解　説

計算機による数式処理の現状*

後藤英一** 佐々木建昭***

1. 序

微積分学を創始したLeibnizは計算機の発明者でもあり，“機械にでもできる計算などという仕事を，貴重な人間の時間を費してこつこつ行うのは全く馬鹿げたことである”と述べている．電子計算機の発達により，近年数値計算を人手でこつこつ行う必要は全くなくなった．しかし数値計算から一歩でると，まだまだLeibnizのいう“馬鹿げたことをこつこつ行う”必要がある．因みに筆者の一人（E.G.）は数年前，研究上の必要から電子幾何光学の3次収差の公式を計算したことがあった．公式の導出原理は簡単かつ機械的なのだが，公式の長さが数頁に及ぶと，誤りがないようにするのに大変な労力を要した．手もとにすぐ使える数式処理システムがあったらよいのにとつくづく感じた．

応用数学上の問題を解くにあたって，基礎方程式を立てるまでが人間的な仕事であって，公式集などを参照しつつ数式を変形処理する過程の大部分は全く機械的な作業にすぎない．このような数式処理の作業を計算機に行わせようという試みは，電子計算機誕生直後の1950年代にもいくつかなされたが，計算機の能力の不足で見るべき成果が得られなかった．

計算機による数式処理で特に著名な事例は，1960年代の初頭にMITで，Slagle[1]が初等関数の記号的積分を行うプログラムSAINTを作成したことである．大学1年生の積分の練習問題の95%は計算機があっという間に答を出すということで大変有名になった．しかし，このような先駆的成果があったにもかかわらず，計算機による数式処理が理工学上の実際の問題に適用され，見るべき成果をあげられるようになってきたのは1970年代に入ってからで，SlagleのSAINTから約10年の歳月を要した．その10年の間に，数式処理用の各種のアルゴリズムの研究と数式処理系（プログラミング・システム）の開発が進展し，また計算機自体も進歩（記憶容量の増大と速度の向上）して，数式処理もやっと使えるようになってきたところである．以下ではまずアルゴリズムの研究を展望し，ついで処理系についてふれ，数式処理に関する国際会議としては最も最近に開催されたSYMSAC '76の内容を概観し，最後に問題点と将来について述べることにしたい．なお，数式処理をテーマにしてBartonとFitchが1972年に書いた総合報告[2]があるが，これは今日でも一読の価値がある．また，数式処理に関する文献はACM(アメリカ計算機学会)のSIGSAM BulletinとSIGSAM関係の国際会議(SYMSAM, EUROSAM '74, SYMSAC '76)の議事録[3)~5)]に詳しい．

2. 数式処理アルゴリズム

過去15年間に最も進歩したアルゴリズムは，初等関数の不定積分法と多項式ならびに有理式（有理係数で多変数でもよい）の演算に関するもの，特に二つの多項式の最大公約多項式(以下GCD: Greatest Common Divisorと略す)を求めるアルゴリズムと因数分解アルゴリズムである．

二つの多項式のGCDを求める操作は，分数式計算において数式の爆発（演算のたびに式が長大になること）を防ぎ，式を簡潔にしてその意味を人間に対し明解にするにとどまらず，数式処理の大部分のアルゴリズムに不可欠の重要な操作である．二つの多項式のGCDを求めるには原理上の問題は何もなく，二つの整数のGCDを求めるEuclidの互除法を素直に多項式の場合に一般化すればよいと思われていた．ところがこれを実際に計算機で実行しようとすると，多項式の次数や変数の個数の増大につれて，途中結果に現れる係数（整数）の桁数が急激に増大し，速度と記憶容量のいずれの観点からも実用にならないことがわかっ

* Present Status of Algebraic Manipulations by Computer by Eiichi GOTO (Department of Information Science, University of Tokyo) and Tateaki SASAKI (Information Science Laboratory, The Institute of Physical and Chemical Research).

** 東京大学理学部情報科学科
*** 理化学研究所情報科学研究室

た．そこで種々のアルゴリズムが提案研究されたが，現在では（素数を法とする）有限体を利用する計算法が使われる．この方法では整数係数多項式の係数を素数 p を法として，0から $p-1$ までの有限の範囲に縮約して Euclid の互除法を適用する．p を法とする計算では，途中結果の係数も常に0と $p-1$ との間に入るので，桁数増加の問題は起こらない．これだけでは結果は p を法としてしか出ないが，$p_1, p_2, p_3, \cdots$ のいくつかの素数に対して同じ方法を適用し，それらの結果から中国剰余定理によって正しい GCD を導くというのが一つの方法である[6]．この方法は多項式が密（零係数項が少ない）の場合には能率がよいとされているが，実際に現れる多項式のほとんどは粗（零係数項が多い）なので，p の次には $p^2, p^3, \cdots$ を法として計算を進める“Hensel の補題”に基づく方法[7),8)]が現在のところ最良とされている．

