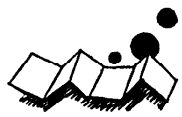
解　説



数式処理と数値処理との界面†

三　井　斌　友††

1. はじめに

今日コンピュータはいろいろなデータの処理と計算(manipulation and computation)を行っているが，筆者が主として関心をもっているのは数値計算 (numerical computation) と数式・代数計算 (symbolic and algebraic computation) である．一見すると，両者は計算——あらかじめ定義された規則のもとでのデータの数理的操作——という観点では同一のカテゴリに属し，数学的内容が濃いという面で，非常に近い関係にあるとみなされよう．ところが，現実には，少なくとも最近まで，両者の間の距離は大きく，相互交流は少なかったのである．

この原因は，コンピュータによる計算の歴史そのものにあると考えられる．実際コンピュータは，人間の手に負えない複雑・大量の数値計算を正確・高速に遂行するという需要を背景に，誕生したといって過言でなかろう．それから半世紀近くを経て，コンピュータによる数値計算は長足の進歩をとげた．かつて L. F. Richardson が提唱した，大劇場に何千人もの計算手を集め，その一人一人に空間の格子点のおのおのを分担させ，指揮者の合図に従って，隣同士の計算手の間のデータのやりとりと計算によって，大気の運動をシミュレートし，数値気象予報をするという構想も，並列コンピュータの実現で，かなりの程度現実のものとなってきた．

これに比べると，本特集の他の記事をお読みいただければ分かるとおり，コンピュータによる数式・代数計算の出発は後であるし，その進歩も遅々としていた．このような両者の関係は，数値計算と数式・代数計算を含む数学計算 (mathematical computation) として，われわれが現在享有しているものの，数千年にわたる歴史と非常に良く似ている．この事情については，筆者がほかで少々論じた[16]ので，詳しくはそちらに譲るとして，要するに，先発・後発という位置関係が，将来急速に縮まっていくことは必至である．

さて，数理的な問題解決は，①現象の定式化，②方程式の数学的解析，③数値計算方法の導出，④さまざまな条件下での数値計算，⑤結果の検討というプロセスの繰り返しとみてよかろう．従来は①～③を紙と鉛筆で行い，④をコンピュータによるというのが一般的であった．しかし，これからは②ないし③にも，数式・代数計算を導入して，問題解決過程全体にコンピュータを総合的に活用することになるであろう．同時に，このことが，数値計算，数式・代数計算の両者の発展，新しい展開を促すものと考えられる．

以上の観点から，コンピュータによる数値計算と数式・代数計算の界面を考察しておくことは，今日的重要性をもっている．ところが，先に述べたような両者の歴史的な相違から，界面の考察は十分であったとは言えない．数値計算の側では，暗黙のうちにアルゴリズムに数式計算を混ぜてはならないという了解が形成されてしまった．これは国内だけではないらしく，第1回数学ソフトウェアシンポジウム (1970年，Purdue) に招かれて，数値解析専門家を前に，"Software for Nonnumerical Mathematics" と題して講演した，数式処理の専門家 J. Sammet は，のちにこの場の雰囲気を「ライオンの巣の中のダニエル」(旧約聖書ダニエル書) のようであったと回想したという[20]．

人によっては，これを「LISP文化」と「FORTRAN文化」の相違，カルチュア・ショックと表現することもある．つまり，数値計算において主流である FORTRAN 言語と，それに依拠した思考方法と，かなり多くの数式処理言語の基礎である LISP と，その思考方法とのギャップは大きく，しかもそれぞれ強く自己主張しているというわけである．

しかし，目的は数理的問題解決にあるのであって，互いの領分争いにあるのではない．お互いに妙な我を

† Interface between Algebraic and Numerical Computations.

by Taketomo MITSUI (Department of Applied Science, Faculty of Engineering, Fukui University).
†† 福井大学工学部応用理学教室

張るのは，やめにすべきだろう．本論では，筆者の乏しい経験からではあるが，コンピュータによる数式処理と数値処理との界面について考察を行い，もって今後の発展のために一石を投じられればと願うものである．

なお，数式処理・数式計算といった用語について，一言触れておくと，かつては数式処理（formula manipulation）と呼ばれていたものが，最近は数式計算（あるいは数式・代数計算）と言われることが多くなった．内容の進歩と用語の変化はついてまわるものだが，本特集のタイトルにあわせて，以下数式処理に統一し，それと対照させる意味で，慣用ではないが，あえて数値処理という用語をとった．ご理解を願いたい．

2. 数式処理・数値処理の界面とは

そもそも，コンピュータによる数式処理・数値処理とはなにを指すのか，その定義をはっきりさせないと，両者の界面を云々することはできない．しかし，その厳密な定義があるわけでもなく，また厳密な定義のための努力が，それほど有用とも思えないので，さしあたり次のように考えておこう．

数値処理とは，有限桁の数体系のなかで，それらの四則演算を論理条件式を加えながら実行するものとする．一方数式処理とは，式（有理係数の多項式・初等関数の組み合わせとそれらの代数的拡張）に対して，可能な代数的演算（処理システムであらかじめ与えたもの，ユーザが定義したもの）を施すものとする．現在までのところ，数値処理言語と数式処理言語とは，お互いに他方の機能が不可能ではなく，若干は他方の機能を組み込んでいることもあるが，明確に分かれているから，数値処理言語で書かれたプログラムの実行が数値処理，数式処理言語でのそれが数式処理と言った方が，分かりやすいかもしれない．

このように考えると，数理的な問題解決過程のなかに，数式処理と数値処理の界面の可能性が浮かびあがってくる．

（1） 数式処理でえられた結果を，数値処理プログラムに渡し，計算を行う場合（図-1(a)）．

冒頭で分析した，数理的問題解決の過程にあるように，考察する問題ないし方程式系を，なんらかの方法で数値計算にのせるプロセスは，数式処理に向いている．たとえば，有限差分法や有限要素法のコードを数式処理によって生成することが試みられている．さらには，数値計算方法の解析，すなわち数値解析の研究のなかでも，数式処理が重要な役割を担いつつある．

（2） 数値処理でえられた情報を，数式処理プログラムに渡し，数式処理による解析に役立てる場合（図-1(b)）．

たとえば，非線型システムの相互関係のうちのいくつかが，特定の点では，数値計算の結果でゼロとみなしてよいと分かり，今後はそれを代数的にゼロとおいて，解析し直すという場合があろう．しかし，この場合の経験例はまだ少ない．

（3） 数値処理と数式処理を並行して実行し，両者の比較によって問題解決に役立たせる場合（図-1(c)）．

非常に簡単な例ではあるが，常微分方程式

$$\frac{dy}{dx}=\frac{x^2\sin y}{1+y^2},\quad x>0$$

が与えられたとしよう．1階常微分方程式を解くルーチンをもつ数式処理システム，たとえばMACSYMAにかけると，図-2のような解がえられる．ここで $LI_n(x)$ で表示される関数は

$$LI_1(x)=-\log(1-x),$$

$$LI_n(x)=\int_0^x\frac{LI_{n-1}(t)}{t}dt$$

で定義される超越関数，ATAN2(Y,X)は通常の記法では $\arctan(Y/X)$ と記される関数である．

本質的には不定積分アルゴリズムに帰着されるのだが，この結果は方程式が簡単な割には複雑であ

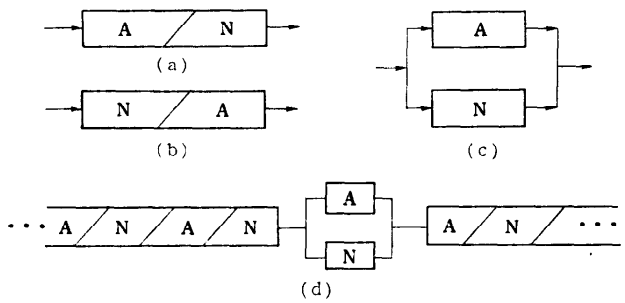


図-1　数式処理と数値処理の界面の概念
A：数式処理，N：数値処理

(D3)−(2%IY^3ATAN2(SIN(Y), COS(Y)+1)+2%IY^3ATAN2(SIN(Y), 1−COS(Y))
+Y^3LOG(2COS(Y)+2)+LOG(COS(Y)+1)−LOG(COS(Y)−1)
−Y^3LOG(2−2COS(Y))−4LI_3(%E%IY)+4%I$Y LI_3$(%E%IY)+4LI_3(−%E%IY)
−4%I$Y LI_3$(−%E%IY))/2=$\frac{X^3+3\%C}{3}$

図-2　微分方程式 $dy/dx=x^2\sin y/(1+y^2)$ の MACSYMA による解

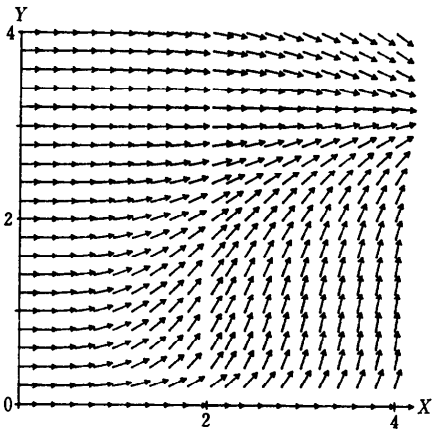


図-3 微分方程式 $dy/dx=x^2\sin y/(1+y^2)$ の勾配場

り，大域的な（$x\to\infty$ の）解の性質の見通しがつけにくい．一方最も簡単な数値手法として，(x, y) 平面に，この微分方程式の勾配場 (gradient field) を描いてみよう．すなわち各点 (x, y) に対して，それを始点に長さ1，角度 $\theta=\arctan(dy/dx)$ の矢印を対応させよう．すると図-3 がえられ，これから $x\to\infty$ のとき $y=\pi$ の定常解に漸近することが理解できる．

すなわち，この場合数式処理と数値処理の結果を照合し，これらを総合して解の状況を理解できるわけである．

実際の問題解決過程は，上記(1)，(2)，(3)が組み合わさり，ひとつの流れをなすものであろう（図-1 (d)）．そこでは，何重にも界面が現れるであろうが，その分析はひとつひとつについて行い，しかるのち全体の解釈をすることが最も妥当であろう．

その際，注意を払わねばならない点として，前にも述べた「LISP 文化」と「FORTRAN 文化」の相違から，数式処理と数値処理との間の結果の受け渡しが，あまりスムーズではないことがある．たとえば，FORTRAN における右辺から左辺への代入の記号 "=" は，ふつうの数式処理言語では，論理関係記号となっているし，また，数式処理でちょっとした計算ののちえられた数式は，しばしば膨大な展開式となり，それをそのまま FORTRAN 記法に書き出すと，コンパイラの制限事項である，継続行の上限にひっかかってしまう．このような側面も，両者の界面として考慮しておかねばならない．

個々の界面として，例もあり，考察もされているものとして，次のものがあげられる．

天体力学におけるポアソン級数の展開と計算
数理物理学における摂動法
常微分方程式の初期値問題に対するテイラ級数法
数式積分と数値積分
有限要素法計算
高次有限差分法公式の生成
数式微分とヤコビアン行列の計算

以下これらのうちのいくつかに触れながら，数式処理と数値処理の界面について，具体的に考察をすすめよう．

3. 常微分方程式の数値アルゴリズムと数式処理

3.1 数値アルゴリズムの意義

数式処理は，数値解析の研究にも大いに役立つ．これは，数学一般の研究に役立つのと似ているが，問題解決過程の分析でいえば，数値手法そのものの解析の段階での，数式処理と数値処理の界面である．

ここで，常微分方程式の数値解法と解析的解法との関係について，一言ふれておこう．数式処理システムが強力となり，常微分方程式の数学的理論に沿って，解析解すなわち閉じた形式での解関数を与えることができるようになれば，数値解法は不要ではないかと誤解される向きもあるかもしれない．しかし事実はそうではない．

第一に，解析解が求められない微分方程式は多数あって，その場合は数式だけの取り扱いは当然限界にゆきあたる．第二に，たとえ解析解を与える方法がよく知られているクラス，ことに2階線型微分方程式でも，事態は簡単ではない．本特集の渡辺隼郎の解説が述べておられる，フックス型微分方程式は，アルゴリズムによるアプローチが成功して，数式処理システムのなかに常微分方程式解法が組み込まれる典型的な成功例であるが，そのような場合でも，解析解のみでは応用上十分ではない．

たとえば，Woods-Saxon 型ポテンシャル・モデル $V(r)$ のもとで，中性子散乱のS波のエネルギ準位 E を求めるため，動径方向シュレディンガ方程式

$$\frac{d^2u}{dr^2}+[E-V(r)]u=0,\quad r>0 \qquad (1)$$

を境界条件

$$u(0)=0,\quad u(\infty)=\text{有限} \qquad (2)$$

の下で解くことを考えよう．この固有値問題に対して，独立変数の変換を行うと，微分方程式は，3個の確定特異点（$z=0, 1, \infty$）をもつ超幾何微分方程式に

帰着できる[3].

それは，いずれも超幾何関数として表せる2個の線型独立な解をもつ．結局もとの問題の解も，超幾何関数を含む線型独立な解の線型結合として表現され，境界条件 $u(0)=0$ から，線型結合の係数の比が決まり，$u(\infty)$ が有限になる条件がエネルギ準位を決める．ただし，それには超幾何関数を含む方程式(超越方程式)を解かねばならない．

数式処理としては，超幾何微分方程式に帰着できる微分方程式を与えて，その一般解を求めることは，不可能ではなかろう．パターンマッチングによっても，アルゴリズムによっても帰着可能であることを識別することは，困難ではないだろう．しかし，境界条件から固有値を決めるとなると，もともと超越方程式に帰着されるものであるから，一般解はほとんど不可能であろう．また，パラメータの特殊な値に対して，その可能性を探るにしても，超幾何関数に対して多数導かれている公式を駆使する必要があり，自動化は困難である．

一方数値的手法では，固有値 E の値を仮定し，$u(0)=0$, $u'(0)=b$ (b はパラメータ) のもとで(1)式を解き，十分大きな r に対して $u(r)$ が有限になる，あるいは $u'(r)$ が0に近くなるように b と E の値を修正していくことが可能であり，しかも実行しやすい．この問題に対しては，数値計算の方が簡便かつ有用なのである．ただ，解析解を求めることで，E と $u'(0)$ の関係が導けると，数値手法に大変役立つ．

要は，数式処理が十分発達しても，それのみで万能というわけにはいかない，やはり数値処理との界面が重要だということである．

3.2 数値解析研究と数式処理

さて，常微分方程式の初期値問題

$$\begin{cases} dy/dx=f(x,y), & a<x<b \\ y(a)=y_0 \end{cases} \qquad (3)$$

の数値解法として，最も有力なクラスは離散変数法 (discrete variable methods) であろう．それは，独立変数の区間 $[a,b]$ をステップ幅 h で等分割し，その離散的な点 x_j (ステップ点)

$$x_j=a+jh \ (j=0,1,2,\cdots,n), \quad h=(b-a)/n$$

における解の値 $y(x_j)$ を近似する y_j を計算するアルゴリズムをいう．詳しくは拙著[17]に譲るとして，離散変数法の解析における数式処理とのかかわりについて，二つの例を述べる．

線型多段階法というのは，y_i から y_{i+N-1} までの値が求まっているとして，

$$\alpha_0 y_i+\alpha_1 y_{i+1}+\cdots+\alpha_{N-1}y_{i+N-1}+\alpha_N y_{i+N} = h(\beta_0 f_i+\beta_1 f_{i+1}+\cdots+\beta_{N-1}f_{i+N-1}+\beta_N f_{i+N}) \qquad (4)$$

によって，y_{i+N} を決めようとするものである．ここで f_j は $f(x_j,y_j)$ を意味する．その際，パラメータ α_0〜α_N, β_0〜β_N を，(x_i,y_i)〜(x_{i+N},y_{i+N}) を通る，なるべく高い次数の多項式解に対して正確にあてはまるようにする．

こうした観点で最もよく使われるのは，アダムス型と呼ばれる族で，微分方程式(3)の両辺を x_i から x_{i+1} まで積分してえられる

$$y(x_{i+1})-y(x_i)=\int_{x_i}^{x_{i+1}} f(x,y(x))dx$$

に対して，右辺の被積分関数を (x_{i-r},f_{i-r}), $(x_{i-r+1},f_{i-r+1}),\cdots,(x_i,f_i)$ を通る補間多項式で近似するものである．補間多項式として，ニュートン型を用いることにすると，後退差分演算子 ∇ を

$$\nabla f_i=f_i-f_{i-1},\ \nabla^2 f_i=\nabla(\nabla f_i),\cdots,\nabla^r f_i=(\nabla^{r-1}f_i)$$

のように定義して，$(r+1)$ 段階公式

$$y_{i+1}-y_i=h\sum_{j=0}^{r}\gamma_j\nabla^j f_i \qquad (5)$$

がえられる．ただし γ_j は有理数で

$$\gamma_j=\int_0^1 \frac{\theta(\theta+1)\cdots(\theta+j-1)}{j!}d\theta \qquad (6)$$

なる定積分で定義される，(5)式を通常の形式(4)式に直すには，二項係数を乗じながら γ_j の和を作ればよいのだが，公式のプログラム・コードの中には，γ_j のまま蓄えておくのがよい．その主な理由は，現実のコードは，可変ステップ・可変次数法 (variable-step, variable-order method, VSVO 法) であり，コード内で適応的に次数を上下させ (r を上下させ) るが，そのとき γ_j は不変であっても，公式のパラメータ (β_0〜β_{N-1}) は変化してしまうので，γ_j から β_j への変換は，必要なときに機械的に行った方がよいからである．

γ_j を(6)式によって求めることは，まったく機械的であるが，正確な有理数として求めておくことが必要である．さらに，アダムス型であるが，(x_{i-r},f_{i-r}), $\cdots,(x_i,f_i),(x_{i+1},f_{i+1})$ を通るニュートン補間多項式で被積分関数を近似すると，陰的公式 (implicit formula) がえられ，この際も(6)式とよく似た積分値が必要である．VSVO 法では，ステップ幅変更を自動的に行うため，ノルツェック (Nordsieck) 行列という正方行列を用意しておくが，この成分も有理数として

機械的に決めることができる．こうした定数の決定は，ぜひ数式処理であらかじめ実行し，これを数値計算コードに書きこんでおくべきである．

線型多段階法と同様に有力な離散変数法のクラスは，ルンゲ・クッタ法である．これは

$$\begin{cases} y_{i+1}=y_i+h\sum_{j=1}^{p}\mu_j k_j \\ k_1=f(x_i,y_i),\ k_j=f\left(x_i+\alpha_j h,\ y_i+h\sum_{l=1}^{j-1}\beta_{jl}k_l\right) \end{cases} \quad (7)$$

と表される．すなわちステップ点 x_i と x_{i+1} の間に分布する $x_i+\alpha_j h$ において，微分方程式(3)の解の勾配を複数個求め，その重みつき平均によってステップ点 x_{i+1} での近似値 y_{i+1} を決めようというものである．自然数 p を公式の段数 (stage number) という．

ルンゲ・クッタ法の解析で面倒な点は，パラメータ $\alpha_j, \beta_{jl}, \mu_j$ の決定のために，ひどく複雑な連立非線型代数方程式が現れることである．微分方程式の解は十分滑らかと仮定して，(7)式の k_j を h のべき級数に展開する．これは2変数関数のテイラ級数展開によるが，$k_1, \cdots, k_{j-1}$ の h のべき級数展開式が必要になり，その結果 f の x と y に関する高階偏微分係数が複雑に掛け合わされた，べき級数の係数式がえられる．これを(7)式の第一式に代入し，$y(x_i+h)$ の h に関するべき級数展開と比較し，各係数を等しいとおき，さらに任意の f に対して恒等的に成り立つよう，高階偏微分係数の係数どうしも等しいとおくと，パラメータに関する連立代数方程式がえられる．これをパラメータの決定方程式系という．

以上の手続は，p が少し大となり，かつ要求される次数も高くなると，到底手計算では及びもつかず，数式処理によらざるをえないが，それでも計算を能率的に行うための工夫がいる．さらに，これら決定方程式系の解を決め，ルンゲ・クッタ公式を導くという仕事になると，数式処理システムと数学的洞察力の両方を駆使することが必要であり，こうしてえられた最近の顕著な結果としては，

(1) 戸田・小野による，実質的に5段5次，6段6次公式の導出[22),28)] (数式処理システムにおける因数分解の機能が，決め手となった.)

(2) J.C. Butcher による，10段公式は7次にしかならず，8次にはなりえない証明[5)]

をあげることができよう．

単に次数を決定するにとどまらず，同じ段数で同じ $k_1 \sim k_p$ を使いながら，重み μ_j を二通りにとることで，次数の異なる公式を併存させる公式（埋め込み型公式）の決定など，ルンゲ・クッタ法の解析において数式処理を活用する局面は広い．

4. 数式処理をとり入れたヤコビアン行列の計算

n 次非線型写像 $f(x)$ のヤコビアン行列 $\partial f/\partial x$，あるいはその勾配 grad f を求める必要性は，きわめて広範囲に存在している．

その第一は，連立非線型方程式

$$f(x)=0 \quad (8)$$

をニュートン反復法

$$x_{k+1}=x_k-[\partial f/\partial x(x_k)]^{-1}f(x_k) \quad (9)$$

で解く場合である．これを幾何学的に解釈すれば，f の成分 f^i ごとに曲面 $y=f^i(x)$ を考え，x_k におけるその接超平面を決め，これら n 枚の超平面の交点として，次の近似値 x_{k+1} を求めるものである．したがって，各成分の勾配 grad f^i を求めることと，ヤコビアン行列を求めることとは等価になっている．

このように考えれば，勾配ベクトルをなんらかの意味で用いる最適化法も，今考えているカテゴリにあると言ってよい．

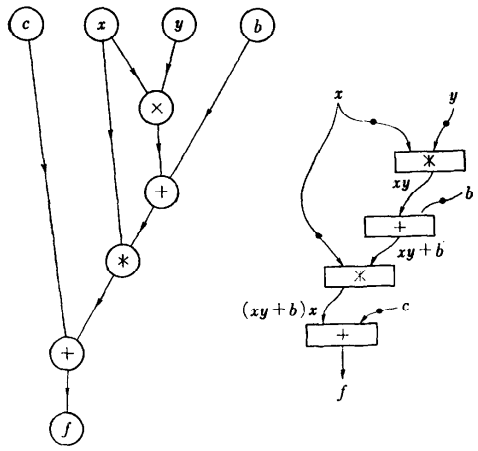
ところが現在までのところ，数値解法のコードのなかで解析的なヤコビアン行列を要求するものは，有力ではない．ひとつの理由は，数式処理が未発達で数値処理との界面が不十分であったことにあるが，もうひとつの理由は，まともに解析的なヤコビアン行列を求めれば，計算量がふえることにあった．そのため，解析的微分に代って数値微分が用いられてきた．これはヤコビアン行列の第 j 列ベクトルの代りに

$$\{f(x+\delta e_j)-f(x)\}/\delta \quad (e_j \text{ は第 } j \text{ 単位ベクトル})$$

を用いる．こうすると，ヤコビアン行列1個につき f の計算が $(n+1)$ 回必要である．

最近になって，解析的なヤコビアン行列を高速に計算する方法が提案され，インプリメントされるに至ってきた[2),12),23)]，それは，直接に数式処理システムの上ではないが，基本的な考え方において数式処理に近く，また数式処理と数値処理の界面の将来にとって重要と思われる．

この方法は，数式のデータ構造として，L.V. Kantorovič の古い論文[13)]に基づく計算グラフ (computational graph) を採用することから出発する．詳しくは，伊理・土谷・星の論文[12)]を参照されたいが，この



(a) 計算グラフ (b) データフローグラフ

図-4 $f(x, y)=(xy+b)x+c$ の計算グラフとデータフローグラフ

ようにすると，導入された中間変数を含め，計算グラフの各ノードを結ぶ枝に要素的 (elementary) 偏導関数が対応させられ，最終的な関数の各変数に対する偏導関数は，中間変数に関する連鎖規則によって合成できる(図-4(a))．重要なことは，関数の計算 (evaluation) のためには，グラフのノードの数に等しい回数の基本演算が必要であるが，そのすべての偏導関数を求めるには，関数自身の計算の手間の，高々定数倍（入力変数の数に依らない）程度で済むことが示されていることである．

さらに，まだあまり明確に指摘されていないが，各偏導関数の計算は，計算グラフが与えられたもとで，並列処理できることも重要である．実際計算グラフはデータフローグラフと同一視できる(図-4(b))から，上記方法による関数値および偏導関数値計算は，データフロー計算機[8),25)]に向いており，数式処理と数値処理の新しい界面を開くものと期待できる．

5. 数式処理によるテイラ級数法

前章の主題とも関係するが，物理学では，パラメータ ε を含む連立非線型方程式

$$f(x, \varepsilon)=0 \tag{10}$$

の解を，$\varepsilon=0$ のまわりで ε のべき級数に展開する摂動法がよく用いられる．$f(x, 0)=0$ の解を $x_0=(x_0^1, x_0^2, \cdots, x_0^n)$ とするとき，(10)式の解の第 i 成分 x^i を

$$x^i=\sum_{j=0}^{\infty} a_{ij}\varepsilon^j \tag{11}$$

と展開する．ここで $a_{i0}=x_0^i$ である．摂動計算は，展開式(11)を N 次で打ち切った近似式を(10)式へ代入し，ε のべきに展開して整理するのであるが，これもニュートン反復によるのがよい．

いま，ε のべき級数である $\varphi(\varepsilon)$, $\psi(\varepsilon)$ が，ε^N を法として合同であることを

$$\varphi(\varepsilon)\equiv\psi(\varepsilon) \mod \varepsilon^N$