数　式　処　理*

黒沢俊雄** 萩野正彦** 小泉　修**

1. まえがき

数値計算の計算機応用は，stored program 方式の計算機が出現して最初の応用であったが，大変大きな成果をあげている．

しかし，記号の数学はどうであろうか．数値計算ほどの普及はみられないが，各分野での活発な研究，開発がすすめられかなりの成果が散見されるようになった．

1966年3月，記号と代数処理に関するACMのシンポジウムがワシントンで開かれたのを皮切りに，同年9月ピサでの IFIP の大会において，記号処理のための言語と技術についてのシンポジウムが開かれ議論が交わされた．

わが国においても，昨年1月のプログラミング・シンポジウムでは，数式処理が主テーマの一つとしてとりあげられ，多くの成果の発表，アイデアの交換が行なわれるなど，活発な活動をみるにいたった．

ここでは，数式処理の概念と方法について解説し，代表的な数式処理言語およびその応用例を紹介することにする．

数式処理には4種の形がある．まず第1に，単純ではあるが，量が多く手計算では間違いのおこしやすい式の計算である．第2は，綿密な思考と技巧を要する特殊な問題である．収束，発散などの極限をあつかう問題で，数値計算上では解決が困難な問題がこの範疇にはいる．第3は，段階をふんで計算を進めなければならないような問題で，前段階の計算結果によって，つぎの段階の計算の方法がきまるような形である．これは man-machine の対話形言語による操作がのぞまれる．そして最後に，数表，公式集，ハンドブックや辞書を自動的にひくようなもので，公式や，変形の規則を検索する方法である．これは scientific-algorithm retrieval と呼ばれている．

* Formula Manipulation, by Toshio Kurosawa, Masahiko Hagino and Osamu Koizumi (Japan Proccess Consultant Co. Ltd.)

** 日本プロセスコンサルタント株式会社

2. イントロダクション

Symbol manipulation は，記号処理とやくされ，記号としてとりあつかわれる非数値情報を処理するところから，数値計算に対して非数値計算(non-numerical computation) などとも呼ばれている．

たとえば

(1)　$(aX+b)(X+c) \rightarrow aX^2+(ac+b)X+bc$

(2)　$X^2+aX+bX+ab \rightarrow (X+a)(X+b)$

などの式の展開や因数分解などを思い出していただければ理解されよう．

もう少しレベルを上げれば，式の微分や積分がある．

(3)　$\frac{d}{dX}[X^3] \rightarrow 3X^2$

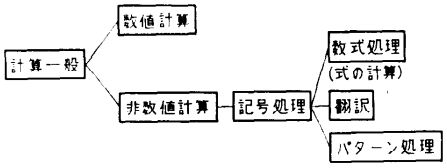
(4)　$\int[\sin X] \rightarrow \cos X$

(1)と(2)の左辺，(3)，(4)の左辺の〔　〕内の式にある操作を加え，右辺の形に式の変換をほどこす．このような作業が記号処理である．この場合いくつかの変換のルールを適用することによって行なわれる．機械翻訳，言語プロセッサーなども記号処理である．

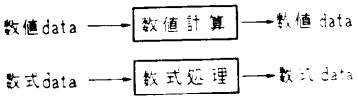
非数値情報をとりあつかう問題となると，きわめて広い範囲におよぶわけであるが，その中で，数学的な問題の中心的なものとして数式処理 (Formula manipulation) がある．

これは，いままで計算機をとりあつかってきた人や数学者にとって，最も興味ある問題であろう．数式処理を式の計算とか代数処理 (Algebraic manipulation) とも呼ばれている．

第1, 2図は数値計算と，数式処理の関係を図示した



第1図



第2図

ものである．これを定式化するために，2つの数値 data x, y と，2つの数式 data f, g と設定する．y は x にアルゴリズム A_n をほどこした結果で，g は f にアルゴリズム A_f をほどこした結果である．

$$A_n\{x\} \rightarrow y$$
$$A_f\{f\} \rightarrow g$$

ここで，A_n は数値計算のアルゴリズムで，A_f は数式処理のアルゴリズムである．

式の計算の典型的な問題に次のような問題がある．

1. 式の微分
2. 式の積分
3. 式の因数分解
4. 式の展開
5. 多項式の計算
6. 式の整頓
7. 式の比較
8. 代入
9. 微分方程式
10. 積分方程式

