グレブナー基底の並列計算と連立代数方程式†

佐々木 建昭†† 竹島 卓†††

有理数体上の多項式イデアルのグレブナー基底（標準基底）の並列算法を提案し，それを用いた連立代数方程式の代数的解法を論じる．多項式イデアルのグレブナー基底の計算は従来の算法では大きな中間式膨張を引き起こすことがしばしばだが，最近，筆者らによって中国剰余定理に基づくモジュラ算法が提案され，中間式膨張問題が解決された．本論で提案する並列算法はこのモジュラ算法の並列化であり，簡単な構造の並列計算機（いわゆるモジュラ型並列計算機）で実行でき，しかも並列プロセッサ数にほぼ逆比例して計算時間が減少する理想的なものである．このことを実際的問題に対する逐次計算機上でのタイミング・データで実証する．本アルゴリズムはモジュラ型であるから，プログラミングの若干の工夫により，逐次計算機上でも擬似並列的に実行可能であり，そのための算法も提示する．これらの算法は連立代数方程式の代数的解法に直ちに応用できるが，その場合には問題の特殊性を利用した効率化が可能であり，それを指摘する．

1. はじめに

最近，数式処理の分野でも並列算法に関する研究が活発になってきたが[1]，それらは多項式あるいは行列や行列式に関するものが大部分であり，これらに対しては非常に効率良い算法の構成が可能である（たとえば文献 2) を見よ）．多項式や線形代数の演算では，それ自体が高い並列性を有していることが多いからである．これに対し，本稿では，一見しただけでは並列性が見て取れない連立代数方程式の代数的解法に対して，非常に効率良い並列算法を提示する．

連立代数方程式の代数的解法としては，現在のところ，多項式イデアルのグレブナー基底を計算する方法[3]が最も有効であるとされているが，この計算はしばしば猛烈な係数成長を起こすことが知られている．この係数成長は，連立代数方程式の低減化に通常用いられる辞書式順序において最も顕著に現れる．しかしながら，係数成長を観察すると，それは中間式に主に生じ，最終の式における係数の大きさはそれほどでないことが経験的に知られる．したがって，係数成長の問題はモジュラ算法により解決できると考えられる．非線形問題である連立代数方程式に対する我々の並列算法とは，このモジュラ算法の並列化であり，実は最も明白な並列性に着目したものである．

多項式イデアルのグレブナー基底のモジュラ算法としては，ヘンゼル構成を用いる方法が Trinks[4]と Winkler[5]により，中国剰余算法に基づく方法が筆者ら[6]と Traverso[7]により，それぞれ発表されているが，筆者らの方法のみが完全で，Traverso の方法は完全性のチェックに他の項順序で計算したグレブナー基底を必要とし，Trinks の方法は特殊な問題に対してのみ適用可能，Winkler の方法は一般問題に適用可能だが確率的で不完全である．2章ではグレブナー基底と連立代数方程式の代数的解法を手短に復習する．3章では筆者らのモジュラ・グレブナー基底算法を簡単に説明し，モジュラ型・並列グレブナー基底算法を提示する．4章では連立代数方程式に応用する場合の効率化について指摘し，実際的問題に対するタイミング・データで本算法の有効性を示す．5章では逐次計算機上で実行しうる擬似並列算法を提示する．

2. グレブナー基底と連立代数方程式

本論文では，多項式はすべて体あるいは環 K 上の多項式環 $K[x_1, \cdots, x_n]$ に属するものとする．K は，整数環 $\boldsymbol{Z}$，有理数体 $\boldsymbol{Q}$，素数 p_i を法とする剰余環 $\boldsymbol{Z}/(p_i)$（ガロア体），$\boldsymbol{Z}/(p_1 \cdots p_k)$，のいずれかであるとする．$K[x_1, \cdots, x_n]$ を簡単のため $K[x]$ と表すことにする．

多項式 $F_i, F_i \in K[x]$, $i=1, \cdots, r$, に対し，

$$\{u_1F_1+\cdots+u_rF_r \mid u_i \in K[x], i=1, \cdots, r\} \quad (1)$$

なる多項式集合を $F_1, \cdots, F_r$ によって生成される多項式イデアルといい，$(F_1, \cdots, F_r)$ と表す，集合 $\{F_1, \cdots, F_r\}$ をイデアルの基底という．多項式 $F \in K[x]$ を単項式の和で表したとき，

$$F=c_1T_1+c_2T_2+\cdots, c_i \in K, T_i=x_1^{i_1}\cdots x_n^{i_n}, \quad (2)$$

† A Parallel Gröbner Basis Method and Solving System of Algebraic Equations by TATEAKI SASAKI (The Institute of Physical and Chemical Research) and TAKU TAKESHIMA (FUJITSU LIMITED).

