数理解析研究所講究録
1138巻 2000年 127-171

グレブナ基底 — 理論, 計算の効率化, 応用

(株) 富士通研究所　野呂　正行(Masayuki Noro) *

1　グレブナ基底

1.1　イデアルおよびグレブナ基底

体 K 上の n 変数多項式環 $R=K[x_1,\cdots,x_n]$ を考える. 以下, $(x_1,\cdots,x_n)$ を X と略記する.

定義 1 R の元 $f_1,\cdots,f_m$ に対し,

$$Id(f_1,\cdots,f_m)=\{\sum_{i=1}^{n}g_if_i|g_i\in R\}$$

を $f_1,\cdots,f_m$ で生成されるイデアルと呼ぶ. $f_1,\cdots,f_m$ を I の生成系あるいは基底と呼ぶ.

定義 2 イデアル I に対し I の K^n における *variety* $V_K(I)$ を

$$V_K(I)=\{a\in K^n|\forall f\in I\quad f(a)=0\}$$

で定義する. 混乱のない場合には $V(I)$ と書く.

系 3

$$Id(f_1,\cdots,f_m)=Id(g_1,\cdots,g_l)\Rightarrow V(f_1,\cdots,f_m)\subset V(g_1,\cdots,g_l)$$

イデアルの基底は一組とは限らないが, 同一のイデアルを生成する基底の共通零点は一致するから, イデアル考える方が, 方程式の解を考える上でより自然であると言える.

例 4 $n=1$ の場合

*noryo@flab.fujitsu.co.jp

$R = K[x]$ は *PID (*単項イデアル環*)* である. すなわち任意のイデアル I はある $f \in R$ により $I = Id(f)$ と書ける. これは, f を基底とすることにより, I に対するメンバシップが,

$$g \in I \Leftrightarrow f|g$$

で判定できることを意味する. I の生成元が幾つか与えられている場合, f は, それらの生成元の *GCD* を求めることで得られる.

一変数の場合, イデアルの生成元は, そのイデアルに属する元のうち, 最も次数の小さいものをとればよかった. これは, 次のようにいいかえられる.

- 多項式の各項を降冪の順に並べたとき, 先頭の項を **頭項** と呼ぶ. この時, 頭項が, イデアルのすべての元の頭項を割り切るような元が生成元となる.

これを多変数に拡張するために, 一変数の場合と同様に, 多変数多項式の項の間に「自然な」全順序を入れる. 以下, 体 K 上の n 変数多項式環 $R = K[x_1, \cdots, x_n]$ を固定して考える. 自然数 $\mathbb{N}$ は, 0 以上の整数を表す.

定義 5 $T = \{x_1^{i_1} \cdots x_n^{i_n} | i_1, \cdots, i_n \in \mathbb{N}\}$ とし, T の元を *term* **(項)** と呼ぶ. この時, T における全順序 $\leq$ が **admissible** であるとは,

1. $1 \leq t$ *for all* $t \in T$
2. $t_1 \leq t_2 \Rightarrow t_1 \cdot s \leq t_2 \cdot s$ *for all* $t_1, t_2, s \in T$

を満たすことを言う.

定義 6 辞書式順序 (lexicographical order; lex)
$x_1^{i_1} \cdots x_n^{i_n} > x_1^{j_1} \cdots x_n^{j_n} \Leftrightarrow$
$\exists m \, s.t. \, i_1 = j_1, \cdots, i_{m-1} = j_{m-1}, i_m > j_m$

この順序は消去法による方程式求解に最も適した形のグレブナ基底を与える. しかし, その直接計算は, 時間, 空間計算量がしばしば極めて大きくなるということから不利である.

定義 7 全次数辞書式順序 (total degree lexicographical order)
$x_1^{i_1} \cdots x_n^{i_n} > x_1^{j_1} \cdots x_n^{j_n} \Leftrightarrow$
$\sum_k i_k > \sum_k j_k$ または
$(\sum_k i_k = \sum_k j_k$ かつ $\exists m \, s.t. \, i_1 = j_1, \cdots, i_{m-1} = j_{m-1}, i_m > j_m)$

入力が斉次の場合, この順序のもとでのグレブナ基底と辞書式順序によるグレブナ基底は一致する. degree compatible order による計算は, そうでない order による計算に比べ

て効率がよいことが経験的に知られており, 斉次な場合にこの順序で辞書式順序グレブナ基底を計算する, あるいは非斉次な入力を斉次化して, この順序でグレブナ基底を計算し, 非斉次化することでもとの入力の辞書式順序グレブナ基底を求めるということが行われる.

定義 8 全次数逆辞書式順序 (total degree reverse lexicographical order; drl)
$x_1^{i_1}\cdots x_n^{i_n} > x_1^{j_1}\cdots x_n^{j_n} \Leftrightarrow$
$\sum_k i_k > \sum_k j_k$ または
$(\sum_k i_k = \sum_k j_k$ かつ $\exists m\, s.t.\, i_n = j_n, \cdots, i_{m+1} = j_{m+1}, i_m < j_m)$

一般に, 最も高速にグレブナ基底を計算できるが, グレブナ基底の個々の元の持つ性質は掴みにくく, 連立方程式を直接解くことは困難である. しかし, 次元, Hilbert function その他の不変量を計算する場合など, グレブナ基底という性質のみが必要とされる場合に, 高速に計算できるという特性を生かして用いられる場合が多い. また, 後に述べる基底変換の入力として用いられることも多い.

定義 9 block order
$\{x_1, \cdots, x_n\} = S_1 \cup \cdots \cup S_l$ *(disjoint sum)* とし, $<_i$ を $T_i = K[y_1, \cdots]$ $(y_k \in S_l)$ 上の *admissible order* とする. このとき, T 上の *order* を, $<_i$ の *order* を順に適用して決める.

辞書式順序は $\{x_1, \cdots, x_n\} = \{x_1\} \cup \cdots \cup \{x_n\}$ なる分割による block order であるが, 単に, 幾つかの変数を消去した結果を求めたい場合には, 効率を考えれば問題がある. このような場合に, S_1 に消去したい変数, S_2 に残りの変数, と分割し, それぞれに対し, 例えば drl order を設定することで S_1 に属する変数を消去できる. これについては後で述べる.

定義 10 matrix order
M を 次を満たす実 $m \times n$ 行列とする.

1. 長さ n の整数ベクトル v に対し, $Mv = 0 \Leftrightarrow v = 0$
2. 非負成分を持つ長さ n の整数ベクトル v に対し, Mv の 0 でない最初の成分は正.

この時, $\mathbb{N}^n$ のベクトル u, v に対し,

$$u > v \Leftrightarrow M(u-v) \text{ の } 0 \text{ でない最初の成分が正}$$

で定義すれば, この *order* は *admissible order* となる. これを, M により定義される *matrix order* と呼ぶ.

命題 11 *(Robbiano [[5]])* 任意の *admissible order* は, *matrix order* により定義できる.

例 12 よく知られた *order* を定義する *matrix* の例

$$M_{dlex} = \begin{pmatrix} 1 & \cdots & 1 \\ 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix} \quad M_{drl} = \begin{pmatrix} 1 & \cdots & 1 \\ 0 & & -1 \\ & \iddots & \\ -1 & & 0 \end{pmatrix} \quad M_{lex} = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}$$

M_{dlex}, M_{drl}, M_{lex} はそれぞれ全次数辞書式, 全次数逆辞書式, 辞書式順序を定義する.

例 13 *weighted order*

$$M_{wdrl} = \begin{pmatrix} w_1 & \cdots & w_n \\ 0 & & -1 \\ & \iddots & \\ -1 & & 0 \end{pmatrix}$$

第一行は, 指数ベクトル $(d_1, \cdots, d_n)$ に対して, $\sum_{i=1}^{n} w_i d_i$ すなわち *weight* 付きの全次数で最初に比較を行うことを意味する.

例 14 *block order*

$$M_{block} = \begin{pmatrix} M_1 & & 0 \\ & \ddots & \\ 0 & & M_l \end{pmatrix}$$

各 M_k は, 各ブロックに対する *admissible order* を定義する *matrix* である.

定義 15 指数を $\mathbb{N}^n$ の元と考えて, $\mathbb{N}^n$ における *admissible order* $<$ を

1. $0 = (0, \cdots, 0) \leq \alpha$ *for all* $\alpha \in \mathbb{N}^n$
2. $\alpha_1 \leq \alpha_2 \Rightarrow \alpha_1 + s \leq \alpha_2 + s$ *for all* $\alpha_1, \alpha_2, s \in \mathbb{N}$

を満たすものとして定義できる.

定義 16 $L \subset \mathbb{N}^n$ がモノイデアルとは,

$$\text{任意の}\alpha \in L, \beta \in \mathbb{N}^n \text{に対し} \alpha + \beta \in L$$

が成り立つことをいう. また, $S \subset \mathbb{N}^n$ に対し,

$$mono(S) = \{\alpha + \beta | \alpha \in S, \beta \in \mathbb{N}^n\}$$

を S で生成されるモノイデアルと呼ぶ.

定義 17 *term* 間の全順序を多項式の半順序に自然に拡張する.
$f, g \in R$ で, $f = \sum_{i \geq 0} c_i f_i$, $g = \sum_{i \geq 0} d_i g_i$ ($c_i, d_i \in K, f_i, g_i \in T, i > j \Rightarrow f_i > f_j, g_i > g_j$) とする時, $f > g$ を

$$f > g \Leftrightarrow \exists i_0 \, s.t. \, (i < i_0 \Rightarrow f_i = g_i, f_{i_0} > g_{i_0})$$

で定義する.

定義 18 $M = \{c \cdot t | c \in K, t \in T\}$ とし, M の元を **monomial** と呼ぶ.

定義 19 *admissible order* を一つ固定した時, 多項式 f に表れる *term* の中で, その *order* において最大のものを **頭項 (head term)** と呼び, $HT(f)$ と書く.
$HT(f)$ の係数を, $HC(f)$ と書く.
$HC(f) \cdot HT(f)$ を $HM(f)$ と書く.
$HT(f)$ の指数を $HE(f)$ と書く. $HE(f) \in \mathbb{N}^n$ である.
さらに, $f - HM(f)$ を $red(f)$ (**reductum** *of* f) と書く.

補題 20 $\mathbb{N}^n$ の任意のモノイデアル L は有限生成.

定義 21 $S \subset R$ および *admissible order* $<$ に対し,

$$E_<(S) = \{HE(f) | f \in S\} \subset \mathbb{N}^n$$

と定義する. 以下混乱のない場合には $<$ を省略して $E(S)$ と書く.

補題 22 イデアル I に対し, $E(I)$ はモノイデアル.

一変数多項式環におけるイデアル I の生成系 $G = \{f\}$ の性質は,

$$E(I) = mono(E(G))$$

と書くことができる. 一般の場合にもこのような性質を満たす G を考えることは有用である.

定義 23 イデアル I に対し, その有限部分集合 G で,

$$E(I) = mono(E(G))$$

を満たすものを I の $<$ に関するグレブナ基底と呼ぶ.

この定義は, イデアルのすべての元の頭項が, G のいずれかの元の頭項で割り切れることを意味する.

命題 24 任意の *admissible order* $<$ に関し, イデアル I のグレブナ基底は存在する.

命題 25 イデアル I の グレブナ基底 G は I を生成する.

定義 26 $f, g \in R$ とし, f に現れるある *monomial* m が $HT(g)$ で割り切れるとする. このとき f は g で **簡約可能 (reducible)** であるという. このとき $h = f - m/HT(g) \cdot g$ に対し, $f \underset{g}{\rightarrow} h$ と書く.
$G \subset R$ についても, G のある元により f が h に簡約されるとき $f \underset{G}{\rightarrow} h$ と書く. さらに, G による簡約を 0 回以上繰り返して h が得られるとき, $f \underset{G}{\overset{*}{\rightarrow}} h$ と書く.
f のどの項も, G で簡約できないとき, f は G に関して **正規形 (normal form)** であるという.

命題 27 イデアル I の グレブナ基底 G について, 以下のことが成り立つ.

1. $f \in I \Leftrightarrow f \underset{G}{\overset{*}{\rightarrow}} 0$
2. $f \underset{G}{\overset{*}{\rightarrow}} f_1, f \underset{G}{\overset{*}{\rightarrow}} f_2$ かつ f_1, f_2 が正規形 $\Rightarrow f_1 = f_2$
3. $f \in G, \exists h \in G\, s.t.\, HT(h) | HT(f) \Rightarrow G \setminus \{f\}$ は I のグレブナ基底

系 28 $G = \{g_1, \cdots, g_l\}$ をイデアル I の グレブナ基底とする. このとき, $\exists H \subset G\, s.t.\, H$ は I の グレブナ基底かつ $HT(g_i)(g_i \in H)$ のどの二つも互いに他を割らない.

