代数幾何と数式処理

高山信毅

1　互除法とグレブナ基底

本屋さんへいくと, Mathematica とか Maple の解説本が山積みになっている. 喜ばしいことである. が, ここでこれらの紹介をするわけではない. この解説ではもっと基礎的な話 (裏方ばなし?), つまり “計算代数” とか, “計算代数解析” などの研究の最前線を紹介しようというのが目的である. これらの研究はこの 10 年間に長足の進歩をとげ, 著者の力量不足により, 全てを紹介することは到底できない. この解説では, 自由分解の計算とその応用に焦点をしぼる. 自由分解はさまざまな不変量の計算のための基礎として重要である.

さて, 代数的な問題の一般的な解法を考えているのが, 計算代数である.

商品として売っている数式処理ソフトは, 計算代数の研究の成果のうち, 発売元が商品として必要と思う機能を取り込んで販売している. たとえば, `GCD[7,11]` と入力すれば 7 と 11 の最大公約数を計算してくれて答えを返してくれる.

話をここからスタートしよう. 二つの数 a, b $(a > b)$ の最大公約数を求める方法を考えてみよう.

一番単純な方法: k を GCD の候補とする. $k = b$ とおき k で, a, b を割ってみる, 両方とも割り切ることができなくてだめなら, $k - 1$ で a, b を割ってみる, だめなら, 今度は, $k - 2$ でと繰り返していって a, b 両方を割り切れれば, それが最大公約数である. これが計算代数の研究だとすると, ここでまず第一の感激がある. われわれは, 二つの数の最大公約数を見つける “一般的な解法” を見つけたのである. “計算可能” な対象を明らかにし, さらにその計算方法 (アルゴリズム) を与えるのが, 計算代数の一つの目的である.

次に, この計算方法をプログラムとして書いてみる. プログラムとして書く前には, 膨大な量の手計算をやってみたほうが, はるかに良いプログラムが書けるし, バグとりも楽である. アルゴリズムの本質的改良や新しい発見があることもおおいので, すぐコンピュータにやらせようとするのは損である. が, とにかく無事プログラムが動き, 実際に数をいれて, 大量の答えがでてくるのをみるのは, 第 2 の感激である.

次にアルゴリズムの効率を良くするというおもしろい問題がまっている. さきほど与えた解法では, 最悪の場合, $2b$ 回の割算が必要である. つまり, 入力された数字の桁数を n としたとき, 大体 e^n に比例する回数の割算が必要となる. さて, この問題の場合には, ユークリッドの互除法をもちいるとアルゴリズムの効率が格段によくなる. つまり, $a = qb + r$, $r < b$ と割算し, b より小さい余り r を求める. (r を a の b によるリダクションの標準形と呼ぶ). このとき, $gcd(a, b) = gcd(b, r)$ となる. こんどは, $b := r, a := b$ とおいて同じ計算を繰り返す. これをくりかえすと有限回の割算で余りが 0 となり, GCD を求めることができる. したがって, GCD の計算は高々 $\log b$ に比例する回数, つまり b の桁数に比例する回数の割算で終了することになる. なお, となりあう フィボナッチ数を a, b としたとき互除法の計算はいちばん長くなる. この計算の副産物として, 整数係数の一次不定方程式を解くこともできる. また, 一変数多項式環においても互除法は機能する. このときは, 割算での $>$ は次数の比較を意味する. 互除法のさらなる効率化は (楕円暗号の効率化からの要請もあり), 現在も研究されていることを注意しておく.

さて次の節では自由分解の構成アルゴリズムを説

明するがその基礎となるのは, グレブナ基底 (または standard basis) を計算するための, Buchberger アルゴリズムである. これについてのやさしい解説はいたるところでみかけるので, 詳しくはこれらの解説や本格的には [1], [3], [8] などを見ていただくことにして, いくつか私が要点と思うことを説明しておく. Buchgerger アルゴリズムは, 一変数多項式の世界では, ユークリッドの互除法にほかならない. 互除法の適用にはモノミアルの間に順序をいれることが必要である. モノミアルの間の順序としては, たとえば辞書式順序 (lexicographic order) $>_{lex}$ がある. つまり, $v = (v_1, v_2, \ldots), u = (u_1, u_2, \ldots)$ とするとき, $v >_{lex} u$ を $v_i - u_i$ のなかで最初の 0 でない元が, 正であると定義し, さらに $x^v >_{lex} x^u \Leftrightarrow v >_{lex} u$ と定義する.

