まで調べるかを指定する。

最後に大阿久の local b 関数の計算アルゴリズム (Algorithm 2.3) を実行させてみる。

```
[1549] lb2(x*(x+1)*(y+1));
comp psi(I): <-- psi(I) の計算
0sec(0.006666sec)
comp local intersection  <-- K[[x]][s] J \cap K[s] の計算
0.01sec(0.005521sec)
-s-1
[1550]
```

結果より、$s+1$ が local b 関数とわかる。$x(x+1)(y+1)$ の local b 関数を計算するので、2 変数用の関数 `lb2` を用いている。

6.2 ライブラリ全体で使用する大域変数 (変数リスト、order matrix、各種のスイッチ)

以下の変数は大域変数として localb.rr 中に定義され、関数の引数として用いられる。

- 変数リスト (VL)
  ss は パラメータ、s は $(-1,1)$ 斉次化の元、dss, ds は ダミーである。
  - 1 変数の変数リスト VL1 = [x,ss,s,dx,dss,ds]
  - 2 変数の変数リスト VL2 = [x,y,ss,s,dx,dy,dss,ds]
  - 3 変数の変数リスト VL3 = [x,y,z,ss,s,dx,dy,dz,dss,ds]

- order matrix(A, B, C)
  単項式順序は行列を用いて定義されている。
  $\mathcal{D}_{alg}[s], \widehat{\mathcal{D}}[s]$ 上の単項式順序として、$<_1$ (Section 3.1)、$<_2, <_3$ (Section 3.2) を定義した。また、それに付随する単項式順序として、$<_1^s$ (3.2)、$<_2^s, <_3^s$ (Section 3.2) を定義した。
  A* は $<_1$、B* は $<_2$、C* は $<_3$ をそれぞれ表す行列。A1 は 1 変数用、A2 は 2 変数用、A3 は 3 変数用。B1、B2、B3、C1、C2、C3 についても同様。
  A*S は $<_1^s$、 B*S は $<_2^s$、C*S は $<_3^s$ をそれぞれ表す行列。A1S は 1 変数用、A2S は 2 変数用、A3S は 3 変数用。B1S、B2S、B3S、C1S、C2S、C3S についても同様。
  具体的に A1, A1S についてどのように定義されているを見ると、

localb.rr

```
VL1 =
[ x,ss, s,dx,dss,ds]$

A1=newmat(5, 6,
[
[ 0, 1, 0, 1, 0, 0],
[-1, 0, 0, 0, 0, 0],
[ 0, 1, 0, 0, 0, 0],
[ 0, 0, 0, 1, 0, 0],
[ 1, 0, 0, 0, 0, 0]
])$

A1S=newmat(6, 6,
[
[ 1, 0, 1, 0, 0, 0],
[ 0, 1, 0, 1, 0, 0],
[-1, 0, 0, 0, 0, 0],
[ 0, 1, 0, 0, 0, 0],
[ 0, 0, 0, 1, 0, 0],
[ 1, 0, 0, 0, 0, 0]
])$
```

  となり、$<_1, <_1^s$ の条件を満たしている。残りの項順序の部分は lex order になっている。

- $(-1,1)$ 斉次化のための重みベクトル
  `V*` は重みベクトル v_1 (Section 3.2) を表す。
  v_1 とは次のような $(-1,1)$ 斉次化を行うためのベクトルであった。

x_1	$\cdots$	x_n	y	s	ξ_1	$\cdots$	ξ_n
-1	$\cdots$	-1	1	-1	1	$\cdots$	1

  `V1` は 1 変数用、`V2` は 2 変数用、`V3` は 3 変数用。
  `VV*` は重みベクトル v_2 (Section 3.2) を表す。
  v_2 とは次のような $(-1,1)$ 斉次化を行うためのベクトルであった。

x_1	$\cdots$	x_n	y	s	ξ_1	$\cdots$	ξ_n
-1	$\cdots$	-1	0	-1	1	$\cdots$	1

  `VV1` は 1 変数用、`VV2` は 2 変数用、`VV3` は 3 変数用。
  `V*` は `A*`,`A*S` とともに使われ、`VV*` は `B*`, `C*`, `B*S`, `C*S` とともに用いられる。