†† 理化学研究所情報科学研究室
††† 富士通(株)国際情報社会科学研究所

とする．このとき，指数の組 $(e_{i1},\cdots,e_{in})$ に適当な全順序を導入することにより，任意の各項を一意的に順序づけることができる．この順序を項順序といい，T_1 の方が T_2 より高順位であるとき $T_1 \triangleright T_2$ で表す．$\triangleright$ には辞書式順序，全次数順序などがある．(2)式において，$T_1 \triangleright T_2 \triangleright \cdots$ であるとき，c_1T_1 を頭項 (head term)，c_1 を頭係数 (head coefficient)，T_1 を頭べき積 (head power product) といい，それぞれ $\mathrm{ht}(F)$，$\mathrm{hc}(F)$，$\mathrm{hpp}(F)$ と表す．さらに，二つの多項式の各項を項順序によって先頭から順次比較していくことにより，項順序 $\triangleright$ は任意の多項式の間の半順序に自然な形で拡張される．

多項式 $G\in K[x]$ を $G=c_1'T_1'+c_2'T_2'+\cdots$，ただし $c_i'\in K$，$\mathrm{ht}(G)=c_1'T_1'$，とする．(2)の多項式 F のある単項 cT に対し，T が T_1' の倍数であるとき，

$$F-(cT/\mathrm{ht}(G))\cdot G=F' \qquad (3)$$

なる多項式 F' を構成すれば，$F\triangleright F'$ となる．これを F の G による M-簡約 (M-reduction) といい，$F\to_G F'$ と表す．なお，K が体のとき $K'=K$，K が環 $\boldsymbol{Z}$ のとき K' は有理数体 $\boldsymbol{Q}$ とする．このとき $F'\in K'[x]$ であることに注意されたい．F を G で可能な限り M-簡約し，どの項ももはや G で M-簡約できなくなったときの式を $\tilde{F}$ とする．$\tilde{F}$ は G に関して M-既約 (M-irreducible) であるといい，

$$F\to\!\!\to_G \tilde{F} \qquad (4)$$

と表す．同様に，F を多項式 $G_1,\cdots,G_s$ で M-簡約して M-既約にした式を $\tilde{F}$ とするとき，

$$F\to\!\!\to_\Gamma \tilde{F}, \qquad (4')$$

と表す．ただし，$\Gamma=\{G_1,\cdots,G_s\}$ である．多項式 F_1 と F_2 に対し，

$$\mathrm{Spol}(F_1,F_2)=\mathrm{hc}(F_2)\frac{lcm}{\mathrm{hpp}(F_1)}F_1-\mathrm{hc}(F_1)\frac{lcm}{\mathrm{hpp}(F_2)}F_2, \qquad (5)$$

lcm は $\mathrm{hpp}(F_1)$ と $\mathrm{hpp}(F_2)$ の最小公倍式，

なる式で計算される $\mathrm{Spol}(F_1,F_2)$ は $K'[x]$ の多項式となる．この多項式を S-多項式 (S-polynomial) という．多項式イデアル $(F_1,\cdots,F_r)$ に対し，多項式集合 $\Gamma=\{G_1,\cdots,G_s\}$ が次の二つの条件を満たすとき，Γ をグレブナー基底 (Gröbner basis) という．

(1) $(F_1,\cdots,F_r)=(G_1,\cdots,G_s)$, (6)

(2) 任意の $F\in(F_1,\cdots,F_r)$ に対し，

$$F\to\!\!\to_\Gamma 0. \qquad (7)$$

グレブナー基底 $\Gamma=\{G_1,\cdots,G_s\}$ が次の性質を満たすとき，正規グレブナー基底という．

(3) $i=1,\cdots,s$ に対し，G_i は $\Gamma-\{G_i\}$ に関して M-既約で $\mathrm{hc}(G_i)=1$.

与えられた $\{F_1,\cdots,F_r\}$ からグレブナー基底を構成する算法として，有名なブックベルガーの算法がある (たとえば文献 3) を見よ) が，それは以下のようなものである．初期設定として，$\Gamma=\{F_1,\cdots,F_r\}$ とおく；Γ の二つの要素，F_i と F_j，$i\neq j$，に対し，$\mathrm{Spol}(F_i,F_j)\to\!\!\to_\Gamma \tilde{F}_{ij}$ を計算する；$\tilde{F}_{ij}\neq 0$ ならば $\Gamma:=\Gamma\cup\{F_{ij}\}$ とする；これを繰り返し，Γ のあらゆる要素 F_i と F_j，$i\neq j$，に対して $\tilde{F}_{ij}=0$ となるとき，Γ はグレブナー基底である．