整数係数の多項式（有理係数多項式の場合は，係数の分母の最小公倍数をくくり出せば，整数係数多項式の場合に帰着する）を整数係数の範囲内で有限回の操作により因数分解する方法は，Kronecker のアルゴリズムの名称で代数学の標準的教科書[9)]に記載されているが，人間の手ではやっかいと思える少し複雑な多項式に対しても，その方法は膨大な数の試行を必要とし全く実用に向かない．GCD と同じく因数分解においても，素数 p を法とする計算法，Hensel の補題を利用する計算法が有力である．因数分解の操作は本質的に試行錯誤であるが，素数 p を法とする有限体上で，試行の回数を極端に少なくする有効なアルゴリズムが Berlekamp[10)]により見い出された．これが基本的な進歩であった．彼はさらに，素数 $p_1, p_2, \cdots$ を法とする有限体上での因数分解から中国剰余定理を使って，整数上での因数分解を導く方法を提案したが[11)]，ここでは，それより一般的にはるかに効率のよい，Hensel の補題に基づく Zassenhaus の方法[12)]を略述しよう．$F(x)=G(x)\cdot H(x)$ を因数分解したい多項式として，$F(x)\equiv G_1(x)\cdot H_1(x) \pmod{p}$ なる互いに素な $G_1(x)$ と $H_1(x)$ が与えられたとしよう．そのような G_1 と H_1 は Berlekamp のアルゴリズムにより容易に求められる．Hensel の補題[9)]によれば，$F(x)\equiv G_2(x)\cdot H_2(x) \pmod{p^2}\equiv G_3(x)\cdot H_3(x) \pmod{p^3}\equiv\cdots$ なる $G_i(x)\equiv G_1(x) \pmod{p}$, $H_i(x)\equiv H_1(x) \pmod{p}$ を順に構成することができる．この拡張を p^i が十分大きくなるまで繰り返せば，G, H と単数（今の例では定数になる）倍の違いしかない G_i, H_i が得られる．最後に単数因子を調節すれば望む G, H が得られる．Zassenhaus はさらに，$p^2, p^{2^2}, p^{2^3}, \cdots$ と順に拡張するアルゴリズムを見いだしている．以上は1変数多項式に関する話であるが，重要なことは，全く同様な論法が多変数多項式の因数分解にも使えることである[13)]．$F(x, y_1, y_2, \cdots, y_n)$ を因数分解したい式とする．$b_1, b_2, \cdots, b_n$ を適当に与えられた整数とし，F を変数 $y_1, y_2, \cdots, y_n$ につきそれぞれ $(y_1-b_1), (y_2-b_2), \cdots, (y_n-b_n)$ のベキ級数に展開し，全ての y_i-b_i の次数の和が m 次以上のあらゆる項を0とおいた多項式を，$S^m=\{y_1-b_1, \cdots, y_n-b_n\}^m$ を法とする F の表現と定義する．今，$F\equiv G_1\cdot H_1 \pmod{p^k, S}$ なる $G_1(x)$ と $H_1(x)$ が与えられたとしよう．$F \pmod{S}$ は変数 x だけの多項式であるから，そのような G_1 と H_1 は容易に与えることができる．Hensel の補題を拡張すれば，$F(x, y_1, \cdots, y_n)\equiv G_2(x, y_1, \cdots, y_n)\cdot H_2(x, y_1\cdots, y_n) \pmod{p^k, S^2}\equiv\cdots$ なる $G_i(x, y_1, \cdots, y_n)\equiv G_1(x) \pmod{p^k, S}$, $H_i(x, y_1, \cdots, y_n)\equiv H_1(x) \pmod{p^k, S}$ を順に構成することができるのである，この方法は EZ アルゴリズムと呼ばれ，非常に有効であることが確かめられている．有理数体上の多項式に限定する限り，1変数であれ多変数であれ，因数分解に実際的困難はないと言える．

