数学入門公開講座

昭和62年8月11日（火）から8月20日（木）まで

日 / 時間	8月 11日（火）	12日（水）	13日（木）	14日（金）	15日（土）	16日（日）	17日（月）	18日（火）	19日（水）	20日（木）
13:15～15:00	齋藤	齋藤	笠原	齋藤	休講		一松	一松	一松	一松
15:00～15:15	休憩						休憩			
15:15～17:00	笠原	笠原	笠原	齋藤			南	南	南	南

主催　京都大学数理解析研究所

講師及び内容

1. 曲面の位相幾何 (7時間)

京都大学数理解析研究所助教授 齋藤恭司

曲面には球面・トーラスやクラインのつぼ等々いろいろあるが、その上にどの様な多角形を敷きつめられるか、という問題をオイラー数等の位相幾何を用いて考えてみる。

2. 微分できない連続関数のお話し (7時間)

京都大学教養部教授 笠原晧司

至るところ微分できない連続関数についての歴史を述べ、今までに考えられた数多くの例について、そのグラフを描き性質を調べる。また、このような関数がどれだけ多数あるかについても考察する。

3. 計算機による数式処理 (7時間)

京都大学数理解析研究所教授 一松信

計算機に数式の計算をさせることは、近年漸く実用になってきた。数値計算と比べて、難しかった原因を中心に、近年開発された因数分解算法などを解説する。

4. 対称性(及び反対称性)は自然界にどのように遍在するか (7時間)

京都大学数理解析研究所助手 南政次

自然界に存在する対称性を概観したあと、特に左右対称の問題に戻り、その意味及び反対称性の働きをロジェ・カイヨワに従って論じ、スピノール表示による電子・陽電子の関係などで例証する。次にアイソスピン空間とリー代数の関係、更に一般にリー代数が素粒子の対称性を如何に支配するかを述べる。時間があれば、その対称性の破れと宇宙初期の問題等に触れたい。

時間割

日 / 時間	8月 11日(火)	12日(水)	13日(木)	14日(金)	15日(土)	16日(日)	17日(月)	18日(火)	19日(水)	20日(木)
13:15～15:00	齋藤	齋藤	笠原	齋藤	休講		一松	一松	一松	一松
15:00～15:15	休憩						休憩			
15:15～17:00	笠原	笠原	笠原	齋藤			南	南	南	南

3. 計算機による数式処理 (7時間)

京都大学数理解析研究所教授 一松 信

1987, AUGUST 17, 18, 19, 20 13:15 - 15:00

昭和62年度(第10回)数学入門公開講座テキスト(京都大学数理解析研究所, 昭和62年8月17日～8月20日開催)

計算機による数式処理

数理解析研究所／一松　信

0．初めに

数式処理とは、一口にいえば計算機に自動的に数式の計算をさせることである。英語では以前には Formula manipulation とか Symbolic manipulationといっていたが、近年ではComputer algebra（計算機代数）という語が使われるようになってきた。　これは元来は「計算機による記号代数」の意味だったが、最近では「計算機による代数計算用の算法の研究」というニュアンスで使われるようになってきた。　　その具体的な意味は順次述べる。

式の計算は同じく「計算」といっても、数値計算とは違った性格が強い。　例えば答の表し方が一通りでなく、目的に応じて色々な形が必要である。　その他の理由もあり、数式処理ではまだ当分は、使用者が明確な目的意識をもって使いこなす態度が必要であり、安易にデータを入れれば自動的に答がでると期待してはいけないようである。

数式処理はようやく近年実用の域に達し、マイコンでもかなりの仕事ができるようになった。　既に数学研究の強力な手段の一つとなっている。　数学教育に対する影響も、真剣に考えなければいけない段階になっている。　しかしまだ使用できる計算機が限られていることや、特に国産のシステムがほとんどないことなど、制約が多い。

今回の公開講義で、私がこのテーマを選んだのも、その↓　（次へ）

現状を話して、特に先生方に一人でも多く関心をもってほしいと思ったからである。

I. 数式処理の展望

1. 数式処理小史

大きな流れとして、天文学などの実用をねらったものと人工知能研究から生じたものとがある。

1953 多項式の微分など最初の試み

1960頃 天体軌道への応用の試み

1961 (改良版1966) MIT で不定積分システム SAINT, SIN

1967 Berlekamp による因数分解算法

1969 不定積分に関する Rischの算法

1971 第 1回国際会議

1970頃 日本で微分方程式に対する渡辺隼郎の先駆的な研究

1976 第 2回国際会議 (筆者も出席; 高根の花という印象)

1978 マイコン用システム mu-Math

1980頃 日本でも REDUCE が普及 (本研究所も導入)

1980 ICME-4で VAXYMA のデモンストレーション講演

1983 MIT が MACSYMAを手離し、Symbolic社が売出す。 SMP も商品化

1985 REDUCE 3.0～3.2;マイコン版も順次現れる

上記は私の周辺を主とした年表であり、完全なものではないが、大体の様子が見られるだろう。

2．数式処理の現状

しかしながら、現在の数式処理システムは、まだ誰でも簡単に使えるとはいいにくい。　例えば有名な MACSYMA, REDUCE といったシステムを利用するには、本式の計算機とかなりの(数百メガバイトの)記憶を要する。　マイコンでも使える MU-MATH は、やはり性能が充分ではない。　その上これらの世界的に広く利用されているシステムでさえ、色々と問題点が多すぎる。―しかし我々は、勿論メーカーに注文も出すが、計算機そのものがまだ発展途上にあると割り切り、何ができないかよりも、今すぐに何ができるかを考える方が実りが多いであろう。