一変数では, 次数が互除法で大事な役目を果たしたが, 多変数では, 重みベクトル $w \in \mathbf{R}^n$ を用意して, たとえば,

$$x^a >_w x^b \Leftrightarrow$$
$$w \cdot a > w \cdot b \text{ または } (w \cdot a = w \cdot b \text{ かつ } a >_{lex} b)$$

で順序を定義する. このようにして, モノミアルの間に全順序がはいる. Buchberger アルゴリズムはこの順序に関する互除法のようなものである. さて, f を多項式とするとき, f にあらわれるモノミアルのなかで, 順序 $>_w$ で最高次数のモノミアルをイニシァルモノミアルとよび, $\mathrm{in}_{>_w}(f)$ と書く. また $\mathrm{in}_w(f)$ で, 重み w に関する最高次の項をあつめた多項式をあらわす. Buchgerger アルゴリズムをもちいると, イデアル I に対して, モノミアルイデアル $\mathrm{in}_{>_w}(I)$ の生成元を計算できる. イデアルの不変量の計算を, $\mathrm{in}_{>_w}(I)$ または $\mathrm{in}_w(I)$ の計算に帰着させるのが, 計算代数の大原則の一つである. これは, 微分方程式論における摂動法の代数版とおもってもよい. $\mathrm{in}_w(I)$ が I の第一近似である. たとえば, ミラー対称性の理論でてくる B モデルの超幾何関数を計算するときには, $\mathrm{in}_w(I)$ からの摂動計算で計算する ([5], [11]).

グレブナ基底の計算例を一つだけあげておく.

$$W = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

をもちいて, $\mathbf{Q}[x, y, z]$ につぎのように順序 (degree reverse lexicographic order) をいれる.

$$x^a y^b z^c >_W x^A x^B x^C$$
$$\Leftrightarrow \quad W({}^t(a \quad b \quad c) - {}^t(A \quad B \quad C)) >_{lex} 0.$$

イデアル

$$I = \langle x^2 + y^2 + z^2, xy + yz + zx, xyz \rangle$$

の順序 $>_W$ に関するグレブナ基底は,

$$\underline{x^2} + y^2 + z^2,\ \underline{xy} + xz + yz,\ \underline{xz^2} + yz^2,$$
$$\underline{y^3} - x^2 z - xyz + yz^2,\ \underline{-y^2 z^2} + xz^3 + yz^3,\ \underline{-z^4}$$

となる. 下線でしめしたのは, $\mathrm{in}_{>_W}(I)$ を生成するモノミアルである.

2 自由分解の構成

さて, 以下ではグレブナ基底の初歩を知っていると仮定して話を進める.

S を $\mathbf{Q}$ 上の多項式環とし, I を S のイデアルとしよう. S/I の自由分解を構成する効率のよいアルゴリズムを説明するのがこの節の目的である. 記号を導入しよう. 自由分解を次のようにかくことにする.

$$\cdots \xrightarrow{\varphi_3} F_2 \xrightarrow{\varphi_2} F_1 \xrightarrow{\varphi_1} F_0 \xrightarrow{\varphi_0} S/I \longrightarrow 0$$

ここで, $F_0 = S$ であり, F_i は, ランク r_i の S 自由加群 S^{r_i} に等しいとする. また, φ_i は, $r_i \times r_{i-1}$ 行列であり, 成分は S の元である. 自由分解だから, φ_i の像は, φ_{i-1} のカーネルに等しい (このように写像をつくっていくのが, 自由分解の定義). また, F_i の標準基底を, $e_0, e_1, \ldots, e_{r_i - 1}$ と書くことにする. e_k たちは横ベクトルと理解し, 行列 φ_i には, 左より掛

けるものとする. 与えられたイデアル I に対して自由分解は必ず存在するが, 一意的とは限らない.