- e 重みベクトル
  `W*` は重みベクトル e (Section 4.5) を表す。
  e とは次のような重みベクトルで、$\widehat{\mathcal{D}}[s]$ の近似割り算の時に用いられる。

x_1	$\cdots$	x_n	y	ξ_1	$\cdots$	ξ_n
0	$\cdots$	0	1	1	$\cdots$	1

  `W1` は 1 変数用、`W2` は 2 変数用、`W3` は 3 変数用。

6.3 Mora の割り算アルゴリズムの関数

- `weyl_mora(P, PL, VL, Ord, Vect)`

引数：`P` 割られる元, `PL` 割る元のリスト, `VL` 変数リスト, `Ord` s を含んだ order ($<_1^s, <_2^s, <_3^s$ なる order), `Vect` $(-1,1)$ 斉次化のための重みベクトル

返り値：次を満たすような商 Q, 単元 A, 余り R のリスト

$$AP = Q[0]PL[0] + \cdots + Q[N-1]PL[N-1] + R$$
$$Q[i] \neq 0 \Rightarrow \mathrm{LM}_<(Q[i]PL[i]) \leq \mathrm{LM}_<(P)$$
$$R \neq 0 \Rightarrow \mathrm{LM}_<(R) \text{ は } \mathrm{LM}_<(PL[i]) \text{ のいづれでも割り切れない}$$

ただし、$<$ は `Ord` に対応する s なしの order のこと、 すなわち `Ord` が $<_1^s$ ならば $<_1$ であり、$<_2^s$ ならば $<_2$、$<_3^s$ ならば $<_3$ である。また、 N はリスト PL の長さを表す。

説明：Mora の割り算アルゴリズム（Algorithm 3.4) を実行する関数である。商と余りを返す。引数の `Ord` に $<_1^s, <_2^s, <_3^s$ のいづれかを与えることにより $<_1, <_2, <_3$ についての Mora の割り算を実行する。

実行例：$P = \partial_x^2 + \partial_x + 1$ を $PL = [\partial_x + x\partial_x + x]$ で $<_1$ について Mora の割り算を行う。

```
[381] weyl_mora(dx^2+dx+1, [dx+x*dx+x], VL1, A1S, V1);
[<0>3<[0,1,3]>+(2)2<1>3]
[x,x+1,[dx]]
```

$P = xy\partial_x\partial_y$ を $PL = [x+y+\partial_x, y+y^2+y^3]$ で $<_1$ について Mora の割り算を行う。（ここでの x, y は x_1, x_2 のことで、y はパラメータでない。）

```
[392] weyl_mora(x*y*dx*dy, [x+y+dx, y+y^2+y^3], VL2, A2S, V2);
[<[0,2,3]>+(3)1<[1,2,3]>+(4)3<1>9<3>13<3>10]
[(3*y^2+2*y+1)*x^2+(2*y^3+y^2)*x,y^2+y+1,[(dy*y^3+dy*y^2+dy*y)*x,-dy*x^2-dy*y*x]]
```

- `dp_weyl_mora(P, PL, VL, Ord, Vect)`

引数：`P` 割られる元（分散表現多項式）, `PL` 割る元のリスト（分散表現多項式のリスト）, `VL` 変数リスト, `Ord` s を含んだ order ($<_1^s, <_2^s, <_3^s$ なる order), `Vect` $(-1, 1)$ 斉次化のための重みベクトル

返り値：次を満たすような余り R

$$AP = Q[0]PL[0] + \cdots + Q[N-1]PL[N-1] + R$$
$$Q[i] \neq 0 \Rightarrow \mathrm{LM}_<(Q[i]PL[i]) \leq \mathrm{LM}_<(P)$$
$$R \neq 0 \Rightarrow \mathrm{LM}_<(R) \text{ は } \mathrm{LM}_<(PL[i]) \text{ のいづれでも割り切れない}$$