この系により, 冗長な元をすべて除去した グレブナ基底に対し, 各元を, 基底の他の元に対して正規形となるよう簡約を行ない, 頭項の係数を 1 となるようにした基底を被約 **(reduced)** グレブナ基底 と呼ぶ. これが実際に元のイデアルの グレブナ基底になっていることは, 頭項が変わっていないことよりわかる. さらに次の命題は, 定義よりただちに得られる.

命題 29 被約グレブナ基底は集合として一意的に定まる.

以上が グレブナ基底の定義からただちに導かれる基本的な性質であるが, 実際に グレブナ基底を構成するアルゴリズムを得るためには, グレブナ基底の定義の言いかえをいくつか行なう必要がある.

定義 30 $G=\{g_1,\cdots,g_l\}$ を *(グレブナ基底とは限らない)* R の有限部分集合とする. これに対し, 写像 d_1 を次で定義する.

$$\begin{array}{lll} d_1: & R^l & \longrightarrow R \\ & (f_1,\cdots,f_l) & \longmapsto \sum f_i\cdot HT(g_i) \end{array}$$

定義 31 $f=(f_1,\cdots,f_l)\in R^l$ が T-斉次 とは, ある *term* t が存在して, すべての i に対し, $f_i=0$ または $t=f_i\cdot HT(g_i)$ と書けるときを言う.

定義 32 $e_i\in R^l$ を $e_i=(0,\cdots,1,\cdots,0)$ *(第 i 成分のみ 1)* と定義する.
$i_1,\cdots,i_k$ に対し, $T_{i_1,\cdots,i_k}$ を

$$T_{i_1,\cdots,i_k}=LCM(HT(g_{i_1}),\cdots,HT(g_{i_k}))$$

と定義する. 特に, $T_i=HT(g_i)$ である.

命題 33 イデアル I について, 次は同値.

1. $G=\{g_1,\cdots,g_l\}$ は I のグレブナ基底
2. $f\in I\Leftrightarrow f\xrightarrow[G]{*}0$
3. $f\in I\Leftrightarrow \exists f_i(i=1,\cdots,l)\,s.t.\,f=\sum_i f_ig_i$かつ $HT(f_ig_i)\le HT(f)$
4. L を T-斉次 な $Ker(d_1)$ の基底とするとき, 任意の $h=(h_1,\cdots,h_l)\in L$ に対し, $\sum h_i\cdot g_i\xrightarrow[G]{*}0$

命題 34 $G=\{g_1,\cdots,g_l\}$ を $HC(g_i)=1$ なる R の有限部分集合とする. $i,j\in\{1,\cdots,l\}$ に対し, $S_{ij}\in R^l$ を $S_{ij}=T_{ij}/T_ie_i-T_{ij}/T_je_j$ で定義する. この時, $L=\{S_{ij}|i<j\}$ は $Ker(d_1)$ の T-斉次 な基底となる. この基底を **Taylor 基底** と呼ぶ.

定義 35 $f,g\in R$ に対し, **S 多項式** $Sp(f,g)$ を,

$$Sp(f,g)=\frac{HC(g)T_{fg}}{HT(f)}\cdot f-\frac{HC(f)T_{fg}}{HT(g)}\cdot g$$

($T_{fg}=LCM(HT(f),HT(g))$) と定義する.

以上により, 新たな グレブナ基底の定義が得られる.

命題 36 イデアル I について, 次は同値.

1. $G=\{g_1,\cdots,g_l\}$ は I のグレブナ基底
2. 任意の対 $\{f,g\}(f,g\in G;f\neq g)$ に対し, $Sp(f,g)\xrightarrow[G]{*}0$

1.2 Buchberger アルゴリズム

前節の最後の命題により, 次のアルゴリズムが導かれる.

アルゴリズム **37 (Buchberger**[4]**)**
入力 : R の有限部分集合 $F = f_1, \cdots, f_l$
出力 : F で生成されるイデアルの グレブナ基底 G
$D \leftarrow \{\{f, g\} | f, g \in F; f \neq g\}$
$G \leftarrow F$
while ($D \neq \emptyset$) *do* {
　　$\{f, g\} \leftarrow D$ の元
　　$D \leftarrow D \setminus \{C\}$
　　$h \leftarrow Sp(f, g)$ の正規形の一つ
　　if $h \neq 0$ *then* {
　　　$D \leftarrow D \cup \{\{f, h\} | f \in G\}$
　　　$G \leftarrow G \cup \{h\}$
　　}
}
return G

以下で, D の元を対 (pair) と呼ぶことにする.

定理 38 アルゴリズム *37* は停止し, グレブナ基底を出力する.

このアルゴリズムが Buchberger アルゴリズムの最も原始的な形であるが,

- 正規形が 0 でない場合, D の要素が G の要素の個数だけ増加する.
- D から一つ元を選ぶ方法が明示されていない.

などの点で実用的でない. 実際に計算機上にインプリメントする場合, 上記二点に関して工夫をする必要がある.

2 グレブナ基底計算の効率化

前節で, グレブナ基底を計算するアルゴリズムである Buchberger アルゴリズムの最も原始的な形を示した. しかし, そこで述べたように, アルゴリズム 37 をそのまま実行することは, 効率の点からみて問題がある. これは, アルゴリズムの進行につれて, 正規化して 0 になる対が増加していくためである. また, 対の選び方にも任意性がある. どのような選

び方を用いても, 最終結果であるグレブナ基底の一意性は保証されているが, 選び方により効率に大きな差がでることが多くの実験により確かめられている.

本節では, グレブナ基底の計算を効率化するさまざまな方法を紹介する.

記法 39 以下で, 次のような記法を用いる.

p : 素数.

$GF(p)$: p 元体.

$\mathbb{Z}_{(p)} = \{a/b | a \in \mathbb{Z}; b \in \mathbb{Z} \setminus p\mathbb{Z}\} \subset \mathbb{Q}$: $\mathbb{Z}$ の $p\mathbb{Z}$ における局所化.

$X = \{x_1, \cdots, x_n\}$: 不定元.

ϕ_p : $\mathbb{Z}_{(p)}[X]$ から $GF(p)[X]$ への標準的射影. $\phi_p(a/b) = \phi_p(a)/\phi_p(b)$. $(a \in \mathbb{Z},\ b \in \mathbb{Z} \setminus p\mathbb{Z}.)$

$<, <_0, <_1, <_i$: *admissible order.*

$HT_<(f)$: f の $<$ に関する頭項.

$HC_<(f)$: f の $<$ に関する頭係数.

$T(f)$: f に現れる項全体の集合.

$GB_<(S)$: S の $<$ に関する被約グレブナ基底.

$f_1, \cdots, f_m$: $\mathbb{Z}[X]$ の元.

$NF_<(f, G)$: f の G に関する正規形の一つ.

$Init_<(I)$: $\{HT_<(f) | f \in I\}$ で生成されるイデアル.

$Useless\_Pairs(D)$: 不必要対の検出.

$Select\_Pair(D)$: 対の選択

$Spoly(C)$: 対の S-多項式の計算

2.1 不必要対の検出

Buchberger アルゴリズムにおいて, 正規化により 0 になる対を不必要対と呼ぶ. これらは, ある多項式集合が グレブナ基底であることの確認のためにのみ意味があり, もし計算せずに 0 に正規化されることがわかれば計算効率の点からみて極めて好都合である. さらに, この 「0 への正規化」は, アルゴリズムのある時点においてではなく, グレブナ基底のすべての元が生成されたのち, 0 に正規化されるのであってもよい. 実際, そのような状況は, 命題 33 において現れている. この命題を, Taylor 基底に適用すれば, Taylor 基底のうち, 冗長な (すなわち他の基底により生成される) 基底をとり除いた残りについて 0 に正規化されればよいことになる. ここで取り除かれる基底に対応する多項式対のほか, 他の方法により 0 に正規化されることがわかり, とり除かれる対もあり得る. これらについて述べる.

2.1.1 冗長な基底の除去

Taylor 基底から冗長な基底を取り除くための基本的な道具は, 基底間の次の自明な恒等式である.

$$\frac{T_{ijk}}{T_{ij}}S_{ij} + \frac{T_{ijk}}{T_{jk}}S_{jk} + \frac{T_{ijk}}{T_{ki}}S_{ki} = 0$$

この恒等式において, いずれかの係数が 1 であれば, その項に対応する基底は, 他の基底により生成される. これを繰り返して冗長な基底を取り除いて行くが, この際, 相互に依存し合う基底を誤って取り除くことを避けるため, 基底間に全順序を入れ, ある基底がそれより順序の低い基底により生成されている場合にのみ, その基底を取り除くこととする.

定義 40 $\{S_{ij}|i<j\}$ における全順序を次のように定義する.
$S_{ij} < S_{kl} \Leftrightarrow$
$T_{ij} < T_{kl}$ または
$(T_{ij} = T_{kl}$ かつ $(j < l$ または $(j = l$ かつ $i < k)))$

定義 41 対 (i,j) に関する三つの性質を次のように定義する.
$M(i,j) \Leftrightarrow \exists k < j\, s.t.\, T_{kj}|T_{ij}$ かつ $T_{kj} \neq T_{ij}$
$F(i,j) \Leftrightarrow \exists k < i\, s.t.\, T_{kj} = T_{ij}$
$B(i,j) \Leftrightarrow \exists k > j\, s.t.\, T_k|T_{ij}$ かつ $T_{jk} \neq T_{ij}$ かつ $T_{ik} \neq T_{ij}$

これらは, 前記恒等式において, S_{ij} の係数が 1 かつ S_{ij} の順序が最大になる条件を場合分けにより表したものである. いずれの性質においても $T_k|T_{ij}$ より S_{ij} の係数は 1. それぞれについて確かめれば, 実際に S_{ij} が最大順序となっていることがわかる. よって, 次の命題がなり立つ.

命題 42 (Gebauer and Möller[6]) *Taylor* 基底は, $\{S_{ij}|\neg M(i,j), \neg F(i,j), \neg B(i,j)\}$ で生成される.

注意 43 実際のアルゴリズムにおいては, *0* でない正規形が生成されて, D に対を追加する際に, 上記の性質を満たす対を削除していくが, 性質 M, F, B のうち, M, F は, 多項式集合 G に追加する元を含む対が対象となるのに対し, B は既に存在している対が対象となる. この事実は, 正規化対の選択戦略との関連で, アルゴリズムの進行に重大な影響を及ぼす場合がある.

2.1.2 0 に正規化される対の除去

前節で述べたもの以外に, 頭項のみにより, 0 に正規化できると判断できる対があり得る.

命題 44 (Buchberger)
$GCD(HT(f), HT(g)) = 1$ ならば $Sp(f,g) \xrightarrow[\{f,g\}]{*} 0$

この命題により, 前節の操作で残った基底の中からさらに 0 に正規化される対を除去することができる.

2.2 正規化対の選択戦略

アルゴリズムのループにおける, 正規化の対象となる対の選択は, アルゴリズムの正当性にはなんら影響しないが, 実際の計算効率に大きく影響を与える. Buchberger により提案された戦略は次のものである.

定義 45 normal strategy
T_{ij} が最小な対から選択する戦略をいう.

この戦略は, 順序の低い頭項をもつ基底を早めに生成するという実用的な意味をもつ他に, この戦略によれば S 多項式は, 簡約の仕方によらず一意的な正規形となることが示されている. この戦略は, term order によっては極めて悪い結果をもたらす場合が多く, 特に, 辞書式順序のように, 全次数による比較を行わない順序で甚だしい.

定義 46 斉次化
t を斉次化変数とし, $R = k[x_1, \cdots, x_n]$ *and* $R_h = k[x_1, \cdots, x_n, t]$ とする. $f \in R$ の斉次化を f^*, $g \in R_h$ の非斉次化を g_* と書く. 以下, f の全次数を $tdeg(f)$ と書く.

$$f^* = t^{tdeg(f)} f(x_1/t, \cdots, x_n/t)$$

$$g_* = g|_{t=1}$$

である. 多項式集合 $F \subset R$, $G \subset R_h$ に対しても, 各元を斉次化, 非斉次化したものをそれぞれ F^*, F_* と書く.

このとき, R の *order* $<$ に対して, R_h の *order* $<_h$ が

(H) 任意の斉次多項式 $g \in R_h$ に対し, $HT(g)_* = HT(g_*)$

を満たすとき, $<_h$ を $<$ の斉次化と呼ぶことにする.