式の微分は，計算機の非数値数学への応用として，第一にとりあげられた問題であって，1953年に kahrimanian によって，はじめて手がけられた．

当時のプログラムは，機械語か，低いレベルの言語でかかれたものと思われ，式の表現形式も**第3図**のよ

式	pseudocode			
$W = C_0SV$	C_{00}	$_{00}V$	000	$_{00}W$
$V = X^2$	E_{00}	$_{00}V$	$_{00}2$	$_{00}V$

第3図

うに，12字の string による 3 address pseudocode のテーブル形式で表現したものであった．つづいて Nolan や Hellerman らによってとりあつかわれ，この種の問題に努力がはらわれた．

Nolan の方法でも，計算する公式を入力および出力する表現方法に，12字の string からなるテーブル形式をとった．高次微分は一次微分の組合せによって表現し，微分は一番外側のカッコから recursive に，微分演算子がなくなるまで行なう方法をとった．

1962年ごろになると，数式をもっと数学的に表現できるようになった．Hanson, Caviness や Joseph らによる方法では，FORTRAN の演算式と同様に演算子として

$$+, -, *, /, (,)$$

の使用がゆるされ，exp や三角関数，逆三角関数，楕円関数，対数などの関数も式の中に表現できるようになった．入力された式は内部で三又木に変換され，演算法則がこの木に適用される．三又木の1本が演算子で他の2本がオペランドである．

3. 数式処理用プログラミング言語

LISP[7] や IPL[12] などの記号処理用言語が開発されると，式の計算のプログラムの開発がますます盛んになってきた．

しかし，LISP や IPL は記号処理用言語ではあるが，汎用の記号処理用言語であるので，その応用範囲は広いが，式の計算に必要な機能が豊富に備えてあるわけではなく，微分その他のルールを，いちいちサブルーチンとして準備する必要があった．

数式処理用の言語が，システムとして必要になってきた．その必要性から数式処理専用の言語の出現をみるにいたった．IBM 7090 用に開発された FORMAC (Formula manipulation compiler)[2][4] などがそれである．

その他に，最近開発されたものには ALPAK, Formula ALGOL[13] がある．これらはいずれもある形の数式処理能力を備えたプログラミングシステムである．これらの中で Formula ALGOL と FORMAC は一般式を取り扱うことができる．ALPAK は多項式および有理関数のシステムである．これらのプログラミング・システムの実際問題への応用について述べた文献もある．

最初から ALPAK 言語は FAP（アセンブラー）であった．たとえば多項式の比をあつかうには，ALPAK サブルーチンへの呼出し命令を含んだ ALPAK マクロ命令を用いてコーディングが行なわれた．ALPAK システムに対して，さらに高級な言語 ALPAK が開発された．

Formula ALGOL は ALGOL を拡張したものであって，数式処理，リスト処理および有限ストリング処理のための機能を持っている．それは ALGOL の数値上ならびにコントロール上の能力を持っており，また，リスト処理およびストリング処理の能力を持つと

いう点で，ALPAK や FORMAC よりも一般的なプログラミング言語である．Formula ALGOL は特別な数式演算の命令を持っていない．使用者が独自の数式の演算のプロセスをプログラムすることができるようになっている．たとえば，使用者は独自の展開のルーチンや式の簡単化のルーチンをプログラムしなければならない．

最初の Formula ALGOL は ALGOL 言語を試験的に拡張したものであった．改訂された Formula ALGOL もある．このシステムは1966年3月の ACM のシンポジウムで発表された．

FORMAC 言語は FORTRAN Ⅳ の拡張である．したがって，FORTRAN Ⅳ を適当なサブセットとして含んでいる．入力や出力，繰返しのコントロールのステートメントはもとより，FORTRAN の数値上の演算の能力は，すべて FORMAC においても有効である．FORMAC は EXPAND (展開) のような特別な命令と自動数式簡単化 (AUTSIM) とをシステムの中心部分として備えている．そしてこのシステムの特長は