Γ を正規化するには，Γ のあらゆる要素に対して，他の要素で M-簡約し，M-簡約の結果 0 になるものは捨て，そして頭係数を 1 にするように各要素を頭係数で割ればよい．この構成手順は停止性をもつことが証明されている．この手順において，$\tilde{F}_{ij}$ が $F_1,\cdots,F_r$ の線形結合で表されることは容易に分かる．同様に，$i=1,\cdots,s$ に対し

$$G_i=u_{i1}F_1+\cdots+u_{ir}F_r,\ u_{ij}\in K'[x], \qquad (8)$$

と表すことができ，u_{ij} はグレブナー基底の構成に付随して構成できる．

次式で与えられる連立代数方程式を考える．

$$\{F_1(x_1,\cdots,x_n)=0,\cdots,F_r(x_1,\cdots,x_n)=0\}, \qquad (9)$$

ただし，$F_i\in\boldsymbol{Q}[x]$，$i=1,\cdots,r$，とし，方程式は (一般に複素数の) 解をもち，その個数は有限個と仮定する．この方程式を代数的手法で解くのは以下のようになる．

代数的フェイズ：連立方程式(9)を以下の形に変換する．

$$\{G_1(x_1)=0, G_{21}(x_1,x_2)=0,\cdots, G_{31}(x_1,x_2,x_3)=0,\cdots, G_{n1}(x_1,x_2,\cdots,x_n)=0,\cdots\}, \qquad (10)$$

ただし，$G_{ij}\in\boldsymbol{Q}[x_1,x_2,\cdots,x_i]$，$i=2,\cdots,n$，であり，$\mathrm{ht}(G_{ij})=c_{ij}x_i^{m_{ij}}$，$c_{ij}\in\boldsymbol{Q}$，である．なお，(10)式は以下の簡単な形式になることが多い．

$$\{G_1(x_1)=0, x_2-\tilde{G}_2(x_1)=0,\cdots, x_n-\tilde{G}_n(x_1)=0\}. \qquad (10')$$

数値的フェイズ：連立方程式(10) (あるいは(10′)) を数値的に解く．連立方程式(10)は数値解法に非常に好都合な形式であることに注意されたい．まず $G_1(x_1)=0$ を解く．この解のひとつ，それを α とする，を他の方程式に代入すれば，第2式は $G_{21}(\alpha,x_2)=0$ となり x_2 に関して数値的に解ける，以下等々．

したがって，問題は(9)式を(10) (あるいは(10′))

の形に変換することであるが，グレブナー基底の理論によると，$x_n \triangleright x_{n-1} \triangleright \cdots \triangleright x_1$ なる辞書式順序でイデアル $(F_1, \cdots, F_r)$ のグレブナー基底 Γ を計算すれば，$\Gamma = \{G_1, G_{21}, \cdots, G_{n1}, \cdots\}$ となることが証明されている．それゆえ，問題はグレブナー基底の計算に帰着される．

3. モジュラ・グレブナー基底算法とその並列化

前章で述べた数値的フェイズを並列化するのは極めて容易である．まず，$G_1(x_1)=0$ を解く．この根を $\alpha_1, \cdots, \alpha_\nu$ とするとき，各 α_i に対して $x_2, x_3, \cdots$を順に解いていく計算は全く独立であり，並列化できる．ν は x_1 の解の個数に等しく，一般にかなり大きな値になるから，この並列計算の並列度はほぼ ν である．以下では，代数的フェイズの計算のみを論じる．

まず，代数的フェイズの計算の基礎となるモジュラ・グレブナー基底算法を大雑把に解説する．多項式 $F_1, \cdots, F_r$ は $\boldsymbol{Q}[x]$ の要素とする．係数の分母を払うことにより，これらは $\boldsymbol{Z}[x]$ の要素とみなせる．したがって，素数 p_l に対して，多項式の係数をガロア体 $\boldsymbol{Z}/(p_l)$ に落としてグレブナー基底 $\Gamma^{(l)}$ を計算できる．この計算では係数成長が生じる余地はない．$\Gamma^{(l)}$ の構成は $\boldsymbol{Q}$ 上のグレブナー基底 Γ の構成と全く同じ手順によって行う．すなわち，以下の3条件を課す．

（a） 多項式 F を $\{F_1, \cdots, F_r, \cdots\}$ で M-簡約するときは，M-簡約に用いる多項式の選び方を同じとし，選び方は係数によらないものとする．

（b） グレブナー基底構成算法において，Spol(F_i, F_j) の F_i と F_j は係数によらない同じ選び方で選ぶとする．

（c） グレブナー基底の正規化を係数によらない同じ方法で行う．

これらの条件を満たすには，Γ と $\Gamma^{(l)}$ の構成を同じプログラムで係数演算だけを変えて行えばよい．

今，既約化された S-多項式 $F_{r+1}, F_{r+2}, \cdots$を順に計算して Γ が得られるとし，$\Gamma^{(l)}$ は同様に $F^{(l)}_{r+1}, F^{(l)}_{r+2}, \cdots$を順に計算して得られるとする．