SAINT のあとをうけて数多くの記号積分プログラムが作成されたが，そのうち最も成功を収めたものは Moses によるプログラム SIN である[14)]．SIN には数学者が机上計算において用いる数々の発見的手法が組み込まれており，答は被積分関数に近い形で出力される．しかしながら，積分のアルゴリズムにおける画期的な進歩は Risch によってもたらされた．初等関数（実あるいは複素変数の関数で，有理関数，代数関数，指数関数，および対数関数，あるいはこれらを任意に組み合わせて得られる関数の総称．たとえば，$\log\log(e^{x^2}+1)$）の不定積分に関しては，もし不定積分がやはり初等関数であるならば同じ階数の初等関数とその log で表現できる，という有名な Liouville の定理[15)]がある．しかし，任意の初等関数が与えられた場合，その不定積分は初等関数であるのかどうか，またそうである場合にはその不定積分を有限回の手続きで求めるアルゴリズムは存在するのかどうか，存在する場合にはそのアルゴリズムを求めることは19世紀以来の未解決問題であった．これに対して，1967年（不完全ではあるが）解答が与えられた．すなわち，初等関数の部分集合（有理関数と指数ならびに対数関数，およ

びこれらを任意に組み合わせて生成される関数属）に対して，不定積分が初等関数で表わされるかどうかを判定し，表わされる場合にはそれを求めるアルゴリズムが Risch により発見された[16]．Risch のアルゴリズムは理工学で用いられる関数の多くを処理しうる一般的なものだが，被積分関数が exp と log を含む場合には，それらのうちから代数的に独立な要素を選び出す必要があり，さらに不定積分における log の項を求めるためには，多項式を完全に因数分解する必要がある．前者は後に述べる簡単化の問題と関連して，後者は，5次以上の方程式は一般にベキ根では解けないとの有名な定理と関連して，数学的に深刻な問題であるが，Risch のアルゴリズムの有効性を大きく損なうものでは決してない．アルゴリズムの主部は問題を連立一次方程式に帰着させることであり，計算に要する時間は驚くほど短い．

数式処理のアルゴリズムのなかで最も基本となるものは（記憶装置の容量が許す限り任意の桁数まで計算する），任意多倍長整数演算と多項式の四則演算である．任意多倍長整数の乗算を 2^n 進表示（2^n は計算機の1語長）で行うと，N桁の整数を2個乗ずるのに必要な操作数は $O(N^2)$ となるが，有限体 GF(p) 上の高速 Fourier 変換 (FFT) を利用すると，$O(N \log N \log\log N)$ でできることが Knuth の著書[11]に述べられている．次数が $n, m\ (n \geqq m)$ で非零係数の項数がそれぞれ $N, M\ (N \geqq M$ とする）の二つの多項式の乗算についてもよく調べられている．多項式が密（$n \approx N$, $m \approx M$）な場合には，零係数項を補充して降ベキ順に整列しておくと，必要な演算操作数は古典的方法で約 $2nm$，FFT を利用すれば $O(n \log n)$ になる[17]．実際に表れる多項式の大部分は粗（$n \gg N$, $m \gg M$）で降ベキ順に整列されていないことも多い．その場合には指数の同定に多数回の比較を要し，演算操作数は古典的方法で $O(NM^2)$ となるが，整列アルゴリズムを利用すると $O(NM \log N)$ となる[18),19]．注意すべきは，多項式の次数と密さ加減により最も有利な方法が異なり，$n, m \leqq 20$ ではむしろ古典的方法が有利である場合が多い．最近，粗多項式の乗算を $O(NM)$ で実行する方法が発表された（4. を参照）[20),21]．

次に，数式処理上重要な操作である数式の簡単化にふれよう．多項式については因数分解が，有理式については GCD 操作が重要な簡単化であり，それらについては既にふれた．さらに一般的な式について，一見すると簡単化は何でもないようであるが，実はそうでない．多項式の有理数ベキ（たとえば $\sqrt{x}$）に限っても，分枝 (branch) を指定しなければ与えられた式が0かどうか判定できない．これまでによく研究された数式は有理指数表式 (rational exponential expression) である．ここで有理指数表式とは，n個の独立変数と定数 $i=\sqrt{-1}$ および π とから構成され，任意の有理指数表式の加減乗除および指数関数がまた有理指数表式である，と定義される．したがって，三角関数や双曲線関数，指数関数もこれに含まれる．これらの数式に関しては，数学的に証明されてはいないが非常にもっともらしい仮定に基づいて，与えられた指数表式が0に等しいかどうかを判定するアルゴリズムが Brown[22] により考え出された．このアルゴリズムでは与えられた数式を標準形に変換する必要がある．そのようなアルゴリズムの存在は証明されたが[23]，アルゴリズム自体は有理指数表式の部分集合についてしか作られていない[23),24]．簡単化は別の関数集合，たとえば実変数 x に四則演算を含めて，exp, sin, cos それに絶対値の log を任意に組み合わせて生成される関数の集合[25]，についても調べられた．しかし，数多くの簡単化のアルゴリズムが考案される一方で，簡単化には限度があることが Richardson[26] により明らかにされた．