(1) 覚えやすく使いやすいこと．
(2) このシステムは任意の複雑さと深さを持った関数のネストからなる初等関数を含んだ一般の数式をとりあつかうことができる．
(3) 数式の自動簡単化ルーチンと万能微分作用素がシステムの一部として用意されている．
(4) 有理係数や浮動小数点係数を持った式をとりあつかうことができる．
(5) FORMAC の記号処理能力と FORTRAN の数値計算能力とは，記号分析と数値計算とが同一のプログラムで実行できるように接続可能である．
(6) FORMAC プログラミング・システムは汎用モニターシステム IBSYS の能力が活用できるように組み込まれている．

リスト処理能力や記号処理能力を備えた汎用プログラム言語に PL/1 がある．PL/1 はまた，bit 処理能力も持ち，小さな容量の計算機でも，多量の情報を記憶する能力を持っているので，大容量を必要とするこの種の問題に適している．

この理由から，PL/1 がこの種の問題のために，注目されていることは，当然といえよう．

PL/1 を基礎とした数式処理用言語として，PL/1 FORMAC の開発が計画されている．これは FORMAC と同様な方式によって，PL/1 を拡張し，数式処理能力を組み込んだシステムである．

APL (associative programing language) もまた PL/1 を拡張した言語である．これは PL/1 の機能のほかに，同様の要素を集めたり，要素の集合をとりあつかったりすることができる．また，要素の集合を network としてとりあつかうこともでき，階層構造を持たせることもできる．これは data の ENTITY と ATTRIBUTES によって連想的に行なわれる．

その他，数式処理用，または記号処理用言語が開発されているが，その代表的なものの特長を**第1表**に示した．

4. 応 用 例

すでに実用化されている応用例は少なくない．その中のいくつかを紹介しよう．これらの例から，応用範囲が際限なく広いことに気づく．

この節での実例の中で，原子核の三体問題，逆行列の計算，非線形回帰は，筆者が三菱原子力工業株式会社に在職中，同社上村義明部長付，前田英次郎技師，京都産業大学藤井宏講師ら，数式解析グループと着手した問題である．未知の分野であり，暗中模索した時代がなつかしい．この経験から数式処理の可能性と，問題点が明確となった．未解決のままに残されている問題もいくつかあるが，かなりの成果があったものと思う．このほか同グループが着手した問題に，Stiffness matrix の計算などがある．これらの実例は，昨年1月のプログラミング・シンポジウムで発表済みであるので，アルゴリズムの詳細については，同報告書を参照されたい．

その他の例は，Robert G. Tobey によって報告された Formac などの実用例である．ここに論じていないものに，プラズマ物理学，ヤコービ多項式の生成，安定性分析，有理関数の記号での積分，結晶体理論，対称多項式，Frame の収束値の生成，Fourier 波形解析，宇宙船再突入の研究や，計算機工学においては論理回路の自動設計などがある．

4.1 非線性回帰

いくつかの変数の関数のノルムを最小にするようなパラメータの値を決める問題である．すなわち，

独立変数 $\boldsymbol{X}=(x_1, x_2, \cdots x_n)$，観測値 $y(\boldsymbol{X})$ の数値表と，関数 $f(\boldsymbol{X}, \boldsymbol{a})$ が記号で与えられた場合