ただし、$<$ は `Ord` に対応する s なしの order のこと。N はリスト PL の長さを表す。

説明：Mora の割り算アルゴリズム（Algorithm 3.4) を実行する関数である。余りのみを返す。商の計算を必要としない分 `weyl_mora` より計算は軽い。また与える引数、返り値ともに分散表現多項式になっている。$\mathcal{D}_{alg}[y]$ のグレブナ基底計算の関数 `mora_gr` から、呼び出される Mora 割り算の関数である。

- `weyl_mora_mod(P, PL, VL, Ord, VL, Prime)`

引数：`P,PL,VL,Ord,VL` は `weyl_mora` の引数と同じ、`Prime` は $GF($ `Prime` $)$ 上の微分作用素の計算を意味する。

返り値：`weyl_mora` と同じ。

説明：`weyl_mora` の modular 計算版。

- `dp_weyl_mora_mod(P, PL, VL, Ord, VL, Prime)`

引数：`P,PL,VL,Ord,VL` は `dp_weyl_mora` の引数と同じ、`Prime` は $GF($ `Prime` $)$ 上の微分作用素の計算を意味する。

返り値：`dp_weyl_mora` と同じ。ただし、係数は有限体になっていることに注意する。

説明：`dp_weyl_mora` の modular 計算版。`mora_gr_mod` から呼び出される。

6.4 $\mathcal{D}_{alg}$ のグレブナ基底計算の関数

- `mora_gr(I, VL, Ord1, Ord2, Vect)`

引数：`I` イデアルの生成元のリスト, `VL` 変数リスト, `Ord1` この order についてのグレブナ基底を求める , `Ord2` `Ord1` に対応する s を含んだ order, `Vect` $(-1, 1)$ 斉次化のための重みベクトル

返り値：`I` の order `Ord1` についてのグレブナ基底

説明：Buchberger アルゴリズムと Mora の割り算を用いて、 $\mathcal{D}_{alg}$ のグレブナ基底計算を行う。

実行例：$\mathcal{D}_{alg}[ss]$ (ss はパラメータ) のイデアル $J = \mathcal{D}_{alg}[ss] \cdot \left\{x^2, x\partial_x - 2ss\right\}$ の $<_2$ についてのグレブナ基底計算

```
[395] J=jf(x^2);
[x^2,dx*x-2*ss]

[396] mora_gr(J, VL1, B1, B1S, VV1);
[0,1]:Rest 0                ペアの index
N:2,bits of Nf:4            S 式を割り算した時の余りが non-zero
Rest:2                      残りペア数
LM:(1)*<<1,1,0,0,0,0>>,2    余りの Leading monomial
[1,2]:Rest 1
N:3,bits of Nf:9
Rest:2
LM:(1)*<<0,2,0,0,0,0>>,2
[2,3]:Rest 1
[0,2]:Rest 0

Count :   criterion 適用回数
criB:0
criF:0
criM:2

グレブナ基底
[x^2,dx*x-2*ss,(2*ss+2)*x,-2*dx*x-4*ss^2-2*ss-2]
```

- `mora_gr_mod(I, VL, Ord1, Ord2, Vect, Prime)`

引数：`I` イデアルの生成元のリスト, `VL` 変数リスト, `Ord1` この order についてのグレブナ基底を求める, `Ord2` `Ord1` に対応する s を含んだ order, `Vect` $(-1, 1)$ 斉次化のための重みベクトル, `Prime` $GF($ `Prime` $)$ 上の計算

返り値：`I` の order `Ord1` についてのグレブナ基底の候補

説明：`mora_gr` の modular 版。Mora 割り算を modular 上で行っているので、計算結果はあくまでもグレブナ基底の候補である。

- `tangentcone_gr(I, VL, Ord, Vect)`

引数：`I` イデアルの生成元のリスト, `VL` 変数リスト, `Ord` s を含んだ order ($<_1^s, <_2^s, <_3^s$ なる order), `Vect` $(-1, 1)$ 斉次化のための重みベクトル