Hilbert の第10問題の非可解性[27),28]に関連して，初等関数のいくつかに絶対値関数を加えた関数集合で，その任意の要素が恒等的に0であるのかどうか決定できない，との定理が証明されたのである[24),29]．数式の簡単化を考えるに当たっては，これらの非決定性の結果を十分心得ておく必要がある．

その他のアルゴリズムでは，無平方 (square-free) 分解などの多項式に関する演算，行列式や一次連立方程式の高速演算，特殊関数（楕円関数，誤差関数など）に関する演算などに加えて，理工学の特定の分野での必要性により開発されたものがある．天体力学における Poisson 級数（$p \sin q$ と $p \cos q$ の有限級数：ここで p は多項式で q は6個以下の変数の一次結合），素粒子物理学におけるγマトリックス（4行4列の行列で16種類ある）の高速演算アルゴリズムは後者の代表的な例であろう．さらに最新のアルゴリズムについては 4. を参照されたい．

3. 数式処理系

現在までに試作された数式処理系は世界中を合わせれば100以上はあると思われるが，国際的に著名なも

の，現用のものとしては，MIT の MACSYMA[30]，Utah 大学の REDUCE[31]，Cambridge 大学の CAMAL[32]，Bell 研究所の ALTRAN[33]，IBM の，FORMAC[34]，SCRATCHPAD[35] などがあげられる．歴史的には FORMAC が最も古く，かつ計算機メーカが商品として作成した唯一のシステムであるが，時代に先行し過ぎたのであろうか，商品としての価値が低いので IBM は FORMAC の維持，改良，運営から一切手を引いてしまった．このため現在では FORMAC は，Darmstadt 大学の Bahr たちなどのユーザの手によって運営されている．SCRATCHPAD は IBM の研究所内で研究用にだけ使用され，ソフトウェア商品として販売する予定は現在のところないとのことである．

MIT の MACSYMA は，Slagle の SAINT 以来の伝統もあり．積分や因数分解などの各種の進んだアルゴリズムが最も多く組み込まれている．MIT にある記憶容量 500 kW の PDP 10 計算機にインプリメントされており，米国全土の利用者は通信回線を介してこのシステムを利用している．現在のところ MACSYMA システムの他の計算機への移設は積極的には行われていない．

REDUCE は Utah 大学の Hearn が，QED（量子電気力学）上の数式計算の必要から出発して開発したシステムである．MACSYMA よりは小型で組み込まれている機能は少ないが，400 kB 程度の記憶容量の計算機でもかかり，また Hearn は他の計算機に移設することに非常な熱意を持っている．

Cambridge 大学の CAMAL は，天体力学上の数式計算の必要から Barton が作成したシステムから出発し，現在は Fitch が中心になって運営している．

Bell 研究所の ALTRAN は FORTRAN に立脚したシステムで，有理式計算に強いとされている．

国内では柱（東北大学）の FORMAS，電電公社通研の AL[36]，渡辺（津田塾大学）による常徴分方程式の記号解法システム[37] などが作成されている．これらのシステムの詳細についてはそれぞれの作成者に記述していただくのが適当であろうし，直接問い合わせることも容易なので，ここでは割愛させていただく．筆者の一人（E. G.）は LISP にハッシュ符号法を利用するデータ構造を追加した“HLISP”を考案し[38,39]，これに前記の REDUCE をインプリメントした，“HLISP-REDUCE”なるシステムを開発した（この実現には，大学院生である金田，寺島両君に負うところが多い[40,41]）．HLISP 特有の機能を使えば，いくつかの数式処理アルゴリズムの高速化が実現できる．

以下に二，三の実例を示そう．Legendre 多項式を公式 $P_n(x)=\partial^n/\partial y^n(y^2-2xy+1)^{-1/2}|_{y=0/n!}$ より求める．これを行う REDUCE のプログラムは

```
ALGEBRAIC PROCEDURE P(N, X);
SUB(Y=0, DF((Y**2-2*X*Y+1)**
    (-1/2), Y, N))
        /(FOR I := 1 : N PRODUCT I);
```

である．この定義のあと 2*P(2, W) を入力すると

$$3*W^2-1$$

と出力される．このように大部分のシステムでは，入力が容易で出力が手書き形式に近くなるように，大きな努力が払われている．次に SCRACHPAD による不定積分の例を示す．積分ルーチンを呼び出したあと，

```
(log(x)*(1-x)-1)/(exp(x)*log(x)**2)
```