$$\| f(\boldsymbol{X}, \boldsymbol{a}) - y(\boldsymbol{X}) \| \qquad (4.1.1)$$

を最小にするようなパラメータ $\boldsymbol{a}=(a_1, a_2 \cdots, a_F)$ を決

第1表　非数値計算用語の比較表

		Data Format (Input/Output)	Internal Data Format (in Core)	Program Description	Implementation
List Processor	LISP-1·5	Parenthesized list structures and dotted pairs	Binary trees	Elementary operation & subroutine call	IBM 709/90 IBM 1620 B 5500
	IPL-V	List of elements which name data terms or other list	Binary trees	Recursive function & conditional expression	IBM 7090 CDC 1604 UNIVAC 1105
	SLIP	Parenthasized list structures	Headed list of 2 word	Host language extended with list processor	IBM 7090/94 CDC 1604/3200 IBM 360
Linked Block Language	L⁶	Character string, formatted under program control	Linked blocks of various sizes linkages and fields specified programmer	Conditional statements involving concatenated field identifiers	IBM 7094 IBM 360 GE 635/45
	CÖRAL	Character string lightpen or button action	Linked block of formant specified at start of run	Obscure notation	MIT TX-2
	APL	Such as PL/1	Such as L⁶	Facilities for establishing relations between the data elements, developing and restructuring the data structure	extended PL/1 (GM)
String Processor	COMIT	Strings of characters which may have subscripts	Linked 2-word blocks with short-cut links	Pattern detected string processing	IBM 7090/94 IBM 7040/44
	SNÖBÖL	Character string	Indexed blocks of string symbols	〃	IBM 360/40 IBM 7090/94 IBM 7040/44 RCA 601/604 CDC 3100
	TRAC	String of Characters	Linear string or Binary tree	Nested function	IBM 7094 (MIT) GE data net 30
Formula manipulators	FORMAC	Mathematical expressions representing explicitly defined real elementary function	Polish strings	FORTRAN IV extended for formal algebraic operations	IBM 7090/94 TOSBAC 3400 (KO Vniv.)
	ALPAK/ ALTRAN	Mathematical expressions representing rational functions	Graphs with no loops	FORTRAN II extended for rational functions	IBM 7040/44 IBM 7090/94
General Purpose Language	PL/1	Number, truth value arrays, character strings	Number, array, linear list etc.	New programing language with numerical and non-numerical caluctation capability	IBM 360
	Formula ALGOL	Number, truth value. arrays, algebraic formula list structure	Binary tree, number and array, linked list	ALGOL extented for formula and list processing	IBM 360-67
	LISP 2	Character strings parenthesized list	Binary tree, number and array	Function defined by ALGOL like structure	IBM 360-65

定することである．ここで

$$f(\boldsymbol{X},\boldsymbol{a})=\sum a_i f_i(\boldsymbol{X}) \qquad (4.1.2)$$

$f_i(\boldsymbol{X})$ は記号のまま有理関数，三角関数，指数関数など，FÖRTRAN でとりあつかえる初等関数を入力することができる．このプログラムでは，f とその第1次導関数を計算する必要がある．f の微分と，evaluate は数式処理の技法を用いた．

従来の方法では，f の微分や evaluate のサブルーチンを手で作りメインと結びつける必要があった．この一般化されたプログラムによって，特殊なサブルーチンの作成などの，わずらわしさから解放されるようになった．

4.2　逆行列の計算

逆行列の計算は数式処理の基本的なアルゴリズムを使用する．ここでとりあつかう行列の元はすべて数式である．

$$A^{-1}=\frac{1}{|A|}\begin{pmatrix}A_{11}\cdots\cdots A_{n1}\\ \vdots\\ A_{1n}\cdots\cdots A_{nn}\end{pmatrix} \qquad (4.2.1)$$

$|A|$ を求めるアルゴリズムとして，式の展開と簡単化のアルゴリズムを用いた．

4.3　三体問題

proton-neutron-neutron, proton-proton-neutron の三体問題の固有関数を Rayleigh-Ritz 法を用いて求める問題である．

(4.3.1) 式の2次形式の係数を計算することであるが，Ψ は大村形波動関数であり，(4.3.2) で与えられる．

$$\int\Psi\cdot A\Psi\,d\tau=\sum a_{ij}x_i y_i \qquad (4.3.1)$$

$$\Psi=\sum x_i\cdot\Psi_i \qquad (4.3.2)$$

ここで，

$$\Psi_i=\prod_{j=1}^{3}\{\exp\{-\mu(r_j-D)\}-\exp\{-\mu(r_j-D)\}\cdot P_i(r_1,r_2,r_3)\} \qquad (4.3.3)$$

$$P_1=1 \qquad P_2=r_1+r_2$$

$$P_3=r_1^2+r_2^2 \qquad P_4=(r_1+r_2)\cdot r_3$$

$$P_5=r_3 \qquad P_6=r_3^2$$
$$P_7=r_1r_2 \qquad P_8=r_2-r_1$$
$$P_9=r_2^2-r_2^2 \qquad P_{10}=(r_2-r_1)r_3$$
$$P_{11}=r_1^2r_2-r_1r_2^2+r_2^2r_3-r_2r_3^3+r_3^2r_1-r_3r_1^2$$