φ_i の像が, φ_{i-1} のカーネルに等しいように写像をつくっていくのが, 自由分解の定義であるから, 下の方の φ_1 から, φ_{i-1} まで自由分解を構成したら, 一次不定方程式,

$$s_0\varphi_{i-1}(e_0)+\cdots+s_{r_{i-1}-1}\varphi_{i-1}(e_{r_{i-1}-1})=0$$

の解 (シジジーとよぶ), $(s_0,\ldots,s_{r_{i-1}-1})$ の生成元たちを計算して, それを φ_i の各行とすればいい. 多項式環でこの答えはグレブナ基底の計算ででる. これは整数係数の一次不定方程式の整数解をもとめるには互除法を用いればできるということの一般化である.

さきほどあげた, イデアル I に対する自由分解の例を表 1 にあげておく.

φ_i の各行ベクトル (よこベクトル) を順番に c_{ij}, $(j=0,1,\ldots,r_i-1)$ とおき, $c_{ij}=\varphi_i(e_j)$ の集合を C_i と書くことにする.

さて, 図 1 に示した自由分解は Schreyer 分解とよばれている分解になっている. この分解とは何かを説明するために, S の順序 $<_0:=<_W$ から出発して, 各 F_i に次のような Schreyer 順序 $<_i$ を帰納的に導入しよう.

$$\begin{aligned} & s\cdot e_i <_k t\cdot e_j \\ \Leftrightarrow\quad & s\bar{\varphi}_k(e_i) <_{k-1} t\bar{\varphi}_k(e_j) \\ & \quad \text{or } (s\bar{\varphi}_k(e_i) =_{k-1} t\bar{\varphi}_k(e_j) \text{ and} \\ & \qquad i>j \end{aligned}$$

ここで, s,t は, モノミアルであり, $\bar{\varphi}_k(e_i)$ は $\varphi_k(e_i)=c_{ki}$ の順序 $<_{k-1}$ での最高次の項を表す. さて, 自由分解があたえられたとき, 各 $C_k=\{\varphi_k(e_0),\ldots,\varphi_k(e_{r_k-1})\}$ が Schreyer 順序 $<_{k-1}$ での, グレブナー基底になっているときその分解を Schreyer 自由分解とよぶ.

図 1 の分解は, 1 節の最後にあげた W できまる degree reverse lexicographic order から出発した, Schreyer 自由分解となっている. たとえば, $\varphi_i\varphi_{i-1}=0$ となっているのはすぐたしかめられるであろう. 下線で示してあるのは, Schreyer 順序での, 最高次の項である.

Schreyer 順序は, 自由分解の効率的構成にとても重要である. その基礎となるのは, 次の定理である. "$F_{k-1}\supset C_k$ が順序 $<_{k-1}$ での, グレブナー基底になっているとしよう. このとき, S-ペア $sp(c_{ki},c_{kj})$ を C_k で割算 (リダクション) してえられる S-ペアの表現を

$$sp(c_{ki},c_{kj})=t_ic_{ki}-t_jc_{kj}=\sum_{p=0}^{r_k-1}s_pc_{kp}$$

とすれば,

$$-t_ie_i+t_je_j+\sum_{p=0}^{r_k-1}s_pe_p$$

達が, C_k のシジジーの生成元 (C_{k+1} にほかならない) となっており, さらに, 順序 $<_k$ での, グレブナー基底になっている."

この定理のおかげで, Schreyer 分解の計算は, 最初一発グレブナー基底を計算さえすれば, あとは, 順序をとりかえつつリダクションしていくだけである. また, 分散計算をすることも比較的容易である.