を入力すると

$$\frac{X}{EXP(X)LOG(X)}$$

なる答が 631 ミリ秒後に出力される．一方 exp(x**2) を入力すると，116 ミリ秒後

EXPRESSION IS NOT INTEGRABLE

なる答が出力される．なお，大方のシステムでは，巨大桁数の整数（たとえば 100!）を含む式の演算が可能であることを付記しておく．

4. SYMSAC '76

SYMSAC (SYMposium on Symbolic and Algebraic Computation) は ACM の専門分科会 SIGSAM (Special Interest Group for Symbolic and Algebraic Manipulation) が 5 年ごとに開催しているシンポジウムで，その 3 回目が昭和 51 年 8 月，New York 州 Yorktown Heights の IBM 研究所で開催された．発表論文は 54 編，参加者は約 150 名，日本からの論文は 1 編（後藤，金田（東大・理）[20]）で参加は 2 名（後藤（東大・理），一松（京大・数解研））であった．論文の内容は，計算機を用いた群論および整数論，アルゴリズム，処理系，それに応用の 4 種に大別できる．

整数論，群論，それにグラフ理論の分野では早くから研究用に計算機が用いられており，それらは非数値演算への代表的応用例と言えよう．SYMSAC '76 においても有限群論の研究が主要テーマの一つになった．この方面では Cannon を中心とするオーストラリアのグループが世界をリードしている．Cannon は

Neubüser と共に，群の構造の研究用に GROUP なるシステムを作りあげているが，システムをさらに強力にするため，GALOIS と CAYLEY と名づけた言語を設計中である．CAYLEY の言語のスタイルは非常に洗練されており，Cannon は群論の研究のみならず，言語設計と言語処理系の作成にかけても世界一流の力量をもっていることを示した．Illinois 大学では Pless が，GROUP の機能をさらに新しい分野の研究に拡張すべく，CAMAC なるシステムを作成した．その他にも，群論の個々の問題向きに数多くのシステムが作成されたことが各講演からうかがえる．SYMSAC '76 で発表された有限群論の最近の進歩に関する諸講演の内容は，専門外の筆者らのよく理解できるものでないが，同シンポジウムには一松教授も出席されておられたので，これを補っていただけるかと思う．

アルゴリズムに関する講演は全体の 1/3 以上を占め，きわめて多岐にわたり，内容的にも重要な進歩が多く見られた．その一つは代数的数体上における因数分解の新しいアルゴリズムの発見である．体 k 上で多項式が因数分解され，代数的数 α を定義する k 上での定義多項式 $\varphi(\alpha)=0$ が既知ならば，すべての分離的な単純代数的拡大体 $k(\alpha)$ においても，多項式が有限回の手続きで因数分解されることは教科書[9]にも記載されている．残念ながらその方法は時間がかかりすぎて実用的でない．MIT の Trager は同じ定理に基づきながらも，因数分解を高速に行う単純なアルゴリズムを見い出し，分離体を計算するアルゴリズムも提案した．後者は Risch のアルゴリズムにおける log の項の計算に直ちに応用できる．Boston 大学の Epstein は有理関数の任意の有限集合 $\{R_i\}$ が擬乗法独立かどうか，すなわち $\prod R_i^{n_i}$ が定数かどうか，を判定するアルゴリズムを見い出した．これを用いると $\{\log R_i\}$ が一次独立かどうかを判定でき，Risch のアルゴリズムの不完全さ（被積分関数に含まれる超越的要素のうち独立なものを判定できない）が部分的に補われる．

数式の簡単化に関してドイツの Lauer が，多項式イデアルの剰余類においては標準形が存在することを証明した．この存在性は，1974 年の EUROSAM シンポジウムで否定的に推測されていたものであるが，多項式で表現された条件下で多項式の簡単化を計る際に有用である．また，この方面の大御所 Caviness と Fateman から，有理式に根基を一重にのみ作用させて生成される数式の簡単化について，MACSYMA への組み込みと関係づけて報告された．