ただし，F_j, $j=1,2,\cdots$, は(5)式で計算した S-多項式を M-簡約したもので，頭係数は1に規格化する前の多項式とする．このとき，

$$p_l \nmid \mathrm{num}(\mathrm{hc}(F_j)),\ j=1,\cdots,r,r+1,\cdots \qquad (11)$$

であるならば，素数 p_l はラッキー (lucky) であるといい，そうでないならばアンラッキー (unlucky) であるという．ここに num(-) は分数の分子整数を表す．グレブナー基底を計算する S-多項式の系列は有限であることが証明されているから，アンラッキーな素数の個数も有限であることが分かる．

M-簡約および S-多項式構成において，多項式の係数の分母に現れ得るのは $F_1, \cdots, F_r, F_{r+1}, \cdots$ の頭係数だけであるから，p_l がラッキーなとき

$$F_j \equiv F^{(l)}_j \pmod{p_l},\ j=1,\cdots,r,r+1,\cdots \qquad (12)$$

が成立し，相異なるラッキーな素数 $p_1, \cdots, p_k$ に対するグレブナー基底 $\Gamma^{(1)}, \cdots, \Gamma^{(k)}$ から，$\boldsymbol{Z}/(p_1 \cdots p_k)$ 上での基底 $\Gamma^{(0)}$ が中国剰余算法で構成できる．このとき，基底を正規化しておけば

$$\Gamma \equiv \Gamma^{(0)} \pmod{p_1 \cdots p_k} \qquad (13)$$

となる．(証明については，たとえば文献 3) を見よ．) $\Gamma^{(0)}$の要素の係数は整数であるが，一定の条件下で有理数に変換する．すなわち，以下の定理に基づいて変換する．

［定理］ 与えられた整数 u と正整数 m，ただし，$-m<u<m$，に対し，$u \equiv a/b \pmod{m}$ を満たす整数 a と b は，$|a|<\sqrt{m/2}$, $0<b<\sqrt{m/2}$ なる条件下で高々ひとつだけ存在する．□

したがって，こうして得られた $\boldsymbol{Q}$ 上でのイデアル基底 Γ' は，k が十分に大きいとき，Γ に一致するはずである．問題点は，p_l がラッキーでないときどうするか？，k をどのように決定するか？，整数の有理数化の効率的算法，などであるが，いずれも文献 6) で解決されている．本稿ではアンラッキーな素数 p_l についてだけ簡単に説明するに止める．

$\boldsymbol{Z}/(p_l)$ 上でのグレブナー基底 $\Gamma^{(l)}$ の構成において，j 番目に構成する S-多項式として Spol$(F_{j1}, F_{j2}) \to\to F^{(l)}_j$ を計算したとする．このとき，j と hpp $(F^{(l)}_j)$ の対をインデックス j に関する増加リストとして残しておく．このリストを頭項ヒストリ (head term history) と呼ぶことにする．素数 p_l に対する頭項ヒストリと素数 $p_{l'}$ に対する頭項ヒストリが異なっているとき，少なくともどちらかの素数がアンラッキーであり，より小さな j に対して頭項が消えた方の素数に対する計算を捨てる．こうして，$\Gamma^{(1)}, \cdots, \Gamma^{(k)}$ が同じ頭項ヒストリで計算できたとき，結果の Γ' が正しくないのは $p_1, \cdots, p_k$ がすべてアンラッキーな場合のみである．

$\boldsymbol{Z}/(p_1 \cdots p_k)$ 上での基底 $\Gamma^{(0)}$ を有理数変換して得られるイデアル基底を $\Gamma' = \{G'_1, \cdots, G'_i\}$ とする．Γ' に対してまず次の二つのチェックを行う．

チェックⅠ：Γ' の任意の要素 G'_i と G'_j に対し $\boldsymbol{Q}$ 上で

$$\mathrm{Spol}(G'_i, G'_j) \rightarrow\!\!\rightarrow_{\Gamma'} 0; \quad (14)$$

チェックⅡ：$\boldsymbol{Q}$ 上で

$$F_i \rightarrow\!\!\rightarrow_{\Gamma'} 0, \quad i=1,\cdots,r. \quad (15)$$

これらのチェックにパスすれば，Γ' は $(F_1,\cdots,F_r)\subseteq(G'_1,\cdots,G'_{s'})$ なる $\boldsymbol{Q}$ 上のグレブナー基底であることが分かる．つぎに，$\Gamma^{(l)}$ の計算に付随して，

$$G_i^{(l)} \equiv u_{i1}^{(l)}F_1+\cdots+u_{ir}^{(l)}F_r \pmod{p_l} \quad (16)$$