例 **47** $X \cup \{t\}$ (X に関して $<$) による *block order* は $<$ の斉次化の一つであるが, 定義により,

$$ut^m <_h vt^n \Leftrightarrow tdeg(ut^m) < tdeg(vt^n) \text{ または } tdeg(ut^m) = tdeg(vt^n) \text{ かつ } u < v$$

なる $<_h$ も $<$ の斉次化である.

命題 48 $F \subset R$ とし, $<_h$ を $<$ の斉次化 *order* とする. このとき, もし G が $Id(F^*)$ の斉次多項式からなるグレブナ基底なら, G_* は $Id(F)$ のグレブナ基底となる.

斉次化は選択戦略そのものではないが, 入力を斉次化した後, 上の例で述べた全次数比較入りの $<_h$ のもとで normal strategy で計算した後非斉次化するという手順でグレブナ基底を求めると, 不必要対は増加するものの, 計算がスムーズに進行することが経験的に知られていた. 係数体が有限体の場合には, 後で述べる, 不必要対に対する正規形計算で生じる係数膨張の問題が生じないため, 有限体上でのグレブナ基底計算には斉次化は有用である.

定義 49 sugar strategy
グレブナ基底計算の途中に現れる多項式 f に, 次の規則によりある自然数 s_f *(sugar)* を対応させる.

1. 入力多項式 f に対しては, $s_f = tdeg(f)$
2. m が *monomial* の時 $s_{mf} = tdeg(m) + s_f$
3. $s_{f+g} = MAX(s_f, s_g)$

sugar strategy とは, S 多項式の *sugar* の最も小さいものの集合から *normal strategy* で対を選択する戦略をいう.

Giovini ら [8] により提案されたこの戦略は, 入力多項式集合を斉次化してのグレブナ基底計算を仮想的に行なうもので, sugar は斉次化後の次数に対応している.

2.3 trace lifting

sugar strategy により, 少なくとも係数体が有限体の場合には, 十分な効率が達成できるようになった. しかし, 係数体が有理数の場合, 前に述べた検出基準にかからない不必要対の S 多項式の正規化計算にかかるコストが大きい場合がしばしばある. そのため, 次のようなアルゴリズム (**trace lifting**[1)]) が提案された [7].

[1)]提案者に従ってこう呼ぶが, この呼称はあまり適当でないと思う. 特に lifting という用語は, Hensel lifting を想像させ, 誤解を招く.

アルゴリズム **50**

trace_lifting$(F, <)$

入力 : $F \subset \mathbb{Z}[X]$; *admissible order* $<$

出力 : F の $<$ に関するグレブナ基底

do {

$p \leftarrow$ 未使用の素数

$G \leftarrow tl\_guess(F, <, p)$

if $G \neq nil$

かつ G が $<$ に関するグレブナ基底

かつ $\forall f \in F \ NF(f, G) = 0$

then return G

}

アルゴリズム **51**

tl_guess$(F, <, p)$

入力 : $F \subset \mathbb{Z}[X]$; *admissible order* $<$; 素数 p

出力 : F のグレブナ基底候補または **nil**

if $\exists f \in F \quad s.t. \ \phi_p(HC(f)) = 0$ *then return* **nil**

$G \leftarrow F$

$G_p \leftarrow \{\phi_p(f) | f \in G\}$

$D \leftarrow \{\{f, g\} | f, g \in G; f \neq g\}$

do {

$D \leftarrow D \setminus Useless\_Pairs(D)$

if $D = \emptyset$ *then return* G

else { $C \leftarrow Select\_Pair(D)$

$D \leftarrow D \setminus \{C\}$

}

$s \leftarrow Spoly(C)$

$t_p = NF(\phi_p(s), G_p)$

if $t_p \neq 0$ *then* {

$t \leftarrow NF(s, G)$

if $\phi_p(HC(f)) = 0$ *then return* **nil**

$D \leftarrow D \cup \{\{f, t\} | f \in G\}$

$G \leftarrow G \cup \{t\}$

$G_p \leftarrow G_p \cup \{t_p\}$

}

}

アルゴリズム 51 は, 有理数体上での正規形計算を行なう前に有限体上で正規形を計算し, それが 0 となる場合には, 有理数体上の正規形も 0 であると仮定してその計算を省略する. このような操作を行なっても, 生成される 0 でない多項式について, その頭項は, 通常の Buchberger アルゴリズムと同様の性質を持つためアルゴリズムは停止する. しかし, p の選び方によってはグレブナ基底を出力しない場合があるため, 次のようなテストが必要となる.

1. 候補 G があるグレブナ基底となっていること.
2. 入力多項式 F の各元が, あるグレブナ基底である候補 G により 0 に正規化されること.

実際, $Id(G) \subset Id(F)$ は構成法より常になりたつから, 1., 2. が成り立てば $Id(F) \subset Id(G)$ より $Id(G) = Id(F)$ で, G は $Id(F)$ のグレブナ基底となる.

このテストを組み合わせたものが, アルゴリズム 50 である. このテストを通過できるような p としては, 通常の Buchberger アルゴリズムを実行した場合に現れるすべての多項式の頭係数を割らないような素数をとればよい. もちろんこのような素数はあらかじめ決定することはできないが, 不都合な素数は有限個であり, アルゴリズム全体としての停止性も保証される.

2.4 斉次化 と trace lifting の組合せ

trace lifting により, 不必要対に対する S 多項式の計算コストを, 有限体上のグレブナ基底計算および, グレブナ基底候補生成後のグレブナ基底テストと入力多項式のメンバシップテストに置き換えることができた. しかし, 実際に斉次化した場合には生じないような中間係数膨張が, sugar strategy の適用により生じる場合がしばしば見られる. 本節では, この困難を克服するための方法について述べる.

このような係数膨張を簡単な例で示す.

例 52 $I = Id(-3x_1 - 3x_2 - 5, -3x_1x_0^2 - 8x_0 - 6, 4x_0^2 + (-2x_1^3 - 7)x_0 - 3x_3x_1^2 - 2, -x_0^4 - x_0^2 + 3x_1x_0 - 4x_1)$

I の $x_0 > x_1 > x_2$ なる辞書式順序のもとでのグレブナ基底 $GB(I)$ は,

$GB(I) = \{f_3(x_3), f_2(x_2, x_3), f_1(x_1, x_3), f_0(x_0, x_3)\}$

$f_3 = 286978140000x_3^6 - 30735638450064x_3^5 + 1393681087737 18x_3^4 - 401122901874078x_3^3$

$+ 3813285736741284x_3^2 - 17712668879937657x_3 + 25309718023001309$

$$f_2 = -24359129516408255978429894458178435051554483918248x_2$$
$$- 91173387252004471831217734836181460114173800000x_3^5$$
$$+ 94543724248166839034890914739396232954660361748 8x_3^4$$
$$- 12151997137046789022734072995370860483918763507 46x_3^3$$
$$+ 92038914277032211724991748423074087630602785195 44x_3^2$$
$$- 87128026603776953223695902278763008965980027109460x_3$$
$$+ 19807212871855034606976039002295220479435694630719 5$$
$$f_1 = 24359129516408255978429894458178435051554483918248x_1$$
$$- 91173387252004471831217734836181460114173800000x_3^5$$
$$+ 94543724248166839034890914739396232954660361748 8x_3^4$$
$$- 12151997137046789022734072995370860483918763507 46x_3^3$$
$$+ 92038914277032211724991748423074087630602785195 44x_3^2$$
$$- 87128026603776953223695902278763008965980027109460x_3$$
$$+ 23867067791256410603381021411991626321361441950427 5$$
$$f_0 = -91346735686530959919112104218169131443329314693 43x_0$$
$$- 33444249414397078848103849201574810848200000 0x_3^5$$
$$+ 37671432710327352302696037580172208903376423200x_3^4$$
$$- 35235604916430753666624787497338663840087253804 0x_3^3$$
$$+ 49996927085598535116090951898965322091760877126 2x_3^2$$
$$- 68416189150670571465095231265107250942388567714 32x_3$$
$$+ 31588951614319486540726259755101613855437086625238$$

この計算を, 斉次化を経由して行った場合 P6-200MHz 上では 0.2 秒弱で計算終了し, 現れる基底の係数のビット長の和の最大値は 1489 ビットだが, sugar strategy のみを用いて行った場合その最大値は 10258121 ビットとなる. すなわち, このような小さい問題に対しても, sugar strategy がうまく働かない場合には不必要な係数膨張が生じてしまう. 斉次化は生成される基底の係数膨張が起こらないような strategy を与える場合が多いが, 非斉次の場合に比べて前に述べた不必要対の検出にかからない不必要対を多く生成し, 問題が大きくなると, それらの 0 への正規化のコストが非常に大きくなる. これらを克服する有力な方法が, 次のアルゴリズムである.

アルゴリズム **53**

homogenized_trace_lifting$(F, <)$

入力 : $F \subset \mathbb{Z}[X]$; *admissible order* $<$

出力 : F の $<$ に関するグレブナ基底

do {

$p \leftarrow$ 未使用の素数

$G \leftarrow tl\_guess(F^*, <_h, p)$
if $G \neq nil$
　かつ G_* が $<$ に関するグレブナ基底
　かつ $\forall f \in F\ NF(f, G_*) = 0$
then return G_*
}

$<_h$ は 例 47 で述べた order の全次数付斉次化を用いている. この方法と, 単に入力を斉次化したものに対して trace lifting によりグレブナ基底を求めてから非斉次化する方法を比較すると次のようになる. いずれも, 斉次化により選択戦略を安定にし, かつ斉次化により増加した不必要対に関するコストは trace lifting により緩和される.

- 斉次化 ⇒ 候補生成 ⇒ 非斉次化 ⇒ テスト

  非斉次化を行なった時点で冗長な基底を取り除くため, テストにかかるコストは斉次化を行なわない場合と変わらない.

- 斉次化 ⇒ 候補生成 ⇒ テスト ⇒ 非斉次化

  斉次なグレブナ基底候補の場合, 冗長な基底がほとんどなく, trace lifting によりカットされた大部分の不必要対を有理数体上で改めて正規化する必要がある.

すなわち, 入力が既に斉次の場合を除いて, 非斉次化後にテストを行なう方が効率がよい. この方法により, trace lifting のみでは計算不可能だった多くの問題が計算可能になった.

3 change of ordering

前節では, 主として Buchberger アルゴリズムの効率化について述べた. しかし, グレブナ基底の計算法は Buchberger アルゴリズムだけとは限らない. 本節では, 既に何らかの order に関してグレブナ基底になっている多項式集合を入力として, 他の order のグレブナ基底を求める方法について述べる.

3.1 FGLM アルゴリズム

$I \subset K[X]$ を 0 次元イデアルとし, I のある order $<_1$ に関する被約グレブナ基底 G_1 が既に得られているとする. このとき, 他の order $<$ に関する I のグレブナ基底 G を, 主として線形代数により求めるのが FGLM アルゴリズムである.

補題 54 $<$ を *admissible order*, $F = GB_<(I)$ とする. $T = \{t_1, \cdots, t_l\} \subset T(X)$ を項の集合とする. a_i を未定係数とし,

$$E = \sum_{i=1}^{l} a_i NF_<(t_i, F)$$

とおく. E の X に関する係数の集合を C とすれば, $Eq = \{f = 0 | f \in C\}$ は a_i に関する線形方程式となる. この時

$$Eq \text{ が自明でない解を持つ} \Leftrightarrow T \text{ が } K[X]/I \text{ において } K\text{-線形従属}$$

アルゴリズム **55** *(FGLM アルゴリズム)[9]*

$FGLM(F, <_1, <)$
Input : $order <_1, <; F \subset K[X]$ *s.t.* $F = GB_{<_1}(I)$ かつ $dim(I) = 0$
Output : F の $<$ に関するグレブナ基底
$G \leftarrow \emptyset$
$h \leftarrow 1$
$B \leftarrow \{h\}$
$H \leftarrow \emptyset$
do {
  $N \leftarrow \{u | u > h$ かつ $\forall m \in H\, m \not| u\}$
(0) *if* $N = \emptyset$ *then return* G
(1) $h_1 \leftarrow MIN(N)$
  $a_t : t \in B$ に対応する未定係数
  $a_{h_1} \leftarrow 1$
(2) $E \leftarrow NF_{<_1}(h_1, F) + \sum_{t \in B} a_t NF_{<_1}(t, F)$
  $C \leftarrow E$ の X に関する係数の集合
  if 線形方程式 $\{f = 0 | f \in C\}$ が解 $\{a_t = c_t | c_t \in K\}$ を持つ
  then
    $G \leftarrow G \cup \{h_1 + \sum_{t \in B} c_t t\}$
    $H \leftarrow H \cup \{h_1\}$
  else $B \leftarrow \{h_1\} \cup B$
  $h \leftarrow h_1$
}

命題 56 アルゴリズム *55* は $GB_<(F)$ を出力する.