多項式の乗算について，この方面のエキスパートの Moenck により種々のアルゴリズムの精ちな比較がなされた．多項式の次数より十分大きい配列を用いれば，粗多項式の加減算が $O(N)$，並算が $O(NM)$ の操作数で実行できることが IBM の Gustavson と Yun から指摘された．ここで N, M は二つの多項式の非零係数の項数であり，$N \geqq M$ とする．東大の後藤と金田は，ハッシュ符号法を用いれば，多変数多項式においても巨大配列を用いることなく，同じ高速化が実現できることを指摘した．多項式に関してはそのほか，無平方分解（任意の多項式 P に対し，$P=p_1^1 p_2^2 \cdots p_k^k$；$n>1$ なる整数に対し $p_i=q_i^n$ なる多項式 q_i は存在せず，$i \neq j$ であれば，$\gcd(p_i, p_j)=1$，なる $\{p_i\}$ を見い出すこと）や剰余列に関するアルゴリズムの改善などに加えて，完全 n 乗化（たとえば x^2+4x+3 が与えられたとき，1 を加えて $(x+2)^2$ を作る）と合成（与式 u に対し，$u(x)=v(w(x))$ なる多項式 v と w を見い出す）のアルゴリズムは，"身近な" 問題をとらえた研究として新鮮味を与えた．特に合成のアルゴリズムは，MIT の大学 2 年生である 17 才の Barton が堂々と発表して話題を呼んだ．

その他のアルゴリズムでは，粗行列の逆行列や行列式の高速演算などに加えて，Toronto 大学の Lipson は関数の近似を $(\mathrm{mod}\ t^n)$ で定義し，高次の近似を，方程式の根を求める Newton の反復公式を用いて求めるアルゴリズムを与えた．IBM の Yun は，この方法は一般性をもち，種々の応用の道があることを指摘した．さらに二人の大家 Caviness と Collins が共同で，ガウス整数（整数$+\sqrt{-1}\times$整数）に関する新しいアルゴリズムを発表して注目された．一方，プログラムの番外で，不定積分アルゴリズムの教祖 Risch が新しい積分アルゴリズムを発表し多大の反響を呼んだ．この新アルゴリズムはプログラミングが容易で，Risch の所属している IBM 研究所では 3 週間でプログラムが動いたそうである．

一般目的の処理系に関する講演では，MACSYMA の多倍長浮動小数点数の演算機構の説明，REDUCE のデータ構造と今後の改良予定，素粒子物理学上の計算に用いられている，FORTRAN で書かれたシステム ASHMEDAI の紹介などで，特に目をひくものはなかった．数式の演算速度は用いるアルゴリズムにもよるが，数式の内部表現と計算の実行方法にも大きく依存する．その意味で，FORMAC を維持している Bahr から内部表現と関係づけて，多項式の積とベキ

乗に関する種々のアルゴリズムが比較検討されたこと，Jenksによる，SCRATCHPADに組み込まれた，コンパイラ方式によるパターン・マッチの講演は検討に値する．Cambridgeのグループからは，小記憶容量の計算機で数式処理の研究と教育を行うため，COBOLに立脚したシステムをデザイン中であることが述べられた．特定問題向きの処理系としては，群論用のものを除けば，発見的解法に基づく1階1次の微分方程式の解法システムが発表されたのみである．

今回のシンポジウムの特徴の一つに，従来数値計算のテーマであった問題を数式処理の問題として取りあげた講演が多かったことが挙げられる．特に，多項式の根を扱った論文は5編にのぼる．これらの論文で扱われた手法は，Newtonの反復法やSturmの定理に基づく方法など，目新しくはないが，まるめの誤差に無関係な方法として注目される．さらに，数値計算の道具として使うために関数をベキ級数に展開するとか，多変数関数の近似式を非常に多数の項の和で表わすために数式処理の手法を利用する話も報告された．群論における群の位数などのように正確な数を求めるのが目的でない場合には，理工学上の計算では最終的に数値計算に移行するのが大部分であるから，数値計算に関連した論文が数多く現れるのは当然であるし，数式処理もいよいよ広く実用的になってきた証と言うべきかもしれない．

その他の応用例としては，MACSYMAに楕円積分のアルゴリズムを組み込む話，MACSYMAを使用して微分方程式の解を漸近展開形式で対話的に求めた話などに加えて，変ったところでは，数式の定性的性質(実？，正？，連続？，微分可能？，など)の分析に数式処理のアルゴリズム（簡単化など）を利用した例もある．数式処理を純粋に理工学方面に応用した報告は二つだけであったが，これは発表がむしろそれぞれの専門学会で行われるためであろう．応用面における現状を把握するために，応用分野の論文題名などをSIGSAM Bulletinなどに記載することが望ましい．