なる $u_{ij}^{(l)}\in \boldsymbol{Z}/(p_l)[x]$, $j=1,\cdots,r$, を計算する（前章（8）を参照）．$u_{ij}{}^{(l)}$, $l=1,\cdots,k$, に対しても中国剰余算法と整数の有理数化を施して，

$$G'_i \equiv u'_{i1}F_1+\cdots+u'_{ir}F_r \pmod{p_1\cdots p_k} \quad (17)$$

なる $u'_{ij}\in\boldsymbol{Q}[x]$ を構成する．そして以下をチェックする．

チェックⅢ：$\boldsymbol{Q}$ 上で

$$G'_i = u'_{i1}F_1+\cdots+u'_{ir}F_r, \quad i=1,\cdots,s'. \quad (18)$$

(18)が成立するとき，$(G'_1,\cdots,G'_{s'})\subseteq(F_1,\cdots,F_r)$ となり，チェックⅠ，Ⅱと合わせれば $\mathrm{ideal}(\Gamma')=\mathrm{ideal}(\Gamma)$，すなわち Γ' は $(F_1,\cdots,F_r)$ の $\boldsymbol{Q}$ 上でのグレブナー基底であることが主張できる．逆にチェックⅠ，Ⅱ，Ⅲのどれかでも失敗すれば，k の値は十分大きくないとして，さらに別の素数を取り出して計算を続行する．（いくつかの場合には，係数の上限が理論的に計算できるが，この上限は実際の上限よりはるかに大きいことが多く，理論的上限の利用は実際的でない．）アンラッキーな素数の個数は有限であるから，この手順は必ず停止する．

上記の計算手順を見れば，$\Gamma^{(1)},\cdots,\Gamma^{(k)}$ の計算は並列的に実行することができ，しかもラッキーな素数に対する計算はほぼ同時間で終了するはずである．$u_{ij}^{(l)}$ の構成も同様である．また，チェックⅠ，Ⅱ，Ⅲの計算も，$\mathrm{Spol}(G'_i,G'_j)$，F_i，G'_i に対してそれぞれ並列的に実行できる．さらに中国剰余算法と有理数化演算も各 G'_i に対して並列的に実行できる．すなわち，ほとんどの演算が並列的に実行できることが分かる．

以上より，並列プロセッサの個数が N であるときの並列グレブナー基底算法は以下となる．なお，以下では，並列プロセッサの割り当ては $\Gamma^{(1)}$ の計算に対してのみ具体的に示すとし，他の部分の計算に対しては簡単のために省略する．また，"**for** $i=a,\cdots,b$, **do parallelly** (**serially**)…" は指標 i に関して並列的（逐次的）に…を実行することを意味する．

Algorithm Parallel Modular-Gröbner.
Input : a set of polynomials $\Phi=\{F_1,\ldots,F_r\}\in Z[x_1,\ldots,x_n]$;
a set of distinct primes $\{p_1,p_2,\ldots\}$ of word-size;
a term-order $\triangleright$;
Output : a normalized Gröbner-basis $\Gamma=\{G_1,\ldots,G_s\}$ *w.r.t.* $\triangleright$;
Condition : number of parallel processors is N;

Step 0 [initialize] : P:=1; k:=0;
Step 1 [N Gröbner-bases over $\mathbf{Z}/(p_l)$, $l=kN+1,\ldots,kN+N$] :
for N distinct primes $p_l, l=kN+1,\ldots,kN+N$,
calculate parallelly
[$\Gamma^{(l)}$=normalized Gröbner-basis $\{G_1^{(l)},\ldots,G_s^{(l)}\}$ over $\mathbf{Z}/(p_l)$;*1)
$\Upsilon^{(l)}=\{u_{ij}^{(l)}\mid i=1,\ldots,s,\ j=1,\ldots,r\}$ **satisfying** (16);
$H^{(l)}$= head-term history for p_l];
Step 2 [check unlucky primes by using head-term history] :
for $l=kN+1,\ldots,kN+N$, **do serially**
if H=**nil then** $H:=H^{(l)}$
else if *2) $H\triangleright H^{(l)}$ **then** discard results for p_l
else if $H^{(l)}\triangleright H$ **then**
[discard the results for all the previous primes;
$H:=H^{(l)}$; P:=1];
Step 3 [Chinese remainder and integer-rational conversion]:
if there is no surviving prime in $\{p_{kN+1},\ldots,p_{kN+N}\}$ **then goto** Step 5;
for each surviving prime p_l in $\{p_{kN+1},\ldots,p_{kN+N}\}$, **do serially**
[P':=P; P:=$P\times p_l$;
for i=1,…, s, **do parallelly**
[**if** P'=1 **then** $G_i^{(0)}:=G_i^{(l)}$ and $u_{ij}^{(0)}:=u_{ij}^{(l)}$, $j=1,\ldots,r$
else by applying Chinese remainder algorithm,
update $G_i^{(0)}$ in $\Gamma^{(0)}=\{G_1^{(0)},\ldots,G_s^{(0)}\}$ and $u_{ij}^{(0)}$ in $\Upsilon^{(0)}$ so that
$G_i^{(0)}\equiv G_i^{(l)}$, $u_{ij}^{(0)}\equiv u_{ij}^{(l)} \pmod{p_l}$;
G'_i:= convert coefficients of $G_i^{(0)}$ to rationals;
u'_{ij}:= convert coefficients of $u_{ij}^{(0)}$ to rationals];
if conversion fails to some coefficients
then $\Gamma_{(P)}:=\Upsilon_{(P)}:=$ **nil and goto Step 5;**
$\Gamma_{(P)}:=\{G'_1,\ldots,G'_s\}$; $\Upsilon_{(P)}:=\{u'_{ij}\mid i=1,\ldots,s,\ j=1,\ldots,r\}$];
Step 4 [check the termination]:
if P=1 or $\Gamma_{(P')}=$ nil or $\Gamma_{(P)}\neq\Gamma_{(P')}$ or $\Upsilon_{(P)}\neq\Upsilon_{(P')}$ **then** goto Step 5;
4.1: **for** each pair $\{G'_i,G'_j\}$ in $\Gamma_{(P)}$, **check parallelly**
$\mathrm{Spol}(G'_i,G'_j)\rightarrow\!\!\rightarrow_{\Gamma_{(P)}} 0$ over $\mathbf{Q}$;
if the check fails **then goto Step 5;**
4.2: **for** i=1,…, r, **check parallelly**
$F_i\rightarrow\!\!\rightarrow_{\Gamma_{(P)}} 0$ over $\mathbf{Q}$;
if the check fails **then goto Step 5;**
4.3: **for** i=1,…, s, **check parallelly**
$G'_i=u'_{i1}F_1+\cdots+u'_{ir}F_r$ over $\mathbf{Q}$;
if the check fails **then goto Step 5;**
return $\Gamma_{(P)}$;
Step 5 [continue the computation]:
$k:=k+1$ and **goto Step 1.**