FGLM アルゴリズムを計算機上で実装する場合, 特に (2) の部分の実装に工夫が必要となる. 要点をまとめると,

1. 正規形の計算は, 各項につきただ一度だけ行い, 結果は表にして保持する.

2. 毎回独立な線形方程式として解くのではなく, 結果が後で使えるような工夫をする.

1. に関連して, 次のような線形写像を考えることで, 正規形計算の効率を上げることができる.

定義 57 各 $i(1 \leq i \leq n)$ に対し, $\phi_i \in End(K[X]/I)$ を

$$\phi_i : f \bmod I \mapsto x_i f \bmod I$$

で定義する. $H_1 = \{HT_{<_1}(g) | g \in G_1\}$, $MB_1 = \{u \in T | \forall m \in H_1\, m \not| u\}$ とおけば MB_1 は $K[X]/I$ の K-基底より, $\{NF_{<_1}(x_i u, G_1) | u \in MB_1\}$ を全て計算することで, ϕ_i が表現できる.

あらかじめ, ϕ_i を計算しておけば, $NF(x_i t, G_1) = \phi_i(NF(t, G_1)$ より, 既に得られているはずの $NF(t, G_1)$ の像としてあらたな項の正規形が計算できる.

注意 58 一般には, *FGLM* アルゴリズムは *0* 次元イデアルの場合にのみ適用可能だが, 目的の *order* が全次数比較を含む場合など, 任意の $s \in T$ に対し $\{t \in T | t < s\}$ が有限集合の場合には, 任意のイデアルに適用できる. しかし, 効率は一般に *0* 次元の場合に比べて期待できない.

3.2 modular change of ordering

FGLM は, 目的の項に到達するまで行列の Gauss 消去を繰り返す方法といえる. この Gauss 消去は有理数体上で行なわれるため, 結果のグレブナ基底の元の係数に比べて途中の係数膨張が激しくなる場合がしばしば生ずる. これは, 次の例で示される.

例 59 $A \in GL(n, \mathbb{Q})$ とする. $V \in \mathbb{Q}^n$ に対し $B = AV$ とすると, 線形方程式 $AX = B$ は $X = V$ を唯一の解とする. この方程式を *Gauss* 消去で解く場合, それは A にのみ注目して行なわれ, B の値に左右されない. すなわち, 解 V の成分が小さい整数の場合でも大きい場合でも解く手間は変わらないことになる.

3.2.1 modular 計算と線形代数によるグレブナ基底候補生成

ここで紹介するアルゴリズムは, modular 計算を応用して, 結果の係数の大きさの程度のコストでグレブナ基底を計算するものである. アルゴリズム 60 では, 有限体上のグレブナ基底計算により, 有理数体上のグレブナ基底の各元に現れる項を推測し, 未定係数法で, それらの項を実際にもつ $I = Id(F)$ の元を求める.

アルゴリズム **60**
candidate_by_linear_algebra$(F, p, <_1, <)$
Input : *order* $<_1$, $<$
　　$F \subset \mathbb{Z}[X]$　*s.t.*　$F = GB_{<_1}(Id(F))$
　　F の各元の $<_1$ に関する主係数を割らない p
Output : F の $<$ に関するグレブナ基底候補または **nil**
$\overline{G} \leftarrow GB_<(Id(\phi_p(F)))$ *(*被約グレブナ基底*)*
$G \leftarrow \emptyset$
for each $h \in \overline{G}$ *do* {
　a_t : $t \in T(h)$ に対応する未定係数
　$a_t \leftarrow 1$ *(*$t = ht_<(h)$ に対して*)*
　$H \leftarrow \sum_{t \in T(h)} a_t NF_{<_1}(t, F)$
　$C \leftarrow H$ の X に関する係数の集合
　if 線形方程式 $E_h = \{f = 0 | f \in C\}$ が解 $S_h = \{a_t = c_t | c_t \in \mathbb{Q}\}$ を持つ
　then $G \leftarrow G \cup \{d \sum_{t \in T(h)} c_t t\}$
　(d : c_t の分母の LCM)
　else return **nil**
}
return G

命題 61 *アルゴリズム 60 は, 有限個の p を除いて $GB_<(F)$ を与える.*

注意 62 線形方程式が全部解けたとしても, 結果が $Id(F)$ のグレブナ基底になっている保証は現時点ではないので, この命題は不十分である. 実は, 次で述べる結果により, アルゴリズム *60* が **nil** でない多項式集合を返せば, それはただちに $Id(F)$ のグレブナ基底となっていることが分かる. これについては, 線形方程式の, 結果の大きさに応じた計算量を必要とする求解法とともに, 後で述べる.

3.2.2 グレブナ基底候補がグレブナ基底となる条件

ここでは, change of ordering の場合には, trace lifting の場合に必要だったグレブナ基底テストとメンバシップテストが不必要になることを示す. $F \subset \mathbb{Z}[X]$ とする.

仮定 63 有理数体上と有限体上の計算が異らないよう次の仮定をおく.

1. 不必要対の検出基準は頭項のみで行う.
2. 正規形計算において, 正規化に用いる元 *(reducer)* の選択は, 正規化される多項式の項および *reducer* の頭項の集合にのみ依存する.

定義 64 (compatibile な素数) 素数 p が F に関し *compatible* とは,

$$\phi_p(Id(F) \cap \mathbb{Z}[X])(= \phi_p(Id(F) \cap \mathbb{Z}_{(p)}[X])) = Id(\phi_p(F))$$

なること.

定義 65 素数 p が $(F, <)$ に関して *strongly compatible* とは p が F に関して *compatible* で

$$E_<(Id(F)) = E_<(Id(\phi_p(F))$$

なること.

定義 66 *permissible* な素数*)* 素数 p が $(F, <)$ について *permissible* とは各 $f \in F$ に対し p が $HC_<(f)$ を割らないこと.

定義 67 $f \in \mathbb{Q}[X]$ が p に関し *stable* とは $f \in \mathbb{Z}_{(p)}[X]$ なること.

定義 68 (modular グレブナ基底の逆像) $G \subset Id(F) \cap \mathbb{Z}[X]$ が F の $<$ に関する *p-compatible* なグレブナ基底候補とは p が $(G, <)$ について *permissible* で $\phi_p(G)$ が $Id(\phi_p(F))$ の $<$ に関するグレブナ基底なるときをいう.

注意 69 *compatibility* は *order* に独立な概念である.

補題 70 $G \subset \mathbb{Z}[X]$, p を $(G, <)$ について *permissible* な素数, $f \in \mathbb{Z}[X]$ とする. このとき仮定 *63* のもとで

$$NF(\phi_p(f), \phi_p(G)) = \phi_p(NF(f, G)).$$

定理 71 $G \subset Id(F) \cap \mathbb{Z}[X]$ を $Id(F)$ の $<$ に関するグレブナ基底とする. もし p が $(G,<)$ に関して *permissible* かつ $\phi_p(G) \subset Id(\phi_p(F))$ ならば p は F に関して *compatible* である. 更に, $\phi_p(G)$ は $Id(\phi_p(F))$ のグレブナ基底で p は $(F,<)$ について *strongly compatible* である.

この定理で, $\phi_p(G) \subset Id(\phi_p(F))$ は $Id(\phi_p(F))$ のグレブナ基底によりチェックできる. よって, p の compatibility のチェックは有理数体上, 有限体上の任意の order でのグレブナ基底を計算することで行うことができる.

もし, 入力が既にある order でのグレブナ基底なら compatibility のチェックは極めて簡単である.

系 72 $G \subset \mathbb{Z}[X]$ が $<$ に関する $Id(G)$ のグレブナ基底とする. もし p が $(G,<)$ に対し *permissible* ならば $\phi_p(G)$ は $Id(\phi_p(G))$ のグレブナ基底で p は $(G,<)$ に対し *strongly compatible.*

次の定理は, グレブナ基底候補が実際にグレブナ基底になるための十分条件を与える. すなわち, 我々が求めるものである.

定理 73 p が F について *compatible* で G が $<$ に関して *p-compatible* なグレブナ基底候補ならば, G は $<$ に関する $Id(F)$ のグレブナ基底である.

次の定理は前定理の精密化である. すなわち, 昇順に計算された部分的な p-compatible なグレブナ基底候補が実際にグレブナ基底の一部となっていることを保証する. これは, 途中までの結果を再利用できるという点で有用である. また, 後で述べるように, グレブナ基底のある特定の元, 例えば, 順序最小の元のみを求めたい, あるいは elimination 後の結果のみを求めたい場合にも有用である.

定理 74 p が F について *compatible* とする. $\overline{G} \subset GF(p)[X], \overline{G} = GB_<(Id(\phi_p(F))$ とし $\overline{g}_1 < \cdots < \overline{g}_s$ なる $\overline{g}_i$ により $\overline{G} = \{\overline{g}_1, \cdots, \overline{g}_s\}$ と書く. 更に, ある正数 $t \leq s$ に対し, $g_i \in Id(F) \cap \mathbb{Z}_{(p)}[X]$ $(1 \leq i \leq t)$ が存在して $\phi_p(g_i) = \overline{g}_i$ かつ g_i は $\{g_1, \cdots, g_{i-1}\}$ について被約とする. このとき, $g_1, \cdots, g_t$ は $GB_<(Id(F))$ の最初の t 個の元に一致する.

以上述べたことにより, 次のような一般的な change of ordering アルゴリズムが得られる.

Procedure 1

$candidate(F, p, <)$
$Input : F \subset Z[X]$

素数 p
$order <$
Output : F の *p-compatible* なグレブナ基底候補または **nil**
(各 F に対し, **nil** を返す p の個数は有限個でなければならない.)

アルゴリズム **75** (compatibility check によるテストの省略)

gröbner_by_change-of-ordering$(F, <)$
Input : $F \subset \mathbb{Z}[X]$
$order <$
Output : $Id(F)$ の $<$ に関するグレブナ基底 G

$G_0 \leftarrow F$ の, ある $order <_0$ に関するグレブナ基底; $G_0 \subset \mathbb{Z}[X]$
again:
$p \leftarrow (G_0, <_0)$ に関して *permissible* な未使用の素数
$G \leftarrow$ *candidate*$(G_0, p, <)$
If $G =$ **nil** *goto* **again:**
else return G

candidate() においては, *p*-compatible なグレブナ基底候補を返す任意のアルゴリズムが使用可能である. これまで述べたものでは,

- *tl_guess*()
- 斉次化 + *tl_guess*() + 非斉次化
- *candidate_by_linear_algebra*()

が適合する. これらのうち, 前者 2 つについては明らかだが, 最後のものについては検証を要する. これについて次節で述べる.

3.2.3 *candidate_by_linear_algebra*()

補題 76 アルゴリズム *60* において C に属する多項式は p について *stable* で, $E_{h,p} = \{\phi_p(c) = 0 | c \in C\}$ は一意解を持つ.

系 **77** n を不定元 a_t の個数とすると, E_h から次の性質をもつ *subsystem* E'_h を選ぶことができる.

- E'_h は n 個の方程式からなる.
- $\phi_p(E'_h)$ は $GF(p)$ 上で一意解をもつ.

これから次のことが分かる.

- E'_h は $\mathbb{Q}$ 上一意解を持ち, 解は p について *stable.*
- E_h が解をもてば, それは E'_h の一意解に一致する.

定理 78 アルゴリズム *60* が多項式集合 G を返せば, G は F の $<$ に関して *p-compatible* なグレブナ基底候補である.

上の補題を用いて E_h を次の手順で解く.

1. E'_h を選ぶ.
2. $S \leftarrow E'_h$ の一意解.
3. もし S が E_h を満たせば S は E_h の一意解, さもなくば E_h は解を持たない.

E_h は E'_h を $GF(p)$ 上で解く仮定で得られる. 以下では, E'_h を解く方法について述べる. これは次のように定式化できる.

問題 79 M, B をそれぞれ $n \times n$, $n \times 1$ 整数行列とし, X を, 未定係数を成分とする $n \times 1$ 行列とする. $det(\phi_p(M)) \neq 0$ のもとで, $MX = B$ を解け.

M, B は, 一般に, 長大な整数を成分に持つ密行列となる. しかし, もともとの入力多項式の係数が小さい場合, グレブナ基底の元の係数すなわち $MX = B$ の解 X は比較的小さい場合が多い. このような場合にこの問題を Gauss 消去で解くことはこの節の最初の例で述べたように非効率的である. このような場合に有効なのが, Hensel 構成, 中国剰余定理などによる modular 計算である. ここでは Hensel 構成による方法を紹介する.