演算子Aは，微分演算子Kなどであり，積分$\int f\cdot d\tau$は(4.3.5)によって定義されている．

$$K=\sum_{\substack{i=1,2,3\\j=2,3,1\\k=3,1,2}}\left[\frac{1}{r_i^2}\cdot\frac{\partial}{\partial r_i}\cdot\left(r_i^2\frac{\partial}{\partial r_i}\right)+\frac{r_j^2+r_k^2-r_i^2}{2r_jr_k}\cdot\frac{\partial^2}{\partial r_j\partial r_k}\right] \quad (4.3.4)$$

$$\int f\cdot d\tau=\int_D^\infty r_1dr_1\int_D^\infty r_2dr_2\int_D^\infty r_3\times f(r_1,r_2,r_3)dr_3$$
$$-\int_D^\infty r_2dr_2\int_D^\infty r_3dr_3\int_D^\infty r_1\cdot f(r_1,r_2,r_3)\times d(r_1-r_2-r_3)$$
$$-\int_D^\infty r_3dr_3\int_D^\infty r_1dr_1\int_D^\infty r_2\cdot f(r_1,r_2,r_3)\times(r_2-r_3-r_1)$$
$$-\int_D^\infty r_1dr_1\int_D^\infty r_2dr_2\int_D^\infty r_3\cdot f(r_1,r_2,r_3)\times d(r_3-r_1-r_2) \quad (4.3.5)$$

数式処理問題として，微分演算$K\Psi$と$\int f d\tau$を行なうことであった．微分演算$K\Psi$については，LISP, FORMAC, FORTRAN のそれぞれを用いて行なった．

積分$\int f d\tau$は分解すれば$\int_\beta^\infty r^n\cdot\exp(-\alpha r)dr(\alpha>0)$の形に帰着される．したがって，原理的には数式をじかに処理することも可能であるが，計算量が厖大となる．$\int\phi\cdot A\phi d\tau$について簡単なものについて数式処理の方法で試みたが，最終的には FORTRAN で解いた．

FORTRAN では数式を直接処理するのはさけ，被積分関数を(4.3.6)の形の項の和として考え，これを単位としてあつかった．

$$P(r_1,r_2,r_3)\exp\left\{\sum_{k=1}^{3}d_k(r_k-D)\right\} \quad (4.3.6)$$

ここでPは多項式

そこで多項式の表現もテーブル式にした．たとえば，$P=r_1r_2^2-3r_2^3r_3+5r_1-2$については下図のような記憶法を用いた．

LENGTH	4			
COEF	+1	−3	+5	−2
POWER	1	0	1	0
	2	3	0	0
	1	0	0	0

式の性質をあらかじめ検討し，手で処理できる部分については，前もって計算し準備した．こうして容量，速度の制限からのがれることができた．

4.4 微分方程式

David Taylor Model Basin によって2つの微分方程式を解くプログラムが開発された．第1のプログラムは，Kutta-Merson 法による数値積分サブルーチンを用いて1組の常微分方程式を解くものである．第2のプログラムは，線形微分方程式を解くもので，近似解の式をあたえて Gram-Shmidt 法を基礎とした直交化技法による最小2乗近似を行なうものである．

4.5 不完全ベータ関数の計算

不完全ベータ関数の計算には，種分展開がよいとされている．マイアミ大学の Biometric Lab. では，この方法を用いてプログラムした．

4.6 非線形最尤推定

非線形方程式 $f_j(\boldsymbol{X},\boldsymbol{\theta})=u_j$ のパラメータの最尤推定のアルゴリズムに，Eisenpress と Greenstadt の方法がある．

このアルゴリズムでは $f_j(\boldsymbol{X},\boldsymbol{\theta})$ の θ_i および x_i に関する1次，2次，3次の偏微分が必要である．非線形回帰の場合と同様，数式処理の方法がある．このプログラムによって，つぎの2つの機能をはたすことができよう．

(1) 種々の方程式の組 $f_j(\boldsymbol{X},\theta)$, $g_j(\boldsymbol{X},\theta)$ を与えて，特定の観測値に適合する方程式の組を見つけ出す．