*1) 素数がラッキーであれば，異なる素数に対する計算はほぼ同期して終了する．したがって，どれかの素数に対する計算が長引いていればアンラッキーとして捨てればよい．
*2) 頭項ヒストリに対しても記号 $\triangleright$ を用いるが，これはヒストリの先頭から順に，収納されている頭べき積を $\triangleright$ で比較して順序をつけるものとする．

4. 計算効率に関する考察

アンラッキーな素数の個数は有限であり，一語長程度の素数がアンラッキーになる確率は小さいから，k 個の素数 $p_1, \cdots, p_k$ がすべてアンラッキーとなり，しかもそれらの頭項ヒストリが同じとなる確率は極度に小さいと言える．したがって，Step 4.2 をパスした段階でほとんどの場合に正しい基底が得られると言える．一方，以下のタイミング・データが示すように，計算時間の大部分は Step 1 で消費され，しかも Step 1 では $\Gamma^{(1)}$ の計算よりも $\Upsilon^{(1)}$ の計算の方により多くの時間がかかる．したがって，前章に与えた算法は極度に小さい確率でしか起こり得ない場合のために過半数の時間を消費するものである．

そこで，以下の確率的算法を考える．

モジュラ・グレブナー基底算法(確率的バージョン)

前章のモジュラ・グレブナー基底算法において，

1) $\Upsilon^{(1)} = \{u_{ij}^{(1)} \mid i=1, \cdots, s, j=1, \cdots, r\}$ の計算を省略；
2) Step 4.3 をスキップする．

確率的バージョンは，厳密に正しいグレブナー基底を計算するときにはもちろん使えないが，以下の理由により，連立代数方程式に対しては有用である．

(i) Step 4.1 と 4.2 のチェックをパスしたとき，$\Gamma_{(P)} = \{G_1', \cdots, G_s'\}$ は $(F_1, \cdots, F_r) \subseteq (G_1', \cdots, G_s')$ を満たす．したがって，$\{G_1'=0, \cdots, G_s'=0\}$ の解はすべて $\{F_1=0, \cdots, F_r=0\}$ の解となる．ほとんどの場合 $\Gamma_{(P)} = \Gamma$ であるから，欲しい解はほとんどの場合 $\Gamma_{(P)}$ から得られる．

(ii) 連立代数方程式においては，元の方程式から解の個数の上限を容易に決めることができる（いわゆるベズーの上限．実例については文献 8) を見よ)．もしもこの上限が $\{G_1'=0, \cdots, G_s'=0\}$ の根の個数に等しいならば，方程式(9)の解はすべて $\Gamma_{(P)}$ から得られる．なお，重根がある場合，根の個数は多重度を含めて勘定しなければならないからやっかいだが，重根がない場合，(必要ならば適当に線形変数変換をすることにより) 連立方程式(9)は(10′)の形に変換できることが示されており（証明については文献 8) を見よ)，根の個数とは $G_1(x_1)$ の次数にほかならない．