5. 問題点と将来

現在世界で使われている数式処理システムのうち，MACSYMA, SCRATCHPAD, REDUCEなど主要なものはいずれもLISPをホスト言語として書かれており，Cambridge大学のCAMALもLISPに書きかえるとのことである．これらの作成者の語るところによれば，“LISPが特によい言語とは思わなかったが，記号処理言語として他にもよいものがないから使った”とのことである．LISPの開発者の一人であるMinskyも，“数式処理の可能性を追求する段階ではとりあえずLISPを使い，後にもっと能率のよいものに切り換えればよいだろう”と言っていたとのことであるが，これがいつの間にか定着してしまったものらしい．LISPも初期のものは非常に遅かったが，最近はコンパイラが改良されて速度が向上した．MITのPDP 10では，数値計算プログラムなら，LISPはFORTRANなみの速さにまでなっている．しかしこれは，数値計算ではデータの型をコンパイラに対して宣言し，これによって計算機組み込みの浮動小数点演算機構と短整数演算機構をきわめて有効に利用するためである．ところが数式処理の場合には，データの型を実行時に検査しなければならず，今日の計算機ではソフトウェアがこれを実行する．したがって，それだけでも速度は一桁近く落ちる．このように，今日の計算機のハードウェアは数式処理に適してはいない．またソフトウェアの面でも，数式処理用の記号的アルゴリズムを記述するのに，15年も前に作られたLISPがほとんどそのままの形で使われている点も問題であろう．将来の数式処理システムは，数学の公式集に集約されている知識(特殊関数，微分方程式の解析的解法，等々）を次々と活用する方向に進むであろうが，これは必然的にプログラミング・システムの巨大化を招き，プログラミングの方法論を確立しないことには保守と運営に支障をきたすようになるであろう．

一方，理工学の諸分野には，数式の操作以外にも，計算機でもやれるはずのことを人間がこつこつと行っている操作が多数あり，それらは数式処理と同様に次第に計算機化されていくものと思われる．現にその実例としては有限群論の研究（前章を参照のこと）や量子電気力学におけるグラフ処理[42]がある．さらに，数値計算の補助手段あるいは補充手段として数式処理が利用され，数値計算と数式処理の間にある壁が次第に薄くなっていくものと思われる．

6. おわりに

数式処理では，群論を除けば，アメリカが世界をリードしており，なかでもMITがが最も進んでいる．ひるがえって国内では，1974年7月に記号処理シンポジウム[43]が開催され，ついで1975～6の2年間，記号処理研究委員会[44],[45]が設けられ，ようやく基礎ができあがりつつある．従来は処理系の作成に片寄っていた

観があるが，今後はアルゴリズム，応用面を含めて，広く研究の輪が広がることを期待したい．この方面での研究の発展のため，討論と情報交換の場を今後も設けて，組織的に推進することが望まれる．なお，本稿の作成にあたり，記号処理研究委員会での討論が大いに参考になった．関係諸氏に感謝の意を表する．