アルゴリズム **80**
solve_linear_equation_by_hensel(M, B, p)
Input : $n \times n$ 行列 M, $n \times 1$ 行列 B
　　$\phi_p(det(M)) \neq 0$ なる素数
Output : $MX = B$ なる $n \times 1$ *matrix* X

$R \leftarrow \phi_p(M)^{-1}$
$c \leftarrow B$
$x \leftarrow 0$
$q \leftarrow 1$
$count \leftarrow 0$
do {
　　$t \leftarrow \phi_p^{-1}(R\phi_p(c))$
　　$x \leftarrow x + qt$
　　$c \leftarrow (c - Mt)/p$
　　$q \leftarrow qp$
　　$count \leftarrow count + 1$
　　(ϕ_p^{-1} は $[-p/2, p/2]$ に正規化された逆像, $(c - Mt)/p$ は整除.)
　　if $count =$ **Predetermined_Constant** *then* {
　　$count \leftarrow 0$
　　$X \leftarrow inttorat(x, q)$
　　if $X \neq$ **nil** かつ $MX = B$ *then return* X
　　}
}

アルゴリズム **81**
$inttoat(x, q)$
Input : 整数 x, q
Output : $bx \equiv a \bmod q$ かつ $|a|, |b| \leq \sqrt{\frac{q}{2}}$ なる有理数 a/b または **nil**

Predetermined_Constant はある正整数で, 有理数に引き戻してチェックを行う頻度を制御する. アルゴリズム 80 において c は $nMAX(\|M\|_\infty, \|B\|_\infty)$ で押えられることがわかる. これは, 各ステップがある constant 時間内で計算できることを示す. また, 解の分母分子が A で押えられていれば, $q > 2A^2$ になった段階で解を復元できる. これは, 解の大きさに応じた手間で計算できることを意味する. これに対して, 係数膨張を押えた Gauss 消去法である fraction-free 法によっても, 解の大きさにかかわらず, 係数行列の行列式を計算してしまうという点で, Gauss 消去法は, この問題を解くためには不適切である.

4 F_4 アルゴリズム

代数方程式求解に限らず, 代数幾何における不変量の計算などにおいても任意入力多項式集合からの Gröbner 基底基底の計算は, 計算量的にみて dominant step となることが多い. このような場合の計算法としては Buchberger アルゴリズムがほぼ唯一の方法であったが, 最近 Faugère により F_4 (あるいは F_5) アルゴリズムが提案され, その高速性が注目されている. 本節では, このアルゴリズムについて概説する.

4.1 Symbolic preprocessing

アルゴリズム 37 を実行する場合, D からどの元を選んで来るかで, その後の計算の進行の様子が全くことなる, ということが起こる. また, 最善の元を選んだつもりでも, 特に有理数体上などにおいて, $NF(Sp(f,g),G)$ の係数が極端に大きくなり, 計算不能になることも起こる. そのため, trace lifting, homogenization などの種々のテクニックのもとで計算が試みられて来た.

ここで, Buchberger アルゴリズムの核心である $NF(Sp(f,g),G)$ について考察する.

$NF(Sp(f,g),G)$ は次のような手順で計算される.

$r \leftarrow af - bg$ (a,b は単項式)
while (r の項 t を割り切る $h \in G$ がある)
　　$r \leftarrow r - ch$
　　(r に t の項はない)

ここで, r を簡約するのに必要な $h \in G$ は一見実際に簡約操作を実行しなければ分からないように見えるが, 冗長な元 (すなわち, 実際には簡約に用いられない元) も許せば, 次のような方法で事前に得られる.

アルゴリズム **82** *(Symbolic preprocessing)*

入力 : 多項式 f, 多項式集合 G
出力 : $Red = \{ah | a : 単項式, h \in G\}$
$T \leftarrow T(f)$
$Red \leftarrow \emptyset$
while $\exists t \in T \exists g \in G\, s.t.\, HT(g)|t$ *do* {
　　$Red \leftarrow Red \cup \{t/HT(g) \cdot g\}$
　　$T \leftarrow (T \setminus \{t\}) \cup T(reductum(t/HT(g) \cdot g)$
}

return Red

このアルゴリズムで得られた Red (Reducer) は次の性質をもつ:

- $\{f\} \cup Red$ の元のある項 t がある $g \in G$ に対し $HT(g)$ で割り切れるならば, $HT(x) = t$ なる $x \in Red$ がある.

これより, 次が言える.

- $\{f\}$ の, Red の元による K 係数の簡約操作での正規形は, G に関する正規形となる.

これを, 行列の言葉で言い換えるために次の準備をする. まず, $\{f\} \cup Red$ に現れる全ての項を T とし, T の元を順序の高い順に並べたものを $t_1 > t_2 > \cdots$ とする. T で K 上張られる線形空間の元 h の, $(t_1, t_2, \cdots)$ を基底とする行ベクトル表現を $[h]$, 行ベクトル v に対する多項式を $poly(v)$ と書くことにする.

$$[f] = [f_1 f_2 \cdots]$$

$$r_i = [r_{i1} r_{i2} \cdots] \quad (r_i \in Red)$$

とするとき,

$$A = \begin{pmatrix} f_1 & f_2 & \cdots \\ r_{11} & r_{12} & \cdots \\ r_{21} & r_{22} & \cdots \\ & \cdots & \end{pmatrix}$$

とならべ, これを, 行に関する基本変形により次の条件を満たす行列 B に変形する.

- $poly(B_i)$ が 0 でないとき, $poly(B_k)(k \neq i)$ は $HT(poly(B_i))$ を含まない.

$$B = \begin{pmatrix} 1 & 0 & ? & ? & ? & 0 & \cdots \\ 0 & 1 & ? & ? & ? & 0 & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & \cdots \\ & & & \cdots & & & \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots \end{pmatrix}$$

このとき, $poly(B_i)$ のうち, $HT(poly(B_i))$ が $\{HT(r) | r \in Red\}$ に含まれないものが, $NF(f, G)$ となる.

4.2 F_4 アルゴリズム

前節での 正規形計算の言い替えは, 多項式剰余演算計算においてもしばしば用いられ, 特に目新しいものでもない. F_4 アルゴリズムは, 複数の S-多項式をまとめて行列形式で簡約する. そのための symbolic preprocessing も, 出発点が, S-多項式に現れる項の和集合となるだけで, 既に述べたアルゴリズムがそのまま適用される.

アルゴリズム **83** *(Symbolic preprocessing)*

入力 : 多項式集合 F, 多項式集合 G
出力 : $Red = \{ah | a : 単項式, h \in G\}$
$T \leftarrow \cup_{f \in F} T(f)$
$Red \leftarrow \emptyset$
while $\exists t \in T \exists g \in G \, s.t. \, HT(g)|t$ *do* {
    $Red \leftarrow Red \cup \{t/HT(g) \cdot g\}$
    $T \leftarrow (T \setminus \{t\}) \cup T(reductum(t/HT(g) \cdot g)$
}
return Red

行列 A も, 全ての S-多項式に対応するベクトルおよび symbolic preprocessing で得られた Red に属する多項式に対応するベクトルを行とする行列として構成される. この行列を, 行基本変形により, 次の条件を満たす行列 B に変形する.

- $poly(B_i)$ が 0 でないとき, $poly(B_k)(k \neq i)$ は $HT(poly(B_i))$ を含まない.

これは, $t_1 > t_2 > \cdots$ を新たな変数とみて, この順序で reduced な Gröbner 基底基底を計算した結果に対応する.

$$F' = \{h = poly(B_i) | h \neq 0, HT(h) \notin \{HT(r) | r \in Red\}\}$$

$$Red' = \{h = poly(B_i) | h \neq 0, HT(h) \in \{HT(r) | r \in Red\}\}$$

とおく.

命題 84 *1.* $h \in F'$ は $G \cup (F' \setminus \{h\})$ に関して正規形

2. $f \in F$ ならば $f \xrightarrow[G \cup F']{*} 0$

Proof
1: $h \in F'$ ならば $T(h) \cap \{HT(r) | r \in Red\} = \emptyset$. これは, Red の作り方より h が G に関

して正規形であることを意味する. もちろん h は $F' \setminus \{h\}$ に関しても正規形である.
2: $f \in F$ とする.

$$NF(f,G) = f - \sum_{r_i \in Red} c_i r_i \quad (c_i \in K)$$

と書ける. よって, $NF(f,G)$ は $F \cup Red$ で K 上生成される線形空間に属する. ここで, $F \cup Red$ と $F' \cup Red'$ が K 上同一の線形空間を生成し, かつ $NF(f,G)$ には Red' の元の頭項は現れないから,

$$NF(f,G) = \sum_{f_i \in F'} d_i f_i \quad (d_i \in K)$$

と書ける. この線形和は直ちに F' に関する正規形計算となる. ∎

2. により F として, いくつかの S-多項式の集合をとり, F' をまとめて G に付け加えることにより, F に属する S-多項式が 0 に簡約されることが分かる. また, 1. により, アルゴリズムの停止性も言える.

F として一つの S-多項式をとった場合が通常の Buchberger アルゴリズムであり, その場合には, 選び方 (selection strategy) によって効率に大きく差がでることは既に述べた. F_4 においても F の選び方が問題となるが, 複数元を選択できることで, selection strategy の効率に対する影響が少なくなることが実験により知られている.

例えば, 次のような strategy により, 多くの例が効率よく計算できる.

定義 85 *S*-多項式の *sugar* の最小のものを全て選ぶ.

ここで, F_4 の場合, 正規形計算において通常の sugar は意味を持たないので, ある sugar の S-多項式を集めて計算した場合, 生成された基底の sugar もその値であるとして, 再帰的に sugar の値を定めることとする. 特に, 入力多項式集合が homogeneous の場合, この strategy により, 各ステップで計算される基底は, 次数が d の場合, **reduced** な Gröbner 基底基底のうちの, d 次の元全てとなる. これは, homogeneous ideal の場合,

1. d 次の基底を生成するための S-多項式は全て $d-1$ 次以下の基底から作られる.

2. d 次の斉次多項式は, $d+1$ 次以上の基底では簡約されない.

4.3 modular 計算による線形方程式の求解

F_4 においては, selection strategy に従って構成された行列を, 左基本変形するという操作の繰り返しで基底を生成していく. ここで, 左基本変形は, 線形方程式求解とみなすことができる. すなわち,

1. 行列 A から線形独立な行を全て抜きだして A' を作る.

2. A' の列を左から順に見て, それまで取り出した列と線形独立な列を抜き出す, という操作を繰り返して正方行列 A'' を作る.

3. A' で残った列を集めて B'' とする.

4. $A''X = B''$ なる線形方程式を解く.

5. X から, A' の行基本変形の結果を構成する.

元の体上で考える限り, これは単なる言い替えに過ぎないが, 例えば有理数体上の場合, $det(A'') \neq 0$ なる A'' を modular 計算により推測して抜きだし, $A''X = B''$ の解候補を modular 計算などにより構成し,

- A の各行に, 第 5 項の結果を代入して 0 になることを確かめる.

というチェックにより modular 計算の結果が, 有理数体上での行基本変形の結果と等しいこと, 特に A を構成する多項式で張られていること, 線形方程式の解の一意性により言える. このような解候補を与えるものとして, 中国剰余定理および Hensel 構成がある. 中国剰余定理を用いる方法は, 法 p が有理数上の掃き出しと異なるものを与えない限り, 自明な方法で精度をあげていき, 整数-有理数変換の結果が stable になったら A に代入してチェックという方法である. また, Hensel 構成は アルゴリズム 80 そのものである.

Faugère によれば, F_4 においては B が大きくなるため, B の列数が A のサイズを越える場合には中国剰余定理を用いたほうがよいとのことである.

いずれにせよ, modular 計算による方法が効率を上げる場合というのは, $det(A'')$ に比べて解の係数が小さい場合である. これは一般に期待できることではないが, 前節で述べたように, homogeneous の場合に F_4 が reduced な Gröbner 基底基底の一部を生成する, ということから, reduced にした場合に係数が小さくなるような問題では modular 計算が効率を向上させることが期待できる.

4.4 実験

今回, F_4 を実装してみるきっかけとなったのは, McKay 問題 [15] の中間基底において, 16 次の基底を線形形式とみて, inter reduction してみた結果, 係数が極端に小さくなったことであった. もともと, Buchberger アルゴリズムで McKay を計算する場合, 最後に係数が大変小さい基底が何本か出て終了する. 特に, 最初に現れる, 係数の小さい基底が, 16 次の基底のうち最後のものであったため, その基底を用いて一つ前の基底を簡約したところ, 係数が小さくなったため, その操作を基底全てに適用したところ, 16 次の基底全ての係数が, inter reduction により小さくできることが分かったのである.