この方法は実はもっと効率よくできる. `Macaulay 2` ([4]) や, `kan/sm1` ([12]) が採用している計算方法を簡単に説明しよう ([6]). この方法では, まず, イデアル I のグレブナ基底 G を計算する. 次に, G のイニシャルモノミアルたちの生成するイデアルの Schreyer 分解を計算する. この分解を構成する行列を $B_1,B_2,\ldots$ とおく (図 2). 次にこの B_i 達をみて, リダクションすべき S ペアを読み取るのであるが, リダクションを下から順番にやらず, 次に定義するような, s-degree の順に計算していく. リダクションで 0 まで変形していくことに失敗したら, 先に紹介した定理のおかげで自動的にそれは一つ下のレベルのまだもとめていないグレブナ基底の元となる. つまり, うまくいくと計算の手間を半分に減らすことが可能となる.

簡単のためにイデアルの生成元は同次式としよう. s-degree を次のように帰納的に定義する.

$$\text{s-degree}\,(t\cdot e_i)=\deg(t)+\text{s-degree}\,(\varphi_k(e_i)).$$

$$
\begin{aligned}
\varphi_1 &= {}^t(\underline{x^2}+y^2+z^2 \quad \underline{xy}+xz+yz \quad \underline{xz^2}+yz^2 \quad \underline{y^3}-x^2z-xyz+yz^2 \quad \underline{-y^2z^2}+xz^3+yz^3 \quad \underline{-z^4}) \\
\varphi_2 &= \begin{pmatrix}
\underline{y} & -x & 0 & -1 & 0 & 0 \\
\underline{z^2} & z^2 & -x-2z & 0 & 1 & 1 \\
0 & \underline{-z^2} & y & 0 & 1 & 0 \\
0 & 0 & z^3 & 0 & \underline{-z^2} & y^2 \\
0 & 0 & \underline{z^2} & 0 & 0 & x+y \\
-z^3 & 0 & 0 & \underline{-z^2} & -y-2z & -2x-3y-z \\
0 & 0 & \underline{y^2}-2xz & -z^2 & x & -y \\
-xz-yz & \underline{y^2}+z^2 & 0 & -x & 0 & 0
\end{pmatrix} \\
\varphi_3 &= \begin{pmatrix}
\underline{z^2} & -y-2z & -x-y-2z & 0 & -4 & -2 & 1 & 0 \\
2z^3 & -2z^2 & \underline{y^2}-2xz-z^2 & 1 & -2x+y-5z & -x+y-2z & -y & z^2 \\
0 & z^3 & z^3 & -x-y & \underline{y^2}+2z^2 & z^2 & -z^2 & 0
\end{pmatrix}
\end{aligned}
$$

図 1: 自由分解

$$
\begin{aligned}
B_1 &= {}^t(x^2 \quad xy \quad xz^2 \quad y^3 \quad -y^2z^2 \quad -z^4) \\
B_2 &= \begin{pmatrix}
\underline{y} & \underline{-x} & 0 & 0 & 0 & 0 \\
\underline{z^2} & 0 & \underline{-x} & 0 & 0 & 0 \\
0 & \underline{z^2} & \underline{-y} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \underline{-z^2} & \underline{y^2} \\
0 & 0 & \underline{-z^2} & 0 & 0 & \underline{-x} \\
0 & 0 & 0 & \underline{-z^2} & \underline{-y} & 0 \\
0 & 0 & \underline{-y^2} & 0 & \underline{-x} & 0 \\
0 & \underline{y^2} & 0 & \underline{-x} & 0 & 0
\end{pmatrix} \\
B_3 &= \begin{pmatrix}
\underline{z^2} & \underline{-y} & x & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \underline{y^2} & 0 & 0 & x & -y & \underline{-z^2} \\
0 & 0 & 0 & x & \underline{y^2} & 0 & \underline{-z^2} & 0
\end{pmatrix}
\end{aligned}
$$

図 2: モノミアルイデアルの分解

ここで, t は多項式であり, deg は通常の次数である. また, s-degree $(\varphi_0(e_0))$ は 0 と決める.