返り値：`I` の `Ord` に対応する order$(<_1, <_2, <_3)$ についてのグレブナ基底

説明：$(-1, 1)$ 斉次化を行い、項順序 $<_s$ についてのグレブナ基底計算して非斉次化を行うことで、$<$ についてのグレブナ基底計算をおこなう。(斉次化を経由したグレブナ基底計算)

実行例：$\{x\partial_x + xy\partial_y, y\partial_y + xy\partial_x\}$ の $<_1$ についてのグレブナ基底

```
[2599] I=[x*dx+x*y*dy,y*dy+x*y*dx];
[x*y*dy+x*dx,y*dy+x*y*dx]

[2600] tangentcone_gr(I, VL2, A2S, V2);
[0,1]:              ペアの index
N:2,bits of Nf:4    S 式を割り算した余りが non-zero だった
Rest:1              残りのペア数
[0,2]:
N:3,bits of Nf:2
Rest:2
[2,3]:
N:4,bits of Nf:5
Rest:3
[0,3]:
N:5,bits of Nf:2
Rest:4
[3,5]:
N:6,bits of Nf:3
Rest:4
[1,5]:
[4,6]:
N:7,bits of Nf:6
Rest:5
[3,6]:
N:8,bits of Nf:1
Rest:8
[0,6]:
.... 同様の計算がつづく

Count :     criterion 適用回数
criB:10
criF:73
criM:33
```

非斉次化する前のグレブナ基底の表示

```
+x*s*dx+x*y*dy
+y*s*dy+x*y*dx
-x^2*y*dx^2+x*y^2*dy^2-x*y*dx+x*y*dy
+x^2*y*dx-x*y^2*dy
-x*y^2*dx*dy+x*y^2*dy^2+x*y*dx+x*y*dy-y^2*dy
+x^2*y^2*dy-x*y^3*dy
.....
```

グレブナ基底
```
[x*y*dy+x*dx,y*dy+x*y*dx,x*y^2*dy^2+x*y*dy-x^2*y*dx^2-x*y*dx,-x*y^2*dy+x^2*y*dx
, .... ]
```

6.5 $\mathcal{D}[y]$ の正規形の近似計算の関数

- `whdiv(F, G, VL, Ord, N)`

引数：`F` 割られる元, `G` 割る元のリスト, `VL` 変数リスト, `Ord` order $<_r$, `N` 近似の精度

返り値：次を満たすような商 Q, 余り R のリスト

$P = Q[0]G[0] + \cdots + Q[M-1]G[M-1] + R$
$Q[i] \neq 0 \Rightarrow \mathrm{LM}_{<_r}(Q[i]G[i]) \leq \mathrm{LM}_{<_r}(P)$
$Q[i]$ の全次数が $N - |\mathrm{LE}_{<_r}(G[i])| - 1$ 次以下の部分、R の全次数が $N-1$ 次以下の部分は正確

ただし、M はリスト G の長さ。

説明：Weierstrass-Hironaka の割り算の近似アルゴリズム 4.3 を実行する。`d_app_div` から呼び出される。

実行例：x を $1+x$ で WH 割り算を実行する。近似の精度は $N=5$ 次未満の項まで。

```
[377] whdiv(x, [1+x], VL1, A1, 5);
[[ x^5-x^4+x^3-x^2+x ],-x^6,0]
```

これより結果は、$x = (x - x^2 + x^3 - x^4 + x^5)(1+x) - x^6$ となる。商の正確な部分 (4 次以下の部分) は $x - x^2 + x^3 - x^4$ となり、余りの正確な部分 (4 次以下の部分) は 0 となる。

- `d_app_div(P, G, VL, W, Ord, N)`

引数：`P` 割られる元, `G` 割る元のリスト, `VL` 変数リスト, `W` 重みベクトル e `Ord` order $<_1$ `N` 近似の精度

返り値：次を満たすような商 Q, 余り R のリスト

$P = Q[0]G[0] + \cdots + Q[M-1]G[M-1] + R$
$Q[i] \neq 0 \Rightarrow \mathrm{LM}_{<_1}(Q[i]G[i]) \leq \mathrm{LM}_{<_1}(P)$
$Q[i]$ の全次数が $N - |\mathrm{LE}_{<_1}(G[i])| - 1$ 次以下の部分、 R の全次数が $N-1$ 次以下の部分は正確