既に述べたように, このような場合には, modular 計算による解候補の計算が有効となる. しかし, 15 次の基底を inter reduction したところ, 係数の大きさはほとんど変わらず, 大きいままであった.

以上のような背景のもとに, 現在の実装での McKay 問題に対する Buchberger アルゴリズム計算 (homogenization+trace lifting) および F_4 の比較を示す.

表 3: 計算時間の比較

	total	GaussElim	ChRem	IntRat	Check
Buchberger	264710	—	—	—	—
F_4 homo	32880	6437	6300	5763	13340
F_4 non homo	39970	4832	6143	18240	9950

表 4: 各次数の行列のサイズ

	10	11	12	13	14	15
F_4 homo	(56,334)	(119,441)	(182,545)	(362,774)	(671,1009)	(1195,1359)
F_4 non homo	(62,339)	(128,448)	(137,461)	(227,571)	(307,621)	(955,1149)
	16	17	18	19	20	21
F_4 homo	(836,688)	(2323,1761)	(895,964)	(204,271)	(446,515)	(308,374)
F_4 non homo	(555,508)	(2022,1763)	—	—	—	—
	22	23	24	25		
F_4 homo	—	—	—	—		
F_4 non homo	(799,868)	(300,367)	(398,467)	(361,427)		

まず, total 時間が 1/8 程度に減っていることに注目したい. これは, 基底の計算時間の内訳をみればわかるように, Buchberger アルゴリズムによる計算で全体の計算時間の大部分を占めていた 16 次の計算が 1/100 程度の時間で終っているためである. これは, 最大係数のビット長が reduced な基底で非常に小さくなることに対応している. それ以上の次数については, 小さい係数を持つ多項式による簡約化で済むことが高速化の理由である.

4.5 考察

F_4 は本質的には Buchberger アルゴリズムの改良と考えられるが, 次のような長所を持つ.

表 5: 各次数の基底の計算時間

	10	11	12	13	14	15
Buchberger	2.3	11	43	164	1214	32512
F_4 homo	1.8	10	45	357	2574	26519
F_4 non homo	2.2	12	28	166	1064	35906

	16	17	18	19	20	21	22	23	24	25
Buchberger	205234	10300	4530	—	10700	—	—	—	—	—
F_4 homo	1340	1097	359	45	208	150	—	—	—	—
F_4 non homo	937	902	—	—	—	—	341	132	186	164

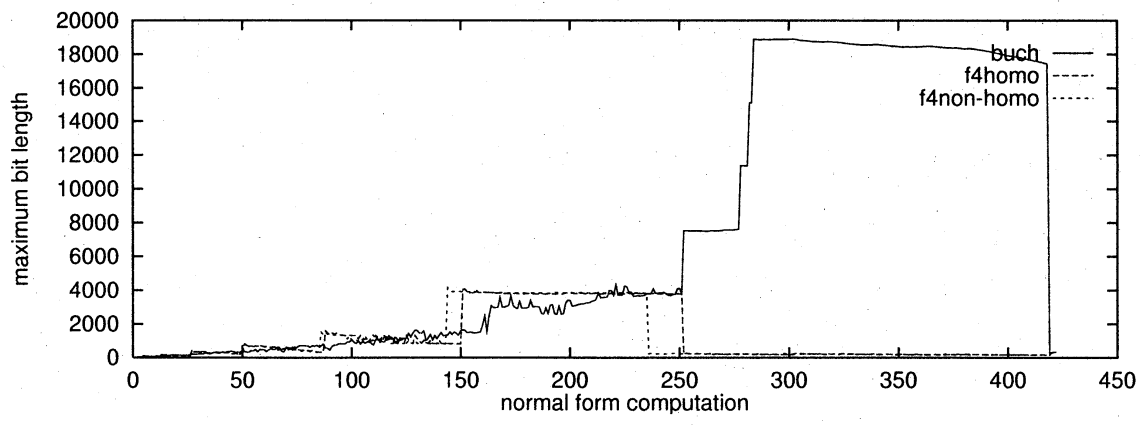

図 1: 中間基底の最大係数のビット長

- 行列の掃き出し中は, 項の順序比較が単なる index の比較になる.
- 中間基底を reduced あるいはそれに近い形に保つため, その後の掃き出し計算が効率化する. (「対角化」された行列による簡約化 vs. 「三角化」さらた行列による簡約化)
- reduced な基底の係数が小さくなる場合には, modular 計算による効率化が期待できる. ただし, この場合には, 行列の各行の 0 簡約チェックが必須である.

効率に関していえば, Risa/Asir での現状は, Faugère の home page

`http://posso.lib6.fr/ jcf/bench.html`

にあるデータに及ぶべくもない. たとえば, McKay で, P6-200MHz PC 上で 54 秒と報告されている. しかし, Faugère[10] に,

Moreover, since big integer computations could be done by means of p-adic or multi modular arithmetics it means that the cost of an integer computation is roughly

time of modular computation * size of the output coeffs

と書かれていることから, 中間基底に対する正当性チェックどころか, 単なる modular Gröbner 基底基底計算を, 中国剰余定理による結果が stable になるまで繰り返しているだけ, という可能性も捨て切れない. 彼がもう少し実装を明らかにしてくれることを望んでいる.

Faugère は,

1. Buchberger アルゴリズムにおける selection strategy と比べて, F_4 では "make no choice"
2. non homogeneous case の挙動を見れば, F_4 は Buchberger アルゴリズムではない.

と書いているが, 1. に関しては, critcal pair の subset をどう選ぶかで, 効率に大きく差が出ることは, McKay の例から明らかなので, ちょっと疑問であり, 2. に関しても, アルゴリズムの基本構造はどう見ても Buchberger アルゴリズムと考えられるのでやはり疑問である. とはいえ, 従来の Buchberger アルゴリズムより効率よく計算できる場合があることは確かであり, 各部の改良を含めて, より効率よい実装を目指すことが実用上重要であると考えられる.

5 グレブナ基底の応用

グレブナ基底は標準基底 (standard basis) とも呼ばれ, 消去法以外にもさまざまな応用を持つ. 本節では, それらのいくつかについて解説する.

記法 86 以下で, 次のような記法を用いる.
K : 体.
$X = \{x_1, \cdots, x_n\}$: 不定元
$R : K[X]$
T : R の項全体
$HT_<(f)$: f の $<$ に関する頭項.
$HC_<(f)$: f の $<$ に関する頭係数.
$GB_<(S)$: S の $<$ に関する被約グレブナ基底.
$NF_<(f, G)$: f の G に関する正規形の一つ. G がグレブナ基底ならば一意的に定まる.

5.1 イデアルに関する演算

命題 87 *(イデアルの相等)*
イデアル $I, J \subset R$ に関し

$$I = J \Leftrightarrow GB(I) = GB(J).$$

命題 88 *(イデアルを法とする合同, メンバシップ)*
イデアル I, $f, g \in R$ に関し

$$f \equiv g \bmod I \Leftrightarrow NF(f, GB(I)) = NF(g, GB(I)).$$

特に

$$f \in I \Leftrightarrow NF(f, GB(I)) = 0.$$

命題 89 *(自明なイデアル)*
イデアル $I \subset R$ に関し

$$I = R \Leftrightarrow GB(I) = \{1\}.$$

命題 90 *(elimination イデアル)*
I をイデアルとする. $X = (X \backslash U) \cup U$ とし, この分割により $\forall u \in T(U), \forall x \in T(X \backslash U), u < x$ なる *order* $<$ を用いると,

$$GB(I \cap K[U]) = GB(I) \cap K[U]$$

証明 $f \in J = I \cap K[U]$ とする. $f \in I$ よりある $g \in GB(I)$ が存在して $HT(g)|HT(f)$. $HT(f) \in T(U)$ より $HT(g) \in T(U)$. ここで, $<$ の性質より, $HT(g) \in T(U)$ ならば $g \in K[U]$. よって $g \in GB(I) \cap K[U]$ で, $GB(I) \cap K[U]$ は J のグレブナ基底. ∎

命題 91 *(イデアルの交わり)* $I = Id(f_1, \cdots, f_l)$, $J = Id(g_1, \cdots, g_m)$ とすると,

$$I \cap J = (yIR[y] + (1-y)JR[y]) \cap R$$

証明 $f \in I \cap J$ とすると, $f = yf + (1-y)f \in (yI + (1-y)J) \cap R$. 逆に $f = yg + (1-y)h$ $(g \in IR[y], h \in JR[y])$ とし, $f \in R$ とする. この時, $y = 0$ を代入して, $f = h|_{y=0} \in J$. $y = 1$ を代入して, $f = g|_{y=1} \in I$ より OK. ∎

系 92 $I = Id(f_1, \cdots, f_m)$, $J = Id(g_1, \cdots, g_l)$ に対し

$$GB(I \cap J) = GB(\{yf_1, \cdots, yf_m, (1-y)g_1, \cdots, (1-y)g_l\}) \cap R$$

により $I \cap J$ が計算できる. *(左辺は $X < y$ なる elimination order で計算する.)*

定義 93 *(イデアル商)* イデアル I, R の部分集合 S に対し, イデアル商 $I : S$ を

$$I : S = \{f \in R | fS \subset I\}$$

で定義する. $J = Id(S)$ とすれば, $I : S = I : J$ で, $J = Id(f_1, \cdots, f_m)$ ならば,

$$I : S = \bigcap_{i=1}^{m} I : Id(f_i)$$

命題 94 $I : Id(f) = \frac{1}{f}(I \cap Id(f))$

証明 $g \in I : Id(f)$ ならば $gf \in I$ より $gf \in I \cap Id(f)$. よって, $g \in \frac{1}{f}(I \cap Id(f))$.
逆に, $g \in \frac{1}{f}(I \cap Id(f))$ ならば $gf \in I \cap Id(f)$ より $g \in I : Id(f)$. ∎

系 95 $I \cap Id(f)$ の生成元が求まれば, それらは f を因子に持つので, それぞれ f で割ることにより $I : Id(f)$ が求まる. 一般の場合 $I : S$ はそれらの交わりとなるが, $I \cap Id(f)$ を含めてイデアルの交わりの計算は系 *92* により計算できるので, イデアル商も計算できることになる.

定義 96 *(saturation)*
I をイデアル, $f \in R$ とすれば, $I : f^i$ はイデアルの増大列だがら, ある $s \in \mathbb{N}$ が存在して

$$i \geq s \Rightarrow I : f^i = I : f^s$$

が成り立つ. このとき

$$I : f^{\infty} = I : f^{s}$$

と定義し, I の f に関する *saturation* と呼ぶ.

命題 97 I をイデアル, $f \in R$ に対し,

$$I : f^{\infty} = (IR[y] + (1 - yf)R[y]) \cap R$$

すなわち $I : f^{\infty}$ は *elimination* イデアルにより計算できる.

証明
右辺 $\subset$ 左辺 $g \in$ 右辺とすると, $g = ah + (1 - yf)b$ $(a, b \in R[y])$ と書ける. この式で, $y = 1/f$ とおいて, 両辺に f^d (d:十分大) を掛ければ $f^d g \in I$ すなわち $g \in I : f^d$. よって $g \in$ 左辺.
左辺 $\subset$ 右辺 $g \in$ 左辺, すなわちある d に対し $f^d g \in I$ とする. このとき

$$g \equiv (yf)^d g \equiv 0 \bmod IR[y] + (1 - yf)R[y]$$

また, $g \in R$ より $g \in$ 右辺. ∎

5.2 剰余環, 次元

命題 98 *(剰余環の表現)*
イデアル I に対し, 剰余環 R/I は, 正規形を元として定義される代数構造に同形である. すなわち, R/I は, 元の集合として
$\{NF(f, GB(I)) | f \in R\}$ と同一視でき, その加法 $(\oplus)$ · 乗法 $(\odot)$ として次式で定義されるものに同形となる.

$$f \oplus g = NF(f + g, GB(I))$$

$$f \odot g = NF(fg, GB(I))$$

命題 99 *(剰余環の線形空間としての基底)*
イデアル I に対し, 剰余環 R/I は K-線形空間とみなせるが, その線形空間の基底としてイデアルの グレブナ基底に対して正規形である項全体がとれる. すなわち, R/I の基底として

$$\{u \in T | HT(f) \not| u (\forall f \in GB(I))\}$$

がとれる.