(2) モデルについての特定の式をあたえて，観測された $\boldsymbol{X}$ の種々の組のそれぞれに対応する θ の値を得る．

5. 計算時間と必要記憶容量

数式を直接代数的に演算する場合に，大容量の記憶装置を必要とし，計算時間もまた大きい．どの程度の計算時間と記憶容量が必要か，つぎの4項式の展開の実験によって示そう．これは

$$(x+y+z+u)^i=(x+y+z+u)^{i-1}(x+y+z+u)$$

をくりかえし計算し，新しい多項式を生成する問題である．この問題で，9乗の計算を行なっている途中でcoreが一杯になった．この実験はIBM 7090を使用した結果である．IBM 7090では，約20 kWのfree storageの使用がゆるされる（第2表）．

第2表 4項式の展開の計算時間

i	時 間 (sec)	次 数
2	1.75	4×4
3	3.03	20×4
4	8.45	35×4
5	14.55	56×4
6	26.55	84×4
7	43.17	120×4
8	68.40	165×4

MITで開発されたprogramにSAINT (Symbolic Automatic Integrator) があるが，MITの新入生の修了試験に出題された積分問題の54題のうち52題を解き，平均2分要している．

たとえば

$$\int e^{x^2} dx$$

の計算に1分半要し，$(1/2)e^{x^2}$ の解答を出した．使用計算機はIBM 7090であった．

6. むすび

数式の計算を計算機でやらせようとするといろいろの問題点がある．

計算機に大学受験の数学の能力をつけさせようとするには，計算機そのものの機能や，構造を改革しなければならない．どのように改革すればよいかは，明らかではないが，まず記憶形式がとりあげられよう．現在の計算機の記憶装置はアレーを中心としたものであり，マトリクスのとりあつかいまでは自在に操作できる．たてよこの単純な構造だからである．しかしながら，数式は自然言語ほどではないが，複雑な文法を持った表現であり，また習慣的な省略を用いたりするので，その処理のむずかしさがある．

したがって，記憶形式において，リスト構造を持った記憶装置や連想式記憶装置 (associative memory) のハードウェアを持てば，複雑な言語構造も記憶しやすく，認識も容易になるだろう．さらに学習機能とか自己改造能力を持てば，heuristicな問題をとりあつかう場合に，計算機に，いろいろな問題を解かせることによって，その間に計算機自身が経験によって熟練し，上達していくようになろう．そうなれば，むだなトライなどをさけ，計算時間も，実行可能なものになるであろうし，また思いがけない結果を生むにちがいない．

ICからLSI (Large scale integration) とそしてIEC (Integrated equipment component) へと，めまぐるしく進歩する電子工学と，これらを駆使し，さらにデザインオートメーションの技術を応用しながら，論理を組立てていく計算機工学の進歩によって，これらの機能の実現も夢ではないと思う．

謝辞 ご指導と貴重な資料を提供して下さった三菱原子力数式処理グループ，電力中央研究所若林 剛氏に感謝する．（昭和43年10月5日）

参考文献

1) Sammet, J. E: Suruey of the use of computers fōr non-numerical mathematics.
2) Bond, E. R FORMAC: experimental Formula manipulation Compiler user's reference manual.
3) Proceedings of the ACM Symposium on Symbolic and Algebraic Manipulation Communication of ACM Vol 1.9 #8.
4) Eliminating Monotonous Mathematics with FORMAC Commuincation of ACM Vol 1.9 #10.
5) George, G. D APL-a language for associative data handling in PL/1.
Proceedings-Fall Joint Computer Conference, 1966.
6) PL/1 Language Specifications IBM.
7) McCarthy, J. LISP 1.5 programmer's manual. MIT press.
8) James R. S.: A Heuristic Program that Solves Symbolicl Integration Problems in Freshman Calculus J. ACM.
9) William E. Ball: Robert I. Berns AUTOMAST: Automatic Mathematical Analysis and Symbolic Translation Communication of ACM Vol 9 #8.
10) Ball, W. E.: The SLIP system for information processing. Proc. 19th GUIDE Internatiōnal Meeting, Oct. 1964.
11) COMIT porgrammers reference manual, MIT Press.
12) Newell, A IPL-V: Infomation prōcessing language-V manual, Preutice-Hall, 1961.
13) Perlis, A. J. Apreliminary sketch of formula ALGOL. CIT Rep. April 9, 1965.