ただし、M はリスト G の長さを表す。

説明：$\widehat{\mathcal{D}}[y]$ の割り算の近似アルゴリズム 4.6 を実行する。local b 関数を計算する関数 `localb_nf` から呼ばれる。

実行例：∂_x^2 を $(1+x)\partial_x + x$ で割る。近似の精度は $N=5$ 次まで。(例 4.7)

```
[2607] d_app_div(dx^2,[(1+x)*dx+x],VL1,W1,A1,5);
M0 :2
M[0] :1
M[1] :1
```

```
M[2] :3
Bound :10
[2,3,4,5,6,7,8,9,10,][1,2,3,4,5,6,7,][0,]

check--------------------  計算結果 (商、余り) の確認
Sum :+dx^2
P :+dx^2
Monos :0
-------------------------

[[ (x^8-x^7+x^6-x^5+x^4-x^3+x^2-x+1)*dx-x^6+x^5-x^4+x^3-x^2+x-1 ],
-x^9*dx^2+(-x^9+x^7)*dx-x^8+2*x^7-2*x^6+2*x^5-2*x^4+2*x^3-2*x^2+2*x-1]
```

これより結果は、

$$\begin{aligned}\partial^2 =&((1-x+x^2+\cdots+x^8)\partial-(1-x+x^2-\cdots+x^6))\cdot((1+x)\partial+x)+\\ &-x^9\partial^2-x^9\partial+x^7\partial-1+2x-2x^2+2x^3-2x^4+\cdots+2x^7-x^8\end{aligned}$$

となり、余りの正確な部分 (余りの項で全次数が 4 次以下の部分) は

$$-1+2x-2x^2+2x^3-2x^4$$

となる。

6.6 local b 関数を計算する関数

- `localb_rr(F, VL, Ord1, Ord2, Vect)`

引数：`F` 多項式, `VL` 変数リスト, `Ord1` order $<_1$ or $<_2$ or $<_3$, `Ord2` `Ord1` に対応する s も含めた order, `Vect` s 斉次のための重みベクトル,

返り値：`F` の local b 関数

説明：global b 関数の約元を総当たりで local b 関数を計算するアルゴリズム 3.13 を実行する。

実行例：$x^2(x+1)^3$ の local b 関数の計算。(例 3.12)

```
[2598] localb_rr(x^2*(x+1)^3, VL2, A2, A2S, V2);
J :
+x^2+3*x^3+3*x^4+x^5
+2*ss+5*x*ss-x*dx-x^2*dx
+2*ss*s^2+5*x*ss*s-x*s*dx-x^2*dx
+x^2*s^3+3*x^3*s^2+3*x^4*s+x^5
... J の生成元の表示

Gr(J) :
+2*ss+5*x*ss-x*dx-x^2*dx
+x^2+3*x^3+3*x^4+x^5
+2*x^3*dx+7*x^4*ss+3*x^4*dx+4*x^5*ss+x^5*dx
```

```
... J のグレブナ基底の表示

compute global-b. global-b 関数の計算

ok : the remainder of global-b is 0.  global-b 関数を Gr(J) で割った余り

0th challenge
Pick :18*ss^4+45*ss^3+40*ss^2+15*ss+2  global-b 関数
length(GenL):4
6*ss^3+11*ss^2+6*ss+1-->zero      Gr(J) で割った余りが 0 かどうか
6*ss^3+13*ss^2+9*ss+2-->zero
9*ss^3+18*ss^2+11*ss+2-->non-zero
18*ss^3+27*ss^2+13*ss+2-->non-zero
InL :2,NotInL :2

1th challenge
Pick :2*ss^2+3*ss+1   gcd(6*ss^3+11*ss^2+6*ss+1,6*ss^3+13*ss^2+9*ss+2) の表
示
length(GenL):2
ss+1-->non-zero
2*ss+1-->non-zero
InL :0,NotInL :2

local-b 関数
2*ss^2+3*ss+1
```