連立方程式(9)が(10′)の形に変換できる場合には，さらに以下の三つの効率的手法が適用できる．

(a) 項順序 $\triangleright$ として，辞書式順序でなく，

$$x_2, \cdots, x_n \triangleright x_1, \text{ および } \{x_2, \cdots, x_n\} \text{ に対しては全次数順序,} \tag{19}$$

を選ぶことができる（証明については文献 9) を見よ)．理論解析[10]および経験によれば，辞書式順序に比べて全次数順序は計算量が一般にはるかに少ない．

(b) Step 4.1 は実行しなくてよい．これはつぎの定理による．

[定理] $\mathrm{hpp}(F_i)$ と $\mathrm{hpp}(F_j)$ が共通因子をもたないとき，$\mathrm{Spol}(F_i, F_j) \to\!\!\to_{F_i F_j} 0$. □

(c) Step 4.2 は代入操作で置き換えることができる．M-簡約を繰り返すより，代入の方がより効率的である．

実際的問題で並列化が極めて有効であることを示そう．前章に述べた並列算法は並列計算機上でテストされたわけではないが，我々はモジュラ・グレブナー基底算法を数式処理システム GAL にインプリメントし，逐次計算機上でテストした．我々の並列算法はモジュラ型なので，その有効性は逐次計算機上のタイミング・データから十分に立証できる．テスト問題は，既にベンチマーク問題となったと言える，桂のスピングラス代数方程式である[11]．

Example (Katsura's equation # 4)

$$P_1 = 2(x_5^2 + x_4^2 + x_3^2 + x_2^2) + x_1^2 - x_1 = 0,$$
$$P_2 = 2(x_5x_4 + x_4x_3 + x_3x_2 + x_2x_1) - x_2 = 0,$$
$$P_3 = 2(x_5x_3 + x_4x_2 + x_3x_1) + x_2^2 - x_3 = 0,$$
$$P_4 = 2(x_5x_2 + x_4x_1 + x_3x_2) - x_4 = 0,$$
$$P_5 = 2(x_5 + x_4 + x_3 + x_2) + x_1 - 1 = 0.$$

この連立方程式では，ベズーの上限は $2\times2\times2\times2\times1 = 16$ であり，辞書式順序に関するグレブナー基底は(10′)の形で，$\mathrm{degree}(G_1) = 16$ となる．また，結果の式の係数は，分子と分母がそれぞれ 40 桁程度以下の有理数となる．上記の例に対し，項順序 $\triangleright$ として(19)をとり，$\Upsilon^{(1)}$ の計算を省略し，上記(b)と(c)の手法を用いてグレブナー基底を計算した．計算には大型機で約 26 秒強（逐次計算機であることに注意）を要したが，その内訳は以下のとおりであった．

Step 1：大きさ $\sim 5\times10^5$ の素数を 16 個使用し，全体で約 26 秒．

Step 3：全体で約 252 ミリ秒．

その他：無視できる程度．

Step 3 の計算量が少ないのは文献 6) で与えられた効率的方法のせいであり，3 章の算法を各多項式に対して逐次的に素直に計算すると約 2 秒を要する．Step 1 においては，用いた 16 個の素数はすべてラッキーで

あり，各素数に対するイデアル基底の計算は約 1.63 秒で終了した．したがって，並列化の効果は絶大であると言える．実際，並列プロセッサ 10 個の計算機を用いれば，上記問題は約 3.3 秒で答えが得られるはずである．

5. 擬似並列算法

4章で与えた例題をモジュラ算でなく有理数算で，同じ条件で計算したところ，10分の制限時間内には計算が終了しなかった．このことから，係数成長のすさまじさが理解できよう．したがって，我々のモジュラ算法は，たとえ逐次的に計算したとしても，大きな問題に対しては実際的に非常に有効な方法であると言える．しかも，並列算法の実行に必要な計算機は最も簡単な構造—いわゆるモジュラ型並列計算機—でよい．

残念ながら，我々は現在，上記の並列算法を実行しうる並列計算機をもたない．そこで以下では，並列計算機によらないで，逐次計算機上でのグレブナー基底の計算を効率化する擬似並列算法を提示する．

モジュラ・グレブナー基底算法を逐次計算機上で実行する場合に，ネックとなるのは多数個の素数に対しグレブナー基底を計算することである．ところで，素数 p_i と p_j がともにラッキーであるとき，$\Gamma^{(i)}$ と $\Gamma^{(j)}$ の計算は全く同じ手順となり，異なるのは多項式の係数の値だけである．しかも，一語長程度の素数がアンラッキーとなることは稀である．そこで，多項式 F の係数を長さ ν のリストとし，