定義 100 I をイデアルとする. $U \subset X$ に対し, $I \cap K[U] = 0$ が成り立つとき U は *independent modulo* I という.

定義 101 *(イデアルの次元)*
イデアル I に対し, イデアルの次元 $dim(I)$ を

$$dim(I) = max(|U||U \subset X \ independent \ modulo \ I)$$

で定義する.

注意 102 *(幾何学的意味)*

1. イデアル I の次元は, K の代数閉包上で考えた代数的集合 $V(I)$ の成分の最大次数に等しい.
2. より一般に, 素イデアルの減少列の長さを用いて環の次元 *(Krull* 次元*)* が定義され, 上の定義と一致することが示される.

定義 103 *(Hilbert function)*
$R = K[X]$ の s-次斉次元全体を R_s と書くことにする. イデアル I に対し, $I_s = I \cap K[X]_s$ と書く. 斉次イデアル I に対し, I の *Hilbert function* $H_{R/I}(s)$ を

$$H_{R/I}(s) = dim_K R_s/I_s$$

で定義する.

命題 104 $<$ を任意の *order* とし, J を I の元の頭項で生成されるイデアルとすると, $H_{R/I}(s) = H_{R/J}(s)$

5.3 消去法

定義 105 イデアル I に対し, I の *radical* (根基) $\sqrt{I}$ を

$$\sqrt{I} = \{f \in R | \exists e \in \mathbb{N} \, s.t. \, f^e \in I\}$$

で定義する. $\sqrt{I}$ もイデアルとなる.

定義 106 L を K の拡大体とし, $V \subset K^n$ とする. このときイデアル $I(V) \subset R$ を

$$I(V) = \{f \in R | f|_V = 0\}$$

で定義する.

次の定理は消去法の基本となる.

定理 107 *(Nullstellensatz; Hilbert の零点定理)*
K を体, $\bar{K}$ を K の代数閉包とする. イデアル $I \subset K[X]$ に対し,

$$I(V_{\bar{K}}(I)) = \sqrt{I}$$

系 108 イデアル I, J に対し,

$$V_{\bar{K}}(I) = V_{\bar{K}}(J) \Leftrightarrow \sqrt{I} = \sqrt{J}$$

命題 109 *(0 次元イデアルの性質)*
代数閉体 K 上の多項式環のイデアル I の零点の個数が有限個 $\Leftrightarrow$ R/I が K 上有限次元の線形空間

証明
$\Rightarrow$) 有限個の解を $r_k = (r_{k1}, \cdots, r_{kn})(k = 1, \cdots, m)$ とする. $f_i(x_i) = \prod_k (x_i - r_{ki})$ とおくと, $f_i(x_i)$ は I の零点上で 0 となるから, Hilbert の零点定理によりある t が存在して $f_i(x_i)^t \in I$. よって, $GB(I)$ にも, 各 i に対し, $HT(g)$ が x_i の冪となるものが存在する. すると, 命題 99 より R/I は K 上有限次元となる.
$\Leftarrow$) 各 i に対し, 変数中で x_i が最低の順序になるような辞書式順序をとれば, $GB(I)$ 中に, $f_i(x_i)$ なる一変数多項式が存在することがわかる. よって, 解は有限個. ∎

変数順序として $x_1 < x_2 < \cdots < x_n$ なる辞書式順序を考えれば, イデアル I の零点が有限個の場合は, すべての i に対し, ある $f_i \in GB(I) \cap (K[x_1, \cdots, x_i] \setminus K[x_1, \cdots, x_{i-1}])$ が存在する. よって $f_1(x_1)$ から根 α_1 を求め, $f_2(\alpha_1, x_2)$ から根 α_2 を求め, という操作を繰り返せば, F の共通零点をすべて求めることができる.

5.4 加群のグレブナ基底

自由加群 $K[X]^l$ および, その部分加群 $M \subset F$ に対してもグレブナ基底が定義される. この場合, 項としては, te_i $(t \in T(X);\ e_i = (0, \cdots, 1, \cdots, 0)$: 第 i 成分のみ 1) をとり,

1. $m \leq tm$ $\forall t \in T, \forall m$: F の項

2. $m_1 \leq m_2 \Rightarrow tm_1 \leq tm_2$ $\forall t \in T, \forall m_1, m_2$: F の項

を満たす全順序を入れる. モノイデアル $E(S)$ も同様に定義され, グレブナ基底も, $E(G)$ が $E(M)$ を生成するものとして定義される. Buchberger アルゴリズムは, S-多項式を, 頭項の F における位置が等しい (すなわち, $HT(a) = t_a e_a$, $HT(b) = t_b e_b$ のとき $a = b$) に対して通常の多項式と同様に定義し, それ以外は 0 と定義すれば全く同様にできる.

加群のグレブナ基底は, syzygy の計算を通して, イデアル, 加群の自由分解 (free resolution) を与える. これにより, イデアル, 加群に対し, Tor, Ext などの計算を可能にするが, ここでは述べない.

6 タイミングデータおよび例

6.1 Shape basis および RUR

定義 110 $f \in K[X]$ が I の *separating element* とは, I の $\overline{K}$ 上の相異る零点 a, b に対し, $f(a) \neq f(b)$ なること.

命題 111 *(shape lemma)*
I を完全体 K 上の *0* 次元 *radical* イデアル とし, f を *separating element* とする. $z << X$ なる任意の順序のもとで, $R[z]$ のイデアル $IR[z] + Id(z - f)$ は

$$\{x_1 - f_1(z), \cdots, x_n - f_n(z), z - f_z(z), m(z)\}$$

という形のグレブナ基底をもつ. この形の基底を *shape basis* と呼ぶ.

shape basis は, 0 次元イデアルの零点を数値で求めようとする場合に, 見掛け上有効な形をしている. 実際, I の零点は, $f_n(x_n)$ の零点により,

$$\{(f_1(\alpha), \cdots, f_n(\alpha)) | m(\alpha) = 0\}$$

と書ける. しかし, 有理数体上で実際に shape basis を求めて見ると, m の係数に比べて f_i の係数が極めて大きくなることが多い. この困難を克服するため, 次の方法が考案された.

命題 112 *(rational univariate representation; RUR)*
前命題と同じ仮定のもとで, $IR[z] + Id(z - f)$ の基底として,

$$\{m'x_1 - g_1(z), \cdots, m'x_z - g_n(z), m(z)\}$$

という形のものがとれる.

m は shape basis の場合と一致する. この基底によるの零点の表現は,

$$\{(\frac{g_1(\alpha)}{m'(\alpha)}, \cdots, \frac{g_n(\alpha)}{m'(\alpha)}) | m(\alpha) = 0\}$$

と書ける. 多くの実例において, g_i の各係数が, m の係数と同程度の大きさに押えられることが分かっており, 0 次元 radical の零点の表現としては RUR によるものが優れているといってよい. RUR の計算法としては, 対称式による方法が最初に提案されていが, modular change of ordering と同様の手法を適用することもでき, RUR が結果の大きさ程度で計算できる.

6.2 タイミングデータ：方程式

$C(n)$ The cyclic n-roots system of n variables. (Faugere *et al.*,1993).

$\{f_1, \cdots, f_n\}$ where $f_k = \sum_{i=1}^{n} \prod_{j=i}^{k+j-1} c_{j \bmod n} - \delta_{k,n}$. ($\delta$ is the Kronecker symbol.)

The variables and ordering : $c_n \succ c_{n-1} \succ \cdots \succ c_1$

$K(n)$ The Katsura system of n+1 variables.

$\{u_l - \sum_{i=-n}^{n} u_i u_{l-i} (l = 0, \cdots, n-1), \sum_{l=-n}^{n} u_l - 1\}$

The variables and ordering : $u_0 \succ u_1 \succ \cdots \succ u_n$.

Conditions : $u_{-l} = u_l$ and $u_l = 0 (|l| > n)$.

$R(n)$ e7 in Rouillier (1996).

$\{-1/2 + \sum_{i=1}^{n} (-1)^{i+1} x_i^k (k = 2, \cdots, n+1)\}$

The variables and ordering : $x_n \succ x_{n-1} \succ \cdots \succ x_1$.

$D(3)$ e8 in Rouillier (1996).

$\{f_0, f_1, f_2, \cdots, f_7\}$

$f_0 = -420y^2 - 280zy - 168uy - 140vy - 120sy - 210ty - 105ay + 12600y - 13440$

$f_1 = -840zy - 630z^2 - 420uz - 360vz - 315sz - 504tz - 280az + 18900z - 20160$

$f_2 = -630ty - 504tz - 360tu - 315tv - 280ts - 420t^2 - 252at + 12600t - 13440$

$f_3 = -5544uy - 4620uz - 3465u^2 - 3080vu - 2772su - 3960tu - 2520au + 103950u - 110880$

$f_4 = -4620vy - 3960vz - 3080vu - 2772v^2 - 2520sv - 3465tv - 2310av + 83160v - 88704$

$f_5 = -51480sy - 45045sz - 36036su - 32760sv - 30030s^2 - 40040ts - 27720as + 900900s - 960960$

$f_6 = -45045ay - 40040az - 32760au - 30030av - 27720as - 36036at - 25740a^2 + 772200a - 823680$

$f_7 = -40040by - 36036bz - 30030bu - 27720bv - 25740bs - 32760bt - 24024ba + 675675b - 720720$

The variables and ordering : $b \succ a \succ s \succ v \succ u \succ t \succ z \succ y$.

Rose The Rose system.

$O_1 : u_3 \succ u_4 \succ a_{46}$, $O_2 : u_3 \succ a_{46} \succ u_4$.

Liu The Liu system.

$\{y(z-t) - x + a, z(t-x) - y + a, t(x-y) - z + a, x(y-z) - t + a\}$

The variables and ordering : $x \succ y \succ z \succ t \succ a$.

Fate The Fateman system, appeared on NetNews.

$\{s^3 + 2r^3 + 2q^3 + 2p^3,\ s^5 + 2r^5 + 2q^5 + 2p^5,$

$-s^5 + (r+q+p)s^4 + (r^2 + (2q+2p)r + q^2 + 2pq + p^2)s^3 + (r^3 + q^3 + p^3)s^2$

$+(3r^4 + (2q+2p)r^3 + (4q^3 + 4p^3)r + 3q^4 + 2pq^3 + 4p^3q + 3p^4)s + (4q+4p)r^4$

$+(2q^2 + 4pq + 2p^2)r^3 + (4q^3 + 4p^3)r^2 + (6q^4 + 4pq^3 + 8p^3q + 6p^4)r$

$+4pq^4 + 2p^2q^3 + 4p^3q^2 + 6p^4q\}$

The variables and ordering : $p \succ q \succ r \succ s$.

$hC(6)$ A homogenization of C(6).
$(C_6 \backslash \{c_1c_2c_3c_4c_5c_6 - 1\}) \cup \{c_1c_2c_3c_4c_5c_6 - t^6\}$
The variables and ordering : $c_1 \succ c_2 \succ c_3 \succ c_4 \succ c_5 \succ c_6 \succ t$.

6.3 タイミングデータ : change of ordering

ここでは, さまざまな change of ordering アルゴリズムのタイミングデータを示す. 計測は, PC (FreeBSD, 300MHz Pentium II, 512MB of memory) で行った. 単位は秒. garbage collection 時間は除いてある.

予め計算してある DRL Gröbner 基底基底から出発して, LEX Gröbner 基底基底計算する. 用いるアルゴリズムは, TL (*tl_guess*()), HTL (斉次化+*tl_guess*()+非斉化), LA (*candidate_by_linear_algebra*(); 0 次元システムのみ) である. 表 6 は DRL から LEX への変換にかかる時間をしめす. *DRL* は, DRL の計算時間を示す. グレブナ基底チェックの効果を示すために, *tl_check*() の時間も示す.

表 6: Modular change of ordering

	$K(5)$	$K(6)$	$K(7)$	$C(6)$	$C(7)$	$R(5)$	$R(6)$
DRL	0.84	8.4	74	3.1	1616	11	1775
TL	∞	∞	∞	∞	∞	∞	∞
HTL	16	1402	1.6×10^5	5.6	2×10^4	383	2.1×10^5
LA	4.7	158	6813	4	435	9.5	258
tl_check	2.3	177	1.3×10^4	1.1	2172	3	40

	$D(3)$	$RoseO_1$	$RoseO_2$	*Liu*	*Fate*	$hC(6)$
DRL	30	0.19	0.15	0.06	0.5	7.2
TL	∞	1.7	354	∞	4	25
HTL	4.1×10^4	1.7	36	18	4	25
LA	585	3.3	12	—	—	—
tl_check	575	0.6	13	17	26	24

整数係数多項式に対し, その **maginitude** を, 係数のビット長の和で定義する. *TL* と *HTL* の差を見るために, 表 7 で, 計算途中における最大 magnitude を示す.