よって $x^2(x+1)^3$ の local b 関数が $s^2+\frac{3}{2}s+\frac{1}{2}$ であることがわかる。

- `localb_nf(F, VL, Ord1, Ord2, Vect, W, Deg, N)`

引数：`F` 多項式, `VL` 変数リスト, `Ord1` order $<_1$ or $<_2$ or $<_3$, `Ord2` `Ord1` に対応する s も含めた order, `Vect` s 斉次のための重みベクトル, `W` 重みベクトル e, `Deg` 十分大きな数 (global b 関数の次数以上の数), `N` 適当な数 (例えば `Deg` + 1) など

返り値：`F` の local b 関数

説明：正規形の近似を用いた local b 関数を計算するアルゴリズム 4.10 を実行する。

実行例：$x^2(y+1)^2z^2$ の local b 関数の計算。(例 4.11)

```
[1991] localb_nf(x^2*(y+1)^2*z^2, VL3, A3, A3S, V3, W3, 6, 7);
0.02sec(0.07359sec)
J :
+x^2*z^2+2*x^2*y*z^2+x^2*y^2*z^2
-dy+x*dx-y*dy
-x*dx+z*dz
-2*ss+x*dx
-x*dx+z*dz
```

```
+s*dy+y*dy-z*dz
+2*ss*s-z*dz
.... J の生成元の表示
Gr(J) :
-x*dx+z*dz
+dy+y*dy-z*dz
+2*ss-z*dz
-2*y*ss*dy+2*z*ss*dz-z*dy*dz
+x^2*z^2+2*x^2*y*z^2+x^2*y^2*z^2
.... J のグレブナ基底の表示
-------------------------
NF : s のべき乗の Gr(J) による正規形の近似の表示
+1
+1/2*z*dz
+1/4*z^2*dz^2+1/4*z*dz
+1/8*z^3*dz^3+3/8*z^2*dz^2+1/8*z*dz
-3/8*z^3*dz^3-31/16*z^2*dz^2-31/16*z*dz-1/4
+23/32*z^3*dz^3+135/32*z^2*dz^2+157/32*z*dz+3/4
-9/8*z^3*dz^3-447/64*z^2*dz^2-555/64*z*dz-23/16

sol_space(7)  線型空間 L_{6,7} の計算
[ 0 0 0 0 0 0 0 ]
[ 1/4 3/2 13/4 3 1 0 0 ]
[ -3/4 -17/4 -33/4 -23/4 0 1 0 ]
[ 23/16 63/8 231/16 9 0 0 1 ]

sol_space(6)  線型空間 L_{5,7} の計算
[ 0 0 0 0 0 0 ]
[ 1/4 3/2 13/4 3 1 0 ]
[ -3/4 -17/4 -33/4 -23/4 0 1 ]

sol_space(5)  線型空間 L_{4,7} の計算
[ 0 0 0 0 0 ]
[ 1/4 3/2 13/4 3 1 ]

sol_space(4)  線型空間 L_{3,7} の計算
[ 0 0 0 0 ]

BSum: local-b 関数の候補
ss^4+3*ss^3+13/4*ss^2+3/2*ss+1/4
[<2>18<2>20<2>25<2>30<[12,2,8]>+(20)38<19>90<19>103<2>118<2>118<2>117<2>122<2>
122<2>127<2>107<2>112<2>94<2>74<2>65<2>56<2>48<2>39<2>32<2>24<2>15<2>9]
weyl_mora(BSum, G)
```

```
0

local-b 関数
ss^4+3*ss^3+13/4*ss^2+3/2*ss+1/4
```

これより、$x^2(y+1)^2z^2$ の local b 関数は $s^4+3s^3+\frac{13}{4}s^2+\frac{3}{2}s+\frac{1}{4}$ となる。

- `lb1(F), lb2(F), lb3(F)`

引数：`F` 多項式,

返り値：`F` の local b 関数

説明：大阿久の local b 関数を計算するアルゴリズム (Algorithm 2.3) を使い、local b 関数を計算する。 `lb` の後の数字は入力する多項式の変数の数を表す。