図 2 を見てみよう. 下線部は, S ペアを作るためのモノミアルのペアである. さて B_1 および B_2, B_3 の下線部のモノミアルのペアで左にあるもの te_k の, s-degree $(te_k)-$level (te_k) を計算して表にしよう. ここで, $te_k \in B_j$ のとき, level を j とおく. この表は次のようになる.

$$
\begin{array}{lcccccccc}
B_1: & 1 & 1 & 2 & 2 & 3 & 3 & & \\
B_2: & 1 & 2 & 2 & 4 & 3 & 3 & 3 & 2 \\
B_3: & 2 & 3 & 4 & & & & &
\end{array}
$$

この表の次数の低いものから順に S ペアのリダクションを計算していくのがポイントである. もうすこし詳しく説明しよう. たとえば, B_1 の次数 1 のもののリダクション計算が終了したら, つぎに B_2 の次数 1 のものの計算にうつる. ここでなにがおこるか観察してみよう. 計算すべきは, S ペア $yc_{10}-xc_{11}$ のリダクションである. このばあいは,

$$yc_{10}-xc_{11} = \underline{y^3}+yz^2-x^2z-xyz$$

がすでにリダクション不能で既約である. したがってこの元 $y^3+\cdots$ は, 自動的に C_1 の元であり, c_{13} となることがわかる. さらにシジジー $ye_0-xe_1-1\cdot e_3$ を得る. 我々は, シジジーと, 一つ前のレベルのグレブナ基底の元 (一般にはシジジー) を一度に得たことになる.

またリダクションのときにこのような “2 倍お得” 現象がおきない S ペアからでてくるシジジーをあつめてくると, 同次イデアルの場合, 極小分解になる. 極小自由分解の具体的構成は, 組合せ論や可換環論,

代数幾何などでいろいろあたらしい問題を提起しており盛んに研究されている. MEGA (計算代数幾何の研究集会) の報告集 (最近は Journal of Pure and Applied algebra の特別号で毎年でてる) などをみると最近の研究の様子がわかるであろう. また, この計算でも $\mathrm{in}_w(I)$ を第 1 近似として計算していくという大原則が生かされている.

アルゴリズムで計算することおよび効率追求のおもしろさを説明しようとして, 細かい式を沢山書いてしまって申しわけない. しかしおもしろさを体感するには, 計算用紙とコンピュータを用意してここに挙げた材料を自分で実際計算してみるのが一番いいかもしれない. 初めて 2 次方程式の公式を使った時の感激がまた味わえることを期待する.

3 自由分解の応用 — de Rham コホモロジ群の計算

極小分解以外の自由分解の応用としては, ローカルコホモロジ群の計算, de Rham コホモロジ群の計算などがある. ここでは, de Rham コホモロジ群の計算法を例を用いて簡略に雰囲気のみ説明しよう. de Rham 理論というのはトポロジーで現れる量を微分形式の空間を用いて表現しようという理論である. したがって, 微分作用素の空間を扱う必要がある. 微分作用素環

$$\mathbf{Q}\langle x_1,\ldots,x_n,\partial_1,\ldots,\partial_n\rangle$$

を D_n と書くことにする. いままではもっぱら多項式環の話をしてきたが, グレブナ基底の理論自身は微分作用素環でほとんど変更なしに成立する. ただし, 微分作用素環では左イデアルと右イデアルの区別があるため, 両方がでてくると本質的な違いが現れる. 直感的にいうとグレブナ基底には無限個の元が必要となりこまる. しかし, そのような場合でも b-関数の根を用いると, そのなかよりいろいろな不変量の計算に必要な有限個をとりだすやりかたがわかる. この重要なアイディアは大阿久により与えられ ([9]), その後の D_n-加群のアルゴリズムの基礎となった.

さて我々の得ている de Rham コモホロジ群に対する定理の一つは以下のとおり ([10]).
“ 任意の $f \in \mathbf{Q}[x_1,\ldots,x_n]$ に対して, $H^k(\mathbf{C}^n \setminus V(f),\mathbf{C})$ は計算可能.”