表より明らかに, *TL* は非斉次多項式に対するグレブナ基底計算に不向きであることがわかる. さらに, 表 6 は *HTL* に対する *LA* の優位性を示している. これは, Buchberger ア

表 7: Maximal magnitude

	$C(6)$	$K(5)$	$K(6)$	$RoseO_1$	$RoseO_2$	Liu
TL	> 735380	> 2407737	> 57368231	69764	947321	> 327330
HTL	1992	44187	422732	37220	70018	21095

ルゴリズムが Euclid の互除法に対応していて, 中間係数膨張で効率が左右されるのに対し, modular アルゴリズムの効率は結果の大きさのみに依存することによる.

6.4 タイミングデータ : RUR

RUR の modular 計算のタイミングデータを示す. 計算環境は前節と同様である. ここでは, 予め modular 計算により separating element を求めてある. これらを用いて, それぞれ次のような多項式を添加したイデアルに対し, w に関する RUR 計算を行う. 表で, Quick Test は modular 計算で w が separating element となることをチェックする時間, Normal Form は, 線形方程式を生成するための, monomial の正規形の計算, Linear Equation は, 線形方程式求解の時間である. 表 6.4 では, LEX 基底と RUR で係数の大きさがどのくらい違うかを示している.

$C(6)$ $\quad w - (c_1 + 3c_2 + 9c_3 + 27c_4 + 81c_5 + 243c_6)$
$C(7)$ $\quad w - (c_1 + 3c_2 + 9c_3 + 27c_4 + 81c_5 + 243c_6 + 729c_7)$
$K(n)$ $\quad w - u_n$
$R(5)$ $\quad w - (x_1 - 3x_2 - 2x_3 + 3x_4 + 2x_5)$
$R(6)$ $\quad w - (x_1 - 3x_2 - 2x_3 + 3x_4 + 2x_5 - 4x_6)$
$D(3)$ $\quad w - y$

表 8: 入力イデアルに関するデータ

	$K(5)$	$K(6)$	$K(7)$	$K(8)$	$C(6)$	$C(7)$	$R(5)$	$R(6)$	$D(3)$
$dim_{\mathbb{Q}} R/I$	32	64	128	256	156	924	144	576	128
DRL GB	0.8	7.2	68	798	3.1	1616	11	1775	30

表 9: 計算時間 (秒)

	$K(6)$	$K(7)$	$K(8)$	$C(6)$	$C(7)$	$R(5)$	$R(6)$	$D(3)$
Total	7.4	69	1209	4.6	1643	52	8768	67
Quick test	0.4	3.2	26	0.5	57	6.5	384	3.1
Normal form	1.1	12	308	1.4	762	15	2861	7.3
Linear equation	4.1	43	775	1.4	641	22	3841	45
Garbage collection	1.7	10	100	1.2	181	7.8	1681	11

表 10: Maximal bit length of coefficients in LEX basis and the RUR

	$K(5)$	$K(6)$	$K(7)$	$K(8)$	$D(3)$
LEX	1421	6704	36181	—	6589
RUR	120	249	592	1258	821

6.5 例：準素分解

次の例は, symplectic integrator と呼ばれる安定な積分スキームの数値計算法に関して現れた方程式系である.

$$
\begin{cases}
d_1+d_2+d_3+d_4=1, c_1+c_2+c_3+c_4=1,\\
(6d_1c_2+(6d_1+6d_2)c_3+(6d_1+6d_2+6d_3)c_4)c_1\\
+(6d_2c_3+(6d_2+6d_3)c_4)c_2+6d_3c_4c_3=1,\\
(3d_1^2+(6d_2+6d_3+6d_4)d_1+3d_2^2+(6d_3+6d_4)d_2+3d_3^2+6d_4d_3+3d_4^2)c_1\\
+(3d_2^2+(6d_3+6d_4)d_2+3d_3^2+6d_4d_3+3d_4^2)c_2+(3d_3^2+6d_4d_3+3d_4^2)c_3+3d_4^2c_4=1,\\
(3d_1+3d_2+3d_3+3d_4)c_1^2+((6d_2+6d_3+6d_4)c_2+(6d_3+6d_4)c_3+6d_4c_4)c_1\\
+(3d_2+3d_3+3d_4)c_2^2+((6d_3+6d_4)c_3+6d_4c_4)c_2\\
+(3d_3+3d_4)c_3^2+6d_4c_4c_3+3d_4c_4^2=1,\\
(24d_2d_1c_3+(24d_2+24d_3)d_1c_4)c_2+(24d_3d_1+24d_3d_2)c_4c_3=1,\\
(12d_2^2+(24d_3+24d_4)d_2+12d_3^2+24d_4d_3+12d_4^2)d_1c_2+((12d_3^2+24d_4d_3+12d_4^2)d_1\\
+(12d_3^2+24d_4d_3+12d_4^2)d_2)c_3+(12d_4^2d_1+12d_4^2d_2+12d_4^2d_3)c_4=1,\\
4d_1c_2^3+(12d_1c_3+12d_1c_4)c_2^2+(12d_1c_3^2+24d_1c_4c_3+12d_1c_4^2)c_2+(4d_1+4d_2)c_3^3\\
+(12d_1+12d_2)c_4c_3^2+(12d_1+12d_2)c_4^2c_3+(4d_1+4d_2+4d_3)c_4^3=1
\end{cases}
$$

これを準素分解にかけると, 次の分解が得られる.

$$\begin{cases} 24c_4^2 - 6c_4 + 1 = 0, \\ c_1 = -c_4 + \frac{1}{4}, c_2 = -c_4 + \frac{1}{2}, c_3 = c_4 + \frac{1}{4}, d_1 = -2c_4 + \frac{1}{2}, d_2 = \frac{1}{2}, d_3 = 2c_4, d_4 = 0, \end{cases}$$

$$\begin{cases} 6c_4^3 - 12c_4^2 + 6c_4 - 1 = 0, \\ c_1 = 0, c_2 = c_4, c_3 = -2c_4 + 1, d_1 = \frac{1}{2}c_4, d_2 = -\frac{1}{2}c_4 + \frac{1}{2}, d_3 = -\frac{1}{2}c_4 + \frac{1}{2}, d_4 = \frac{1}{2}c_4. \end{cases}$$

$$\begin{cases} 48c_4^3 - 48c_4^2 + 12c_4 - 1 = 0, \\ c_1 = c_4, c_2 = -c_4 + \frac{1}{2}, c_3 = -c_4 + \frac{1}{2}, d_1 = 2c_4, d_2 = -4c_4 + 1, d_3 = 2c_4, d_4 = 0. \end{cases}$$

$$\begin{cases} 6c_4^2 - 3c_4 + 1 = 0, \\ c_1 = 0, c_2 = -c_4 + \frac{1}{2}, c_3 = \frac{1}{2}, d_1 = -\frac{1}{2}c_4 + \frac{1}{4}, d_2 = -\frac{1}{2}c_4 + \frac{1}{2}, d_3 = \frac{1}{2}c_4 + \frac{1}{4}, d_4 = \frac{1}{2}c_4. \end{cases}$$

6.6 例 : 代数方程式求解

[15] で解かれた odd order replicable function の方程式 $\widehat{E}_{odd}$ は, Moonshine に関して現れた replicable function のうち, もっとも簡単な場合を全て求めるためのものである. 変数は 4 つだが, 方程式は無限個ある. 実際にはこのうち 20 個を選びんだものを E_{odd} とし, その DRL 基底をまず計算した. 方程式のうち, 最大の項数は 1183 最大次数は 17 である. この計算には, 1999 年版の Risa/Asir でも PentiumII 300MHz で 3 日ほどかかる. DRL が計算できれば, その後の計算は比較的容易で, 全ての複素解が整数解であることが示せた.

6.7 例 : 双対曲線の計算

$f(x_1, x_2) \in \mathbb{Q}[x_1, x_2]$ とし, f の total degree を d とすれば,

$$F(x_0, x_1, x_2) = x_0^d f(x_1/x_0, x_2/x_0)$$

は d 次同次多項式で, F の定義する代数曲線の双対曲線は,

$$\begin{cases} u_i = \frac{\partial F}{\partial x_i}(x_0, x_1, x_2) \ (i = 0, 1, 2), \\ F(x_0, x_1, x_2) = 0, \end{cases}$$

から x_0, x_1, x_2 を消去して得られる. 消去法の一つとしてグレブナ基底による消去が可能である.

$$I = Id(u_0 - \frac{\partial F}{\partial x_0}, u_1 - \frac{\partial F}{\partial x_1}, u_2 - \frac{\partial F}{\partial x_2}, F)$$

とする時, $\{x_0, x_1, x_2\} \succ \{u_0, u_1, u_2\}$ なる任意の消去順序により I のグレブナ基底 $GB(I)$ を計算すれば,

$$I \cap \mathbb{Q}[u_0, u_1, u_2] = Id(GB(I) \cap Q[u_0, u_1, u_2]).$$

以下の例で, $V(g_i)$ は $V(f_i)$ の双対曲線である.

$$\begin{cases} f_1 &= x^5 - x^3 + y^2 \\ g_1 &= 108x^7 - 108x^5 + 1017y^2x^4 - 16y^4x^3 - 4250y^2x^2 + 1800y^4x - 108y^6 + 3125y^2 \end{cases}$$

$$\begin{cases} f_2 &= x^6 + 3y^2x^4 + (3y^4 - 4y^2)x^2 + y^6 \\ g_2 &= -256x^6 + (64y^4 - 192y^2 + 864)x^4 + (-192y^4 + 1620y^2 - 729)x^2 - 256y^6 \\ & \quad +864y^4 - 729y^2 \end{cases}$$

$$\begin{cases} f_3 &= 2x^4 - 3yx^2 + y^4 - 2y^3 + y^2 \\ g_3 &= -12x^6 + (-y^2 + 178y - 37)x^4 + (12y^3 - 768y^2 + 2208y + 4608)x^2 - 32y^4 \\ & \quad +1024y^3 - 7680y^2 - 8192y - 2048 \end{cases}$$

参 考 文 献

[1] Knuth, D.E., The Art of Computer Programming, Vol. 2. Seminumerical Algorithms, 2nd ed. Addison-Wesley (1981).

[2] Cox, D., Little, J., O'Shea, D., Ideals, Varieties, and Algorithms. UTM, Springer-Verlag (1992).

[3] Becker, T., Weispfenning, V., Gröbner Bases. GTM 141, Springer-Verlag(1993).

[4] Buchberger, B., Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Geichungssystems. Aequ. Math. 4/3 (1970), 374-383.

[5] Robbiano, L., Term orderings on the polynomial ring. Proc. EUROCAL'85 (LNCS 204), 513-517.

[6] Gebauer, R., Möller, H.M., On an installation of Buchberger's algorithm. J. Symb. Comp. 6/2/3(1989), 275-286.

[7] Traverso, C., Gröbner trace algorithms. Proc. ISSAC '88 (LNCS 358), 125-138.

[8] 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.

[9] Faugère, J.C., Gianni, P., Lazard, D., Mora, T., Efficient Computation of Zero-dimensional Gröbnerr Bases by Change of Ordering. J. Symb. Comp. 16/4(1993), 329-344.

[10] Faugère, J.C., A new efficient algorithm for computing Groebner bases (F_4), Journal of Pure and Applied Algebra (139) 1-3 (1999), 61-88.

[11] Gianni, P., Trager, B., Zacharias, G., Gröbner bases and primary decomposition of polynomial ideals. J. Symb. Comp. 6/2,3(1988), 149-167.

[12] Shimoyama, T., Yokoyaka, K., Localization and Primary Decomposition of Polynomial ideals. J. Symb. Comp. 22(1996), 247-277.

[13] Alonso, M. E., Becker, E., Roy, M. F., Wörmann, T., Zeros, Multiplicities and Idempotents for Zerodimensinal Systems. Proc. MEGA94, Birkhäuser (1996).

[14] Noro, M., Yokoyama, K., New methods for the change-of-ordering in Gröbner basis computation. Research Report ISIS-RR-95-8E, FUJITSU LABS, ISIS (1995).

[15] Noro, M., J. McKay, Computation of replicable functions on Risa/Asir. Proc. PASCO'97, ACM Press (1997), 130-138.

[16] Noro, M., Yokoyama, K., A Modular Method to Compute the Rational Univariate Representation of Zero-Dimensional Ideals. J. Symb. Comp. **28**/1 (1999), 243-263.