実行例：x^3+y^3+xyz の local b 関数の計算。

```
[1549] lb3(x^3+y^3+x*y*z);
comp psi(I):
0.09sec + gc : 0.03sec(0.1837sec)
comp local intersection
0.05sec(0.06129sec)
-9*s^5-54*s^4-128*s^3-150*s^2-87*s-20
[1550]
```

6.7 utility

- `print_poly(P, VL, Ord)`

引数：`P` 表示したい元, `VL` 変数リスト, `Ord` order

説明：`P` を `Ord` の順序で表示する。

実行例：
```
[400] print_poly(1+x^2+(x+1)^3*dx, VL1, A1);
+dx+3*dx*x+3*dx*x^2+dx*x^3+1+x^2
```

参考文献

[1] 大阿久俊則:グレブナ基底と線型偏微分方程式系 (計算代数解析入門), 上智大学数学講究録 ,No. 38, (1994)

[2] 大阿久俊則:計算代数と D 加群, 朝倉書店, (2002)

[3] 広中平祐, 卜部東介:解析空間入門, 朝倉書店, (1981)

[4] Castro,F.: Calculs effectifs pour les idéaux d'opérateurs différentiels, Travaux en Cours 24, (1987), 1 - 19

[5] Granger,M.,Oaku,T.: Mimimal filtered free resolutions and division algorithms for analytic D-modules ,Prépublications du départment de matheématiques, Univ.Angers, No.170 (2003)

[6] Granger,M.,Oaku,T.,Takayama,N.:Tangent cone algorithm for homogenized differential operators, Journal of Symbolic Computation, 39, 417-431 (2005)

[7] Greuel,G.-M., Pfister,G.: Advances and improvements in the theory of standard bases and syzygies, Arch.Math., 66, 163-176 (1995)

[8] Mora,T.:An algorithm to compute the equations of tangent cones, EUROCAM 82, Springer Lecture Notes in Computer Science, (1982)

[9] Nakayama,H.: Algorithms of computing local b-function by an approximate division algorithm, preprint

[10] Noro,M.: An Efficient Modular Algorithm for Computing the Global b- function , Mathematical Software, (2002), 147 - 157

[11] Oaku,T.: An algorithm of computing b-function, Duke Math. J., 87, (1997), no.1 115 - 132

[12] Oaku,T.: Algorithm for the b-function and D-modules asscociated with a polynomial, J.Pure.Appl.Algebra 117/118, (1997), 495 - 518

[13] Oaku,T. , Takayama,N.:Algorithms for D-modules - restriction, tensor product, localization, and local cohomology groups, J.Pure.Appl.Algebra 156, (2001), 267-308

[14] Saito, M., Sturmfels, B., Takayama,N.: Groebner Deformations of Hypergeometric Differential Equations, Springer, (2000)

[15] Kobayashi,H., Furukawa,A., Sasaki,T.: GRÖBNER BASIS OF IDEAL OF CONVERGENT POWER SERIES, RIMS, 581, (1986)

[16] Oaku,T., Shimoyama,T. : A Gröbner basis Method for Modules over Rings of Differential Operators, J.Symbolic Computation 18, 223 - 248, (1994)

[17] Bernstein,I.N. : Modules over a ring of differential operators, Functional Anal. Appl., 5 (1971), 89-101

[18] Bernstein,I.N. : The analytic continuation of generalized functions with respect to a parameter, Functional Anal. Appl., 6 (1972), 273-285

[19] Kashiwara,M. : B-functions and holonomic systems: rationality of roots of b-functions, Invent. Math. 38 (1976-77), 33-53

[20] Giovini,A., Mora,T., Nielsi,G., Robbiano,L., Traverso,C.: "one sugar cube, please" OR Selection strategies in the Buchberger algorithm, Proc. Issac 91, 49-54

[21] Takayama,N. : Kan — A system for doing algebraic analysis by computer, `http://www.math.kobe-u.ac.jp/KAN`

[22] Noro,M. et al: Risa/Asir, `http://www.math.kobe-u.ac.jp/Asir`