計算可能という意味は, 我々は計算するためのアルゴリズムをもっているという意味である (ただし, こういういいかたをするのは, 暗に効率はよくなくて大きい問題は解けないよといいたいからである). このアルゴリズムは, 二つのステップより構成されている; (1) $\mathbf{Q}[x,1/f]$ の D-加群としての表示 D/I を求める. (2) D/I を D-加群の意味で “積分” する. 積分は $D/\hat{I}$ の自由分解を “切る” ことで計算できる. ここで, $\hat{I}$ は I の形式的なフーリエ変換. どこまで切るのかを決めるのが, b-関数の根である. “切る” ことで複体の完全性がこわれコホモロジ群がでてくる.

このアルゴリズムを例で説明しよう.
例:

$\mathbf{Q}[x]$ の多項式 $f = x(1-x)$ を考える. つまり, $\mathbf{C}\setminus\{0,1\}$ のコホモロジ群の計算をしたい. A でワイル代数 $\mathbf{Q}\langle x,\partial_x\rangle$ をあらわす. また, $F_k = \mathbf{Q}[x\partial_x]\partial_x^k$, $(k>0)$, $F_k = \mathbf{Q}[x\partial_x]x^k$, $(k\le 0)$ とおく. まず, $\mathbf{Q}[x,1/f] \simeq A/\langle p\rangle$, $p = x(1-x)\partial_x - (2x-1)$ となる. p の形式フーリエ変換は $\hat{p} = -x\partial_x^2 - x\partial_x$ である. $-x$ を左より掛けることにより

$$-x\hat{p} = \theta(\theta-1) + x\theta, \quad \theta = x\partial_x$$

をえる. したがって, この場合の b-関数 (常微分方程式の古典的な Frobenius の解法にでてくる $x=0$ での 特性多項式にほかならない) は $s(s-1)$ である. $A/\langle\hat{p}\rangle$ の自由分解は

$$0 \longrightarrow A \xrightarrow{\cdot(x\partial_x^2 - x\partial_x)} A \longrightarrow A/\langle\hat{p}\rangle \longrightarrow 0$$

となる. b-関数の根と自由分解の境界写像の次数を用いて不必要な高次部分を打ち切った複体は

$$0 \longrightarrow F_0/(F_{-1}+xA) \xrightarrow{\cdot(x\partial_x^2 - x\partial_x)} F_1/(F_{-1}+xA) \longrightarrow 0$$

である. われわれは, これを $\mathbf{Q}$ 上のベクトル空間の

複体と思う.

$$F_0/(F_{-1}+xA)=\mathbf{Q},\ F_1/(F_{-1}+xA)=\mathbf{Q}+\mathbf{Q}\partial_x$$

であり, また $F_1/(F_{-1}+xA)$ において $1\cdot(x\partial_x^2-x\partial_x)\equiv 0$ なので $H^{-1}=F_0/(F_{-1}+xA)=\mathbf{Q}$, $H^0=F_0/(F_{-1}+xA)=\mathbf{Q}^2$ となることがわかる. よって $U=\mathbf{C}\setminus\{0,1\}$ のコホモロジ群は

$$H^0(U,\mathbf{C}_U)=\mathbf{C},\quad H^1(U,\mathbf{C}_U)=\mathbf{C}^2$$

となる. ホモロジ群とコホモロジ群のポワンカレ双対性により, $x=0$ をまわる道および $x=1$ をまわる道が 1 次元コホモロジ群の生成元と対応する.

自由分解の計算によりその他いろいろなものが計算できるというのは現在進行形の話題である. こういった応用を読者の方はどのように思うであろうか? 私自身は, アルゴリズムで計算することが出来ること自体がおもしろい (たとえば, [2] を読むと私はうきうきしてくる). こういった問題意識と好奇心で数学をやる人ももっと沢山いてほしいなと思っている.

4 数学ソフトウエア

さて最後にソフトウエアの製作について述べておこう. アルゴリズムの研究とシステムの研究は相互関係しながら発展している. 商用数式処理システム, 商用数値解析システムに載せてないような, 研究レベルのアルゴリズムや数学システムに関する新しい設計概念を実装したシステムを数学ソフトウエアと私はよんでいる. 数学ソフトウエアというのは数学と計算機科学にまたがる分野でありあまり認知されてないかもしれない. だから, これを勉強すれば研究のスタートに立てますという標準的な教科書はない. でも確かに一つの分野として存在しているようである. たとえば, 昨年の暮れ神戸で, “数学ソフトウエア” と題する研究集会を開いて何人かの “数学プログラマー” も参加したが, これらの人とはやってる数学の内容がちがっていても話があうし, 共通の問題意識があり, また議論がかみあうのである.

いくつかの代数的な数学ソフトに関しては, [7] に紹介がある. また以下で言及しているような私の使用しているライブラリ, 数学ソフトなどは [12] からのリンクを見て下さい.

さて勝手流で数学ソフトを書くのも大歓迎だか, 本格的な (代数) 数学ソフトを書けるようになるには, 数学の知識はもちろんであるが, まず, C, C++, Lisp, SmallTalk, Mathematica, Java などの言語の知識は常識として, unix やコンパイラの基礎知識があると便利である. ガーベッジコレクタなどの知識もあると便利だが, Boehm, Demers の便利なガーベッジコレクタライブラリがあり私は愛用している. また代数的な数学ソフトを書くには, 整数計算のアルゴリズム, 多項式計算のアルゴリズムの基礎知識は基本として知っておくとよい. これらをどこまで究めるかは悩ましいが, Knuth の “基本算法” は整数計算に関する古典である. また, 整数計算のためのライブラリも多数存在する. 多項式計算のアルゴリズムの本はいろいろあるが, システム開発者または開発者と親しくしている人の書いた本やノートは役に立つことが書いてあることが多い. また最近は開放型システムが盛んに研究されているので, 通信に関する基礎知識などがあると便利である (誌面を借りて宣伝しておくと, `open asir` なる開放型数式処理系を富士通研の野呂さんと開発中であり, これが印刷されるころには配布を開始していると思う). あとは先人のソースコードを読んで, 数学的な対象をどのようなデータ構造で表現しているのかを調べてみたり, ソースコードに, `print` 文を入れてアルゴリズムの動きを観察したりするといい. ただ残念ながら, あまり文書化はきちんと進んでいないので, 最終的には, 弟子入りしたり共同研究するのが簡単だろう.

参考文献

[1] Adams, W.W. and Loustaunau, P., *An Introduction to Gröbner Bases*, American Mathematical Society, Province, 1994.

[2] アシモフ, ファウンデーション. 早川文庫.

[3] Eisenbud, D., *Commutative Algebra with a View Toward Algebraic Geometry*, Springer-Verlag, New York, 1995.

[4] Grayson, D. and Stillman, M., Macaulay 2, `http://www.math.uiuc.edu/Macaulay2`

[5] Hosono, S., Lian, B.H., Yau, S.-T., Maximal degeneracy points of GKZ systems, Journal of the American Math. Society 10: 427–443, (1997).

[6] La Scala, R. and Stillman, M., Strategies for computing minimal free resolutions, to appear in Journal of Symbolic Computations.

[7] 野呂, フリーの数学ソフトあれこれ. 数学セミナー 6 月号, 28–29 (1997).

[8] 情報処理 2 月号 (1998), 情報処理学会.

[9] Oaku, T., Algorithms for b-functions, restrictions, and algebraic local cohomology groups of D-modules. Advances in Applied Mathematics, **19** (1997), 61–105.

[10] Oaku, T. and Takayama, N., An algorithm for cohomology groups of the complement of an affine variety via D-module computation, `http://xxx.lanl.gov, math.AG/9801114`

[11] Sturmfels, B., *Gröbner Bases and Convex Polytopes*, American Mathematial Society, University Lecture Series 8, Providence, RI., 1995.

[12] Takayama, N., *Kan: A system for computation in algebraic analysis,* 1991—, See `http://www.math.kobe-u.ac.jp/KAN/`

(たかやま · のぶき, 神戸大学理学部)