現在使いにくいといわれる理由の内、使える場所の制約、説明書の不備、実習の機会の不足、などは次第に改善されるであろう。　それよりも本質的なのは、数式の表現上の約束事や、式の計算では使用者が絶えず自分の計算の経過を意識しながら計算機に適切な指示を与えなければならないといった慣れの必要性であろう。

現在の数式処理システムは、まだ不満が多いとはいうものの、かなり多くの機能をもっていて、到底そのすべてを一度に使いこなすことが出来るものではない。

私自身がよく使うのは、式よりもむしろ厳密な計算―長い整数を末位まで求める計算や有理数としての計算―の機能である。

数学教育でこれをどのように扱うかは、いまから考えて置くべき大問題である。ＩＣＭＥ５でも大きな課題だった。―一部の人が主張するような Black Box 的な扱いには反対が多い。　少なくともそこで使われている算法自体を積極的に教えるべきであり、もしもそれが人間にとっても便利な方法なら、使うべきであるという意見が強い。―但しこれまでのような計算技巧の練習に時間を費やすのはやめて、実際の計算は機械にやらせるべきであろう。

ここでもそのような状況を踏まえて、特に多項式の因数分解の算法について述べる。　その技法は少なくとも先生方が知っていて損はしないと信ずるからである。　但し以下に述べる方法でどんな多項式も必ず完全に因数分解できるとは限らない。　ある程度機械的に出来るが、やはり使いこなすのに練習がいる。　じつのところ、現在の数式処理は、不定積分でも微分方程式でも、名人が使えば95%はうまく解けるが、なかなか100 %とゆかない所が悩みの種である。

3. 数式処理の困難性

同じく「計算」なのに、何故計算機にとって数式の計算が、数値計算よりも桁違いに難しく、実用になったのが最近なのか？　その理由は色々ある。

(1) 結果及び経過が前もって予測し難い。　予め完全なプログラムを書くことが困難であり、一歩一歩経過を見ながらきめ細かく進める必要がある。　そのためには、完全な会話型のシステムやマイコンの発展が不可欠だった。

(2) 計算の中間膨張、及び大量の記憶が必要なこと。　実例は後に示すが、数式処理に必要な記憶量は想像以上である。　現在既に10メガバイトでも不足する例は珍しくないし、どれだけあれば充分だという限界はないらしい（無限大！）。だからなるべく記憶を必要としない計算法の工夫が不可欠である。

(3) 式の表現法に関連する問題　例えば1変数の多項式は、その係数の配列として表現できるし、その形で比較的簡単に取扱える。　しかしこのままでは2変数以上の多項式は扱えない。

その辺の事情を端的に物語る一例は、17世紀の日本において、和算の誕生を巡る田中由真の遺題を関考和が鮮やかに解決した話である。

(4) 反復計算を避ける工夫　いわゆる帰納的(recursive)な算法は、プログラムを書く立場では便利だが、下手に実行すると非能率的なことが多い。　その原因も色々あるが、同じ計算を何回となく反復する無駄が多い場合がよくある。前に計算した結果をとっておいて後に参照することは、人間にとっては自然だが、これを計算機にうまくやらせるのは、記憶が充分にあっても意外と難しい。

(5) 公式データベースの困難性　データベースは、いわゆるエキスパートシス

テムの基本だが、予想外の困難が多い。　例えば言語で表現困難な情報（物の形など）に基く検索法は、必要性が叫ばれて久しいが、うまい方法がほとんどない。

数式処理でも、うまい一般的な算法がない級数の和や特殊関数については、公式データベースが不可欠である。　しかし公式を集めるのは簡単だが、それをどのように整理し、どのように検索するかといった基本問題に対して、これまでの数学も計算機科学も何等の答を与えてくれない。　自分が使う場合を思い起しつつ、泥臭く実用に徹するしかない。

(6) 入出力方式　現在のところ、数式の入力は鍵盤から適当な規約に従って入れるしかない。　手書きの数式を認識してくれると便利だが、それは現在のところ重要な研究課題ではあっても、早急な実用化は期待しない方がよい。

出力はかなり以前から、普通の数式に近い形でできるようになっている。　但しそのためにスペースを多くとる悩みがある。

数式の計算ではなく、数式を奇麗に印刷してくれるシステムも、広義の「数式処理」の話題である。　既に数多くのシステムが市販されているが、概して奇麗なものは使い難く、使い易いものはそれほど奇麗でない悩みがある。

(7) 以上はどちらかというと、システム作成者側の問題点だが、使用者側に一言注意する。　初心者がよくやる失敗は、計算機の能力を過大評価して不必要に問題を一般化しすぎることという指摘がある。　或いは問題と計算機の能力とのバランスを考えよということかもしれない。