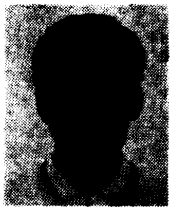
$$F=(c_{11}c_{21}\cdots c_{\nu 1})T_1+(c_{12}c_{22}\cdots c_{\nu 2})T_2+\cdots, \quad (20)$$

と表現する．ここに，$T_1, T_2, \cdots$は係数が1の単項式，c_{ij} は素数 p_i に対する T_j の係数である．そして，S-多項式は M-既約にしたあと，頭係数を常に1に規格化する．こうしておけば，M-簡約において係数演算が複雑となることはなく，異なる素数に対する M-簡約が全く同じ手順となる．さらに計算途上で，ある素数 p_j が他の素数に比べてアンラッキーであることが判明すれば，その時点で p_j に対する係数を **nil** とおき，**nil** とされた係数に対する演算は以後実行しない．このことにより，異なる素数に対する計算の同期終了性が保証される．以上より，表現(20)を用いて，ひとつのグレブナー基底の計算で同時に（アンラッキーの場合も含めて）ν 個分のグレブナー基底が計算できる．素数 p を法とするモジュラ演算では，グレブナー基底の計算に占める係数演算の割合は少ないから，上記の工夫は大きな効率向上をもたらすと期待できる．

参考文献

1) Ponder, C. G.: Parallelism and Algorithms for Algebraic Manipulation: Current Work, *SIGSAM Bull.*, Vol. 22, pp. 7-14 (1988).
2) Sasaki, T. and Kanada, Y.: Parallelism in Algebraic Computation and Parallel Algorithms for Symbolic Linear Systems, *Proc. SYMSAC '81* (ed. Wang, P. S.), ACM, pp. 160-167 (1981).
3) Buchberger, B.: Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory, in *Multidimensional Systems Theory* (ed. Bose, N. K.), pp. 184-232, D. Reidel Publ. Comp. (1985).
4) Trinks, W.: On Improving Approximate Results of Buchberger's Algorithm by Newton's Method, *SIGSAM Bull.*, Vol. 18, pp. 7-11 (1984).
5) Winkler, F.: p-adic Methods for Computation of Gröbner Bases, paper presented at EUROCAL '87, June 1987, also in *JSC*, Vol. 6, Nos. 2 & 3, pp. 287-304 (1988).
6) Sasaki, T. and Takeshima, T.: A Modular Method for Gröbner-basis Construction over $\boldsymbol{Q}$ and Solving System of Algebraic Equations, to appear in *J. of Inf. Process.*, Vol. 12, No. 4 (1990); see also, A Modular Gröbner-basis Method for Algebraic Equations, in Reports of RIKEN Symposium (Feb. 1988).
7) Traverso, C.: Gröbner Trace Algorithm, *Proc. ISSAC '88* (*Lecture Notes in Comp. Sci.*, Vol. 358), pp. 125-138 (1988).
8) Kobayashi, H., Moritsugu, S. and Hogan, R. W.: On Solving Systems of Algebraic Equations, paper presented at ISSAC '88, Rome, July, 1988, to appear in *JSC*, Vol. 8, No. 6 (1989)
9) Sasaki, T.: Some Algebraic Algorithms Based on Head Term Elimination over Polynomial Rings, *Proc. EUROCAL'87* (*Lecture Notes in Comp. Sci.*, Vol. 378), pp. 348-354 (1987).
10) Buchberger, B.: A Note on the Complexity of Constructing Gröbner-bases, *Proc. EUROCAL '83*, pp. 137-145 (1983).
11) Katsura, S., Fukuda, W., Inawashiro, S., Fujiki, N. M. and Gebauer, R.: Distribution of Effective Field in the Ising Spin Glass of the $\pm J$ Model at $T=0$, *Cell Biophysics*, Vol. 11, pp. 309-319 (1987).

(平成元年6月 7 日受付)
(平成元年9月 12日採録)



佐々木建昭（正会員）

昭和21年4月1日生．昭和43年大阪大学物理学科卒業．昭和48年東京大学博士課程修了（素粒子論専攻）．理学博士．昭和49年より理化学研究所勤務，現在情報科学研究室研究員．昭和53年より10か月，米国ユタ大学計算機科学科の客員研究員．情報科学（数式処理のアルゴリズムとシステム開発，数値・数式の融合計算と算法，など），応用数学，計算物理，コンピュータアート，マラソン（自己記録2時間41分51秒，東大伊豆マラソン7連勝），エッセイに興味をもつ．岩波講座情報科学，岩波情報科学辞典など執筆．日本物理学会，日本ソフトウェア科学会，ACM 各会員．

竹島　卓（正会員）

1948年生．1971年東京工業大学理工学部制御工学科卒業．1973年同大学院修士課程，1976年博士課程修了．同年富士通(株)国際情報社会科学研究所に入所，現在に至る．ロボット，データベース，計算機言語，数式処理に興味をもつ．日本ソフトウェア科学会，ACM 各会員．