参 考 文 献

1) J. R. Slage (南雲仁一，野崎昭弘共訳)：人工知能，産業図書，東京 (1974).
2) D. Barton & J. P. Fitch: A Review of Algebraic Manipulative Programs and Their Application, Comput. J. Vol. 15, No. 4, pp. 362～381 (1972).
3) Proc. 2 nd. Symp. on Symbolic and Algebraic Manipulation: edited by S. R. Petrick, SIGSAM, Ass. Comput. Mach., New York (1971).
4) Proc. 1974 EUROSAM Conference: edited by R. D. Jenks, SIGSAM, Ass. Comput. Mach., New York (1974).
5) Proc. 1976 ACM Symp. on Symbolic and Algebraic Computation: edited by R. D. Jenks, SIGSAM, Ass. Comput. Mach, New York (1976).
6) W. S. Brown: On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors, J. ACM Vol. 18, No. 4, pp. 478～504 (1971).
7) J. Moses & D. Y. Y. Yun: The EZ GCD Algorithm, Proc. 1973 ACM National Conference, pp. 159～166 (1973).
8) D. Y. Y. Yun: The Hensel Lemma in Algebraic Manipulation, Doctor Thesis MIT (MAC TR-138) (1974).
9) B. L. van der Waerden (銀林浩訳)：現代代数学 1，東京図書，東京 (1959).
10) E. R. Berlekamp: Factoring Polynomials over Large Finite Fields, Math. Comp. Vol. 24, No. 111, pp. 713～735 (1970).
11) D. E. Knuth: The Art of Computer Programming Vol. 2, Seminumerical Algorithms, Addison-Wesley (1971).
12) H. Zassenhaus: On Hensel Factorization—I, J. Number Theory Vol. 1, pp. 291～311 (1969).
13) P. S. Wang & L. P. Rothschild: Factoring Multivariate Polynomials over the Integers, Math. Comp. Vol. 29, No. 131, pp. 935～950 (1975).
14) J. Moses: Symbolic Integration: The Stomy Decade, Communs. ACM Vol. 14, No. 8, pp. 548～560 (1971).
15) 一松 信：初等関数の数値計算，付録A，教育出版，東京 (1974).
16) R. H. Risch: The Problems of Integration in Finite Terms, Trans. AMS Vol. 139, pp. 167～189 (1969).
17) W. M. Gentleman & G. Sande: Fast Fourier Transforms-For Fun and Profit, Proc. AFIPS 1966 FJCC Vol. 29, pp. 563～578 (1966).
18) S. C. Johnson: Sparse Polynomial Arithmetic, SIGSAM Bulletin Vol. 8, No. 3, pp. 63～71 (1974).
19) E. Horowitz: A Sorting Algorithm for Polynomial Multiplication, J. ACM Vol. 22, No. 4, pp. 450～462 (1975).
20) E. Goto & Y. Kanada: Hashing Lemmas on Time Complexities with Applications to Formula Manipulation, in (5), pp. 154～158.
21) F. Gustavson & D. Y. Y. Yun: Arithmetic Complexity of Unordered Sparse Polynomials, in (5), pp. 149～153.
22) W. S. Brown: Rational Exponential Expressions and a Conjecture Concerning π and e, Amer. Math. Mon. Vol. 76, pp. 28～34 (1969).
23) J. Moses: Algebraic Simplification: A Guide for the Perplexed, Communs. ACM Vol. 14, No. 8, pp. 527～537 (1971).
24) B. F. Caviness: On Canonical Forms and Simplification, J. ACM Vol. 17, No. 2, pp. 385～396 (1970).
25) D. Richardson: A Solution of the Identity Problem for Integral Exponential Functions, Z. Math. Logik Vol. 17, pp. 133～136 (1971).
26) D. Richardson: Some Unsolvable Problems Involving Elementary Functions of a Real variable, J. Symb. Logic Vol. 33, pp. 511～520 (1968).
27) J. V. Matijasevic: Enumerable Sets are Diophantine, Soviet Math. Dokl. (trans.) Vol. 11, pp. 354～357 (1970).
28) M. Davis: Hilbert's Tenth Problem is Unsolvable, Amer. Math. Mont. Vol. 80, No. 3, pp. 233～269 (1973).
29) P. S. Wang: The Undecidability of the Existence of Zeros of Real Elementary Functions, J. ACM Vol. 21, No. 4, pp. 586～589 (1974).
30) M. I. T. The MATHLAB Group: MACSYMA Reference Manual, Laboratory for Computer Science, M. I. T., Massachusetts (1974, 1976).
31) A. C. Hearn: REDUCE-2 User's Manual, University of Utah, Utah (1973).
32) J. P. Fitch: CAMAL User's Manual, University of Cambridge, England (1975).
33) A. D. Hall Jr.: On ALTRAN System for Rational Function Manipulation-A Survey, Communs. ACM Vol. 14, No. 8, pp. 517～521 (1971).
34) K. Bahr: Toward a Revision of FORMAC, SIGSAM Bulletin Vol. 8, No. 1, pp. 10～16

(1974).
35) R. D. J.enks: The SCRATCHPAD Language, SIGSAM Bulletin Vol. 8, No. 2, pp. 20～30 (1974).
36) 池原悟, 他: AL仕様書, 電電公社横須賀通研, 神奈川 (1975).
37) 渡辺隼郎: 常微分方程式の数式処理, 教育出版, 東京 (1974).
38) E. Goto: Monocopy and Associative Algorithms in an Extended LISP, Technical Report 74-03, Imformation Science Laboratory, University of Tokyo, Tokyo (1974).
39) 後藤英一: LISP 入門, bit 連載, Vol. 6, No. 1 (1974)～Vol. 7, No. 2 (1975).
40) Y. Kanada: Implementation of HLISP and Algebraic Manipulation Language REDUCE-2, Technical Report 75-01, Information Scinece Laboratory, University of Tokyo, Tokyo (1975).
41) M. Terashima: Algorithms Used in an Implementation of HLISP, Technical Report 75-03, Information Science Laboratory, University of Tokyo, Tokyo (1975).
42) T. Sasaki: Automatic Generation of Feynman Graphs in QED, J. Comp. Phys. Vol. 22, pp. 189～214 (1976).
43) 記号処理シンポジウム報告集, プログラミング・シンポジウム委員会, 情報処理学会 (1974).
44) 昭和50年度研究報告集, 記号処理研究委員会, 情報処理学会 (1976).
45) 昭和51年度研究報告集, 記号処理研究委員会, 情報処理学会 (1977).

(昭和51年12月7日受付)