とにかく計算機による数式処理を早くマスターする良い方法は、自分が是非とも解きたい問題を持ち、まず手で計算した結果を検算することから始めることである。

II. 多項式の因数分解

1. 多項式の扱い

多項式 $A(x)=a_n x^n+a_{n-1} x^{n-1}+\ldots+a_1 x+a_0$ だけを扱うのならば、これを配列

$$x;n;\ a_n, a_{n-1}, \ldots, a_1, a_0$$

で表現することができる。 変数xを決めておくのならば、冒頭のxを略してよい。 それらに対して、加法・減法・乗法・除法（余りも求める）・微分などの計算を施すのは容易である。

一微分といっても、極限操作は不要である。 形式的に x^n の微分を nx^{n-1} と定義して取扱う。

実際この種の処理は、計算機の歴史でいえば極く初期の1953年に既に成功している。 現在でもマイコン上で、互除法による最大公約式を計算するプログラムを作ってみることは、第一歩のよい演習問題である。

しかしその場合に、係数をいかなる範囲で扱うかが、本質的な問題になる。

(1) 通常の浮動小数点数（実数）として扱う。 簡便だが、割り切れたかどうかを判定するときに、係数が真の0かそれとも0に近いが真の0ではない数なのか、微妙な場合が生ずる。

(2) 整数係数に限り、有理数として扱う。 数学的には一番よいが、特別なプログラムが必要な上に、たいてい能率が悪い。

(3) 適当に定数を掛けて、すべて整数として扱う。 一見簡単だが、下手をするととてつもなく大きな整数が現れる。－Knuth の例。

(4) 整数係数に限り、適当な素数を法とする計算を行う。 かなりの予備知識を要するが、理論的には最も便利である。

まずこの(4) を説明するために、有限体の準備から始める。

2．pを法とする体系

ここでいう「法」とは、除数の意味の古い言葉である。　pを法とするとは、すべての数をpで割った剰余のみを考えることである。　$a \equiv b \mod p$ とは、a-bがpで割り切れることを表す。

pを正の整数とすれば、これは数として有限個の 0,1,...,p-1 しか考えないことである。　0から1ずつ次第に増えていっても、p-1+1 は0に戻ってしまう。――時計はp=12のとき。

pがどのような正の整数であっても、mod p の体系で、加法・減法・乗法が可能である。　普通に計算して、pによる余りをとればよい。

ところが除法（乗法の逆で、余りのない）を実行しようとすると、任意のpでは困る。　例えばp=12のときには $3\times4=0$ となり、一般に3や4で割ることができない。　しかしpが素数ならば、0以外の数で除法ができる。

実際、pが素数のときには mod pの体系　―普通 Z_p と書く―　は、有限体である。　次に Z_p での除法の算法を述べる。

3．Z_p での除法

定理　1（互除法）　2個の正の整数 m,nに対してまず $a_0=m, a_1=n$ とおき、以下

$$a_{i-1}=q_i \times a_i + a_{i+1} \quad , a_{i+1} < a_i \quad , i \geq 1$$

のように除法を繰り返すと、有限回で余り a_{l+1} が0になる。　そのとき最後の除数 a_l が、最初のm,n の最大公約数である。

系　上の互除法において、まず

$$u_0=1,\ u_1=0,\quad v_0=0,\ v_1=1$$

とおき、以下毎回の商 q_i を使って

$$u_{i+1}=u_{i-1}-u_i\times q_i,\quad v_{i+1}=v_{i-1}-v_i\times q_i$$

とおくと、$u=u_l$,$v=v_l$ は $um+vn=d$ を満たす。($u_i m+v_i n=a_i$ に注意)

特にpが素数、$0<b<p$ ならば、 bと pとは互いに素だから最大公約数は1である。 $m=p,n=b$ として系の操作を行うと、$v=v_l$ は $vb=1 \bmod p$ を満たす。 すなわち Z_p での bの逆数である。 a/b を求めるには、a にb の逆数v を掛ければよい。

例1. p=37でb=8 の逆数を求める。 手計算では、次のようにするとよい。

被除数	除数	余り	商	v_i	
37	8	5	4	−4	(0−1×4)
8	5	3	1	5	(1−(−4)×1)
5	3	2	1	−9	(−4−5×1)
3	2	1	1	14	(5−(−9)×1)
2	1	0	2	−37	(−9−14×2)

$v=14$

(余りが1になった段階でやめてよい)

(最後は必ず±pになる。)

――互除法は m,nの小さい方を十進数で書いたときの桁数の5倍以下の回数で完了する(Lamé の定理)。 bの逆数を求めるには、他にもFermatの定理を利用する方法などがあるが、多くの場合上の方法が最も効率的である。―

上記の結果は、多項式の互除法にも、若干修正してそのまま使える。 Z_p での演算については、例えば平方根などについて多くの話題がある。 また複数のpに関する結果から元の数を求めるための「孫氏剰余定理」など、多くの有用な定理があるが、必要の都度「現地調達」で論ずることにする。

4. 多項式の無平方分解

多項式をいつでも完全に因数分解することは、意外に難しい。しかし以下に述べる無平方分解は機械的に可能であり、これだけでも役に立つことが多い。

多項式 $A(x)$ を因数分解した場合に、同一の因子の2乗を含まないとき、すなわち $A(x)=0$ が重複根を持たないとき、$A(x)$を無平方(square free) という。

定理 2　多項式$A(x)$が無平方であるための必要十分条件は、$A(x)$とその導関数 $A'(x)$ とが互いに素なことである。

多項式 $A(x)$ を、無平方な多項式 $P_1(x),P_2(x),\ldots,P_\ell(x)$ によって

$$A(x)=P_1(x)[P_2(x)]^2\ldots[P_\ell(x)]^\ell$$

と表現することを、$A(x)$の無平方分解という。

理論上では、$A(x)$を完全に因数分解した上で、1乗の項、2乗の項，．．．をまとめれば、無平方分解ができるから、これは無意味な概念ではない。しかし実用上では以下に述べる通り、無平方分解は直接に次の算法で、機械的に求めることができるところに重要性がある。

2つの多項式の除法、並びに互除法による最大公約式のプログラムがあるとする。

$$A(x)=a_n x^n+a_{n-1}x^{n-1}+\ldots+a_1 x+a_0$$

に対してその導関数 $A'(x)=na_n x^{n-1}+(n-1)a_{n-1}x^{n-2}+\ldots+2a_2 x+a_1$ との互除法により、両者の最大公約式 $D_1(x)$を求める。

$$E_1(x)=A(x)/D_1(x)$$

を求め、$E_1(x)$ と $D_1(x)$との最大公約式 $F_1(x)$を計算して

$$P_1(x)=E_1(x)/F_1(x)$$

を作ると、これが第1 の因子である。

次に $D_1(x)$ を最初の $A(x)$ と思って、同様に

$$D,E,F,P \text{ を } D_2,E_2,F_2,P_2$$

として求めると、$P_2(x)$ が第2因子である。　この操作を、D_i が定数になるまで反復すればよい。

証明は、A を上記のように無平方分解したとき

$$D_1(x)=P_2(x)[P_3(x)]^2\cdots[P_\ell(x)]^{\ell-1}$$

に注意する。

有理関数の不定積分において、その有理部分は分母の無平方分解のみで計算できる。　異論もあるが、これをHermite の算法ということが多い。　19世紀から既知であり、Goursat の解析教程にも載っていたが、注目をひくようになったのはつい最近である。

Hermite の算法の概要：　不定積分を求める有理関数 $B(x)/A(x)$ において、分子の次数が分母の次数より小さいとしてよい。　分母を無平方分解する。　その最大の累乗指数 ℓ が2以上ならば、以下の操作で $\ell-1$ に下げる。

$Q(x)=P_1(x)\cdots[P_{\ell-1}(x)]^{\ell-1}$ と置くと、$A(x)=Q(x)[P_\ell(x)]^\ell$ であり

$$U_1(x)Q(x)+V_1(x)[P_\ell(x)]^\ell=1$$

を満たす U_1,V_1 によって、$B/A = BV_1/Q + BU_1/P_\ell^\ell$ と分解できる。　P_ℓ と P_ℓ' とは互いに素だから

$$U_2P_\ell+V_2P_\ell'=1$$

を満たす U_2,V_2 によって、後の項 BU_1/P_ℓ^ℓ は、$BU_1U_2/P_\ell^{\ell-1}+WP_\ell'/P_\ell^\ell$；$W=BU_1V_2$ と書くことができる。　この最後の項は部分積分により

$$\int\frac{WP_\ell'}{P_\ell^\ell}dx=-\frac{W}{(\ell-1)P_\ell^{\ell-1}}+\int\frac{W'}{(\ell-1)P_\ell^{\ell-1}}dx$$

と積分できる。　これによりすべての項の分母の最大累乗指数が、$\ell-1$ 以下になった。

5. 多項式の因数分解

多項式の因数分解は、代数方程式を解くためにも必要な技法だが、とかく難問題のための問題が多く、受験数学の癌(?) であった。

それが困難だった理由は、従来の方法が公式あてはめと、有限個の可能性の中から正しい答を探す手法しかなかったことである。 整数係数の多項式の因数分解は、それが可能ならば確かに有限個の可能性のなかに含まれる。 しかしそれだけで実用になるのは、次数が低い（中学校で習う２次式など）ときや、極めて特別な形をしたものに限る。 このような操作を機械化した試みもあったが、それは計算機に人間の真似をさせてみたというだけの意味しかなかった。

既に19世紀に、Eisensteinや Kroneckerが整数係数の多項式を因数分解する算法を研究していたが、彼らの方法は効率的でなかった。 今世紀の初めに、Henselがｐ進数を利用して以下に述べる方法を発見していたが、それが実用にされだしたのは、1970年代以降である。 その間に似たような方法が何度も再発見を繰返している。

現在ではこれから述べるBerlekamp の算法とHenselの補題とによって、大抵の場合手計算よりも速く因数分解ができるようになったが、まだデータをいれればいつでも自動的に正しい答がでるという訳には行かない。 それらを使いこなす努力が必要であり、時にはマイコンに自家製のプログラムを入れて、考えながら使う方がよいこともある。 そのような意味で現在の段階では、まず数学の先生方や数学を好きな人々に、そのような算法を知って頂くのが先決問題のように思う。 （だからこそこういう講義を思いたった次第である。）

多変数の場合にも、一つの変数に注目して類似の方法が適用できるが、多変数の多項式は必ずしも因数分解できないことが多いのに注意する。 例えば２変数の２次式

$$ax^2+2hxy+by^2+2gx+2fy+c$$

が2 個の1 次式の積に因数分解できるための必要十分条件は

$$abc+2fgh-af^2-bg^2-ch^2=0$$

である。

6. Berlekamp の算法

適当な素数 p を法として還元すれば、係数が小さくなって因子を探しやすくなるが、与えられた多項式

$$A(x)=a_n x^n+a_{n-1}x^{n-1}+\cdots+a_1 x+a_0$$

を mod pで因数分解する、以下のような効率的算法を Berlekampが示した。 これは計算機向きの方法であり、必ずしも手計算では便利でないが、マイコンを補助に使って簡単なプログラムで実行できる。

(1) k=0,1,...,n-1 について x^{pk} を A(x) で割った商

$$q^{(k)}_{n-1}x^{n-1}+\cdots+q^{(k)}_1 x+q^{(k)}_0$$

を求める。 実際には割算を実行せずに、x^n を mod A(x) で還元する式を繰返し適用するのがよい。

(2) これらを転置して並べた行列 Q

$$Q=\begin{bmatrix} q^{(n-1)}_{n-1}\cdots q^{(0)}_{n-1} \\ \cdots\cdots \\ q^{(n-1)}_0\cdots q^{(0)}_0 \end{bmatrix}$$

について、固有値 1に対する独立な固有ベクトル $v_1,\ldots,v_r$ をすべて求める。その個数 rが、A(x)を mod pで因数分解したときの因子の個数に等しい。

注意すべきは、$q^{(0)}_i=0$ $(i\geq 1)$; $q^{(0)}_0=1$ なので、$v_1=[0,\ldots,0,1]$ が常に自明な解であり、これを除いて計算する必要があることである。 この計算には、Q-Iに普通のGauss 消去法を、mod p で適用すればよい。 但し枢軸が真に0 になったときにのみ、部分枢軸選びを行う。 その列がすべて0 になって枢軸選びができなくなったら、一旦消去計算を打切り、逆行計算を行って一つの v_i を求める。 次には既に求めた部分を消して、それ以後の部分のみを計算する。

なお行列式 $\det[Q-\lambda I]$ は符号を除いて $\prod_{i=1}^{r}(\lambda^{e_i}-1)$ の形に因数分解できる(e_iに同じ値があってもよい)、そのときA(x)は $e_1,\ldots,e_r$ 次の多項式の積に因数分解できる。

(3) 自明でない固有ベクトル $v_i=[u_{n-1},\ldots,u_0]$ に対して

$$U(x)=u_{n-1}x^{n-1}+\ldots+u_1x+u_0$$

と置く。 適当な定数 cに対して、U(x)-c とA(x)との最大公約式は定数でなく、それがA(x)の一つの因子である。 A(x)の因子は、このようにしてすべて求めることができる。 但し同一の因子が重複して現れうるので、予め無平方分解してから実行した方がよい。

p が小さいときや、U(x)が一次式のときには、c=0,1,...,p-1 を順次試してみるのがよい。 逆にr=2 で、U(x)がn-1 次式というときには、c をパラメタとして割ったときの余りの係数が、c について1 次または2 次式になるので、幾つかの定数c に対する剰余の式からその係数の式を求めると、それらが一斉に0 になるようにc の値を決めることができる。

このようにして与えられた多項式を mod pで因数分解することができるが、mod p で代数方程式 A(x)=0 を解くことは一般に難しい。 1次、2次の因子は容易に解けるし、3次・4次の因子も代数的に解けないことはないが、5次以上の因子には一般的な代数的解法がない。 pが大きいと、有限個の全数を試してみることは、実行不可能である。

有限体 Z_p 上の代数方程式が解きにくいのは、Z_p が離散的な対象であって、その上に位相(topology)がないために、逐次近似の手法が使えないためである。

p進数の応用は、Z_p 上に一種の位相を導入する工夫と解釈できる。

例2 $A(x)=x^4-3x^2+1$. $p=11$.

$q_i^{(q)}$ を計算すると $Q=I$ となり, $r=4$ となる。 c を順次代入して $A(c)=0$ になる c を探し

$$A(x)=(x-3)(x-4)(x-7)(x-8)=\Pi(x\pm3)\Pi(x\pm4)$$

を得る。

付記: Berlekampの算法は, A(x) が既約なことを確認するのに有用なことが多い.(但し既約なのに各pで可約なA(x)もある。)

7 Henselの補題

定理3 (Hensel の補助定理) mod p で多項式A(x)が、互いに素な2 個の多項式 $G_1(x), H_1(x)$の積に分解されたとする。 このとき各正の整数 mに対して

mod p^m で $A(x)=G_m(x)H_m(x)$; mod p で $G_m(x)=G_1(x), H_m(x)=H_1(x)$

を満たす多項式 $G_m(x), H_m(x)$が存在する (一意的ではない)。

証明 mの帰納法。 mのときできたとすると、条件は

$$A-G_mH_m=p^mB$$

である。 G_m, H_m は互いに素にとれるので

$$UG_m+VH_m=B$$

を満たすU,V がある。 それから

$$G_{m+1}=G_m+p^mV,\quad H_{m+1}=H_m+p^mU$$

と置けばよい。

これは純粋に代数学の定理である。 もしも初めの分解が正しい因数分解をmod pで還元したものになっていたならば、mを充分大きくするとこれによって正しい因数分解ができることが多い。 但し出発点が狂っていると、無駄な努力に終る。

例3 $A(x)=x^4-3x^2+1$, p=11で $G_1(x)=x^2-5$, $H_1(x)=x^2+2$ とすると

$$B(x)=1;\ U=3, V=-3;\ G_2=x^2-38,\ H_2=x^2+35$$

となる。 確かに$G_2H_2=A-11^3$ であるが、この操作をいくら反復しても正しい因子はえられない。

これは出発点が誤っているためである。 4 個の因子の組変えを行い

$$G_1(x)=(x-3)(x+4)=x^2+x-1,\quad H_1(x)=(x-4)(x+3)=x^2-x-1$$

とすれば、正しい因子がでている。

実用上ではさらに、最高次の係数が1でないとき、その分配に注意を要する。

例4: $54x^3+57x^2+28x+15$; $p=11$ ととると

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

$G_1(x) = -x+1$, $H_1(x) = x^2 - x + 4$ とおくと

$$A(x) - G_1(x)\cdot H_1(x) = 11(5x^3+2x^2+3x+1) = 11B(x)$$

$$B(x) = (5x+7)G_1(x) + (5x+4)H_1(x) \pmod{11}$$

$$G_2(x) = 54x+45 = 9(6x+5),$$

$$H_2(x) = x^2+54x+81 = x^2+27(2x+3).$$

$9\times 27 = 243 \equiv 1 \pmod{11^2}$ に注意すると，9を G_2 から H_2 に移して，$\bmod 11^2$ で $6x+5$，$9x^2+2x+3$ となる．これは正しい因子である．

III　多変数の多項式

1．多変数多項式の扱い

2変数以上の多項式については、まずその表現が問題になる。　最も完全にするには、適当な順序に揃えた上で、各項を例えば

$3x^3y^2z$　ならば　(3; x,3; y,2; z,1)

のように表すのだが、実際には都合のよいように色々と便法を使う。

理論的にはどうでもよいのに近いが、実用上では項の順序が重要である。　多くの場合、その順序は項の指数に応じた全順序であって、次の性質を満たすものならば何でもよい。　(順序を▷で表す。)

(1) A▷B, C▷D　ならば AC▷BD　(2) 順序▷に関する無限降下列がない。

この目的のために普通によく使うのは、変数の文字に適当な順序（例えばアルファベット順）を付けた「辞書式順序」である。　しかしこれを機械的に適用すると、しばしば甚だしく非能率になる。　プログラムが複雑になるが、まず全次数の大きい方から順序を付け、同じ次数の項は辞書式順序にする方が優れている。

2. 多変数多項式の因数分解

理論的には、いくつかの変数のうち一つxを主変数とし、他を副変数（パラメタ；まとめてy）とみなして、yの多項式を係数にもつxの多項式と考えれば、同様の手法が適用できる。

このような2個の多項式 $A(y;x),B(y;x)$ に対して、A-B のすべての係数が yについてm 次以上の多項式(0は無限大次とみなす) になるとき、 $A\equiv B \mod y^m$ と表すことにする。

このとき次のようなHenselの補題の類似が成立する。 証明も同様にできる。

定理4 $A(y;x)$ に対して、x の最高次の係数が $y=0$ のとき0 でないと仮定する。ーこの条件は理論的には、それが恒等的に0 でなければ、座標変換によっていつでも満たすようにできる。ー $y=0$のとき $A(0;x)=G_1(x)\,H_1(x)$ と互いに素な因数に分解できたとする。 このとき各正の整数m に対して、

$$A(y;x)=G_m(y;x)H_m(y;x) \mod y^m;\ G_m(0;x)=G_1(x),H_m(0;x)=H_1(x)$$

を満たす $G_m(y;x),H_m(y;x)$ が存在する。

例5 $A(y;x)=x^4-2x^3y-x^2y^2-2xy^3+y^4+4x^2y+4xy^2-3x^2-3y^2+1$.

これはx,y の対称式だが、今はそれを無視して、x の4次式とする。

$$A(0;x)=x^4-3x^2+1=G\,H\ ;\ G_1=x^2+x-1,\ H_1=x^2-x-1$$

$$A-G_1H_1=yB_1,\ B_1=(4x^2-2x^3) \mod y;\ B=(x-1)G_1-(3x-1)H_1$$

$$G_2=x^2+x-1-(3x-1)y,\ H_2=x^2-x-1+(x-1)y$$

$$A-G_2H_2=yB_2,\ B_2=2x^2-2-2xy \pmod y =G_2+H_2$$

$$G_3=G_2+y^2=x^2-3xy+y^2+x+y-1,\ H_3=H_2+y^2=x^2+xy+y^2-x-y-1$$

最後の G_3,H_3 は正しい因子である。 H_3は既約だが、G_3は無理数を使えばさらに

$$(\tau x-\tau^{-1}y-1)(\tau^{-1}x-\tau y+1),\ \tau=(\sqrt{5}-1)/2$$

と因数分解できる。

この場合には対称式の積に因数分解できるので、$s=x+y,t=xy$の多項式に直して計算すると、いくらか早くできる。

すなわちこの式は

$$s^4-6s^2t+5t^2+4st-3s^2+6t+1$$

と書くことができる。　これは例えば tの 2次式として容易に

$$(t-s^2-s+1)(5t-s^2+s+1)$$

と因数分解できる。　これを元の x,yの多項式に戻すと、前の結果の符号を変えた因子になる。

対称式に関して、例えば累乗和 $x_1^\ell+\ldots+x_n^\ell$を基本対称式で表す公式がある。ーNewtonの公式とよばれている。ー　その種の扱いも、近年計算機に組込まれてきた。

3．Gröbner 基底

多(n) 変数の連立代数方程式 $A_1=0,\ldots,A_m=0$ は、 n変数 $x_1,\ldots,x_n$の多項式環の中の「イデアル」とみなすことができる。　1変数のときと違って、その基底には任意性があって、標準的なものが作りにくい。　それが永年悩みの種だったが、近年一つの案が現れた。

Gröbner 基底の概念は、B.Buchbergerが1970年に導入したものだが、最初の(その次のも) 論文は記法・証明ともわかりにくく、余り注目をひかなかった。わかりやすい解説論文が現れ、有用な手法として普及してきたのは、1980年以降である。　それは一つには計算機上に実現するのに時間が掛ったせいであるが、不定積分のRisch の算法の場合と同様に、数学の新しい概念・算法の普及が、物理学の場合とは違って意外に時間が掛る例のようでもある。

Gröbner 基底の理論については、日本ではまだまとまった成書がなく、数式処理関係の報告集に解説記事がある程度である。　ここでは例を示すのに留める。

これを活用すると、連立代数方程式のすべての根を求めることや、線型不定方程式の解を求めることができる。　しかし実用上では、Gröbner 基底を完全に求めようとすると、莫大な記憶を必要とするので、一般論によるのでなく色々と簡便法が利用される。

多項式の集合 $F = \{P_1, \ldots, P_m\}$ に対して、別の多項式A が<u>M-可約</u>とは、A の項の中に、P_i のどれか少なくとも一つの「最大」指数の項よりも、すべての変数について指数がそれ以上のものがあることである。 例えば

例6 $P_1 = xy - 1$, $P_2 = x^2 - y$ に対して

$A = x^2 y$ はM-可約であり、$B = y^2 + x$ はそうではない。

A がF についてM-可約ならば、適当な P_i と単項式C とをとって

$$A_1 = A - C\,P_i \text{ がA より順序が下}$$

であるようにできる。 これを P_i による<u>還元</u>という。 この操作を反復して、有限回で（これは定めた順序に関する無限降下列がないことによる）、F についてM-可約でない多項式にまで還元することができる。

$F = \{P_1, \ldots, P_m\}$ が<u>Gröbner基底</u>とは、F から生成されるイデアルJ の 0でない要素A が常にF についてM-可約なときである。 そのようなF に対して、J の要素A は常に0 に還元できる。 またA-B がJ の要素ならば、適当に共通のC をとって、A もB もC に還元できる。

二つの多項式A,B に対して、両者の最小公倍式をL とし 最大指数の項 $\tilde{A}, \tilde{B}$ の

$$P = L/\tilde{A},\quad Q = L/\tilde{B}$$

と置く。 定数c をA の最大指数の項の係数を、B の最大指数の項の係数で割った商とし

$$Sp(A,B) = PA - cQB$$

をA,B から作った<u>S-多項式</u>という。

<u>Gröbner 基底の基本定理</u> 1°. $F = \{P_1, \ldots, P_m\}$ がGröbner 基底ならば、任意のi, j について $Sp(P_i, P_j)$ は0 に還元できる。

2°. FのGröbner 基底を求めるには、その中の任意の二つA,B を取出して Sp(A, B)を還元し、結果が0 でなければそれを追加するという操作を反復すればよい。
——有限回で終ってGröbner 基底をうる。

これがBuchbergerの基本定理だが、証明は易しくない。 現在では2°.のみの直接証明もできているが、それだけでもかなりの時間を要するので、今回は省略せざるをえない。

4. 連立代数方程式

連立代数方程式を一般的に解くことは容易でない。　理論的には順次変数を消去して、1変数の代数方程式にまで還元すればよい。　それもある程度機械的にできる。　しかし実際には計算機によっても必ずしも簡単ではなく、対称性などをうまく利用して解く工夫が有用だった。

例7　3次のパーマネントの極値を求める方程式：

$$2x^2-x+4x(u+y-1)+1-2u-2y+4(xv+uy)=0$$
$$2v^2-v+4v(u+y-1)+1-2u-2y+4(xv+uy)=0$$
$$2y^2-y+4y(x+v-1)+1-2x-2v+4(xv+uy)=0$$
$$2u^2-u+4u(x+v-1)+1-2x-2v+4(xv+uy)=0$$

差をとれば

$$x=v \quad \text{または} \quad 2(x+v)+4(u+y)=5$$
$$y=u \quad \text{または} \quad 4(x+v)+2(u+y)=5$$

をえ、それらの組合せ4種からそれぞれ4通り、合計16通りの解をうる。　それらは元に返せば、本質的に次の3個の行列の成分を入換たものに当る。

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$
これが真の最小

$$\begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \\ \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \\ \frac{2}{3} & \frac{2}{3} & -\frac{1}{3} \end{bmatrix}$$
入れ換え4通り

$$\begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$
入れ換え6通り

ところでもしも連立代数方程式 $P_1=0,\ldots,P_m=0$ について、$P_1,\ldots,P_m$ のGröbner 基底Gを求めることができたならば、次のような判定及び解法が成立する。

1°. 上の連立代数方程式に解があるための必要充分条件は、0でない定数がGに含まれないことである。

2°.上の連立代数方程式の解が有限個であるための必要充分条件は、各 i(=1,...,n)に対して、x_iの適当な累乗を最大指数の項とするP_jがG 中に存在することである。

3°.連立代数方程式$P_1=0,\ldots,P_\ell=0$ を解くには、つぎのようにする。

(i) まずGröbner 基底 $Q_1,\ldots,Q_m$ を作る。 但しこのときに各j についてQ_jが、それを除いた残りに対してM-可約でないように還元しておくものとする。

(ii)このように還元すると、上の性質2°.により、G 中にx_nだけの多項式Q が唯一つ存在する。 $Q(x_n)=0$ を解き、その解の各々a_kについて、それを他のQ_jに代入整理する。

(iii) えられたn-1 変数の連立代数方程式について、同じ操作を繰替す。

但しこれは理論上の話である。 実際にこの通りの計算を実行しようとすると、ごく簡単な場合以外には、たちまち大量の複雑な式が現れて、手に負えなくなるのが普通である。 特に順序を機械的に辞書式順序にとったときそうなりやすい。

実際にはGröbner 基底自身を作らずに、全次数の最高の項（同一の次数については辞書式順序による）に着目し、それを他の式について還元する操作を反復することと、因数分解ができる項にはそれを施すことを繰替すのがよいようである。

例えば上記の例 7について、かなり機械的にこの操作を試みたところ、2 変数に還元された段階で、全部で10個近くの因子に分解され、あっさり解けてしまった。

もう少し「実際的」な問題では、桂先生が提出したスピングラスの計算のための 4元連立代数方程式の例がある。

ここで大事なのは、次の点であろう。 連立代数方程式には一応このような一般的な解法があるが、これまでの手計算では極めて簡単な場合以外には手に負えず、やむなく個々の問題に応じた一つ一つの工夫が必要だった。 しかし今や計算機による数式処理により、ある程度まで機械的な計算で一般的に解けるようになった。 もちろん現状では「何でも自動的に解ける」というのは誇大宣伝だが、解ける範囲が広がったこと自身、重要な進歩であろう。

Ⅳ　今後の問題

1．残したテーマ

当初計画したが、時間の関係で省略した関連話題が少なくない。　その項目だけを挙げておく。

(i) Z_p を使うときの素数pの選定。素数pの個性が反映する。　Henselの補題と併せて Berlekampの算法を使うとき、p を大きく取りすぎるのは損だが、2, 3,5,7 などは特殊性がありすぎて、うまく行かないことが多い。　私の経験では 11,13,17,19,...を使うのが無難なようである。

また Z_p に対する2次方程式の解法、特にmod p の平方根の計算法をめぐって、興味ある結果が多い。

(ii)二つの多項式の「部分終結式」と、互除法の途中現れる多項式との関係。また三つ以上の多項式の間の「互除法」と、その応用。一線型不定方程式など。

(iii) 不定積分に関するRisch の算法。歴史的にいうとLiouville が1830年代に行った「不定積分が初等関数で表されるか否か」に関する研究に端を発する。これは150 年前のほとんど忘れられていた研究が、劇的に復活したという、他の自然科学では考えられないような実例でもある。

2．数式処理自体の今後の問題

(i) 代数体上の因数分解。上に述べたのはほとんどが有理数または整数上の因数分解だが、 2次体（複素数体も）や aのn 乗根を添加した体上の因数分解が必要なことが多い。　MACSYMA では x^3-2a^3 程度の式は扱えるが、これらを一般的に組込みたい。

(ii)数値計算との接点。これが実用に当っての最大の問題である。　これは大袈裟にいうと、既にかなりの歴史をもつFORTRAN 文化とLISP文化との「文化摩擦」である。　双方とも相手方から相当の「内政干渉」を受容しない限り、簡単に解決しそうな問題ではない。　当面は中間に処理システムを介するのがよいらしい。

なお導関数を使うには、在来の数式処理によるよりも、関数の「計算グラフ」に基く「高速自動微分法」（他にも多くの名がある）を活用する方がよいらしい。

これも今回述べる暇がなかったテーマだった。

以上駆け足で「計算機代数」の入門（というより私が興味をもっている部分）を解説した。　計算機による数式処理にいくらかでも興味をもって下さるきっかけになったのならば、望外の幸である。

日本語の参考書

[1] 後藤英一他編、数式処理のすすめ　bit 別冊,1986,共立出版

[2] 佐々木建昭、数式処理、情報処理ハンドブック,1981,オーム社

[3] 佐々木建昭、数式処理、岩波講座情報科学 no.23　「数と式と文の処理」,1981;再版1984, 岩波書店

他に専門家向けのニューズレターとして、サイエンティスト社発行の「数式処理通信」（年 4回）がある。　また研究発表では、数理解析研究所の講究録に数冊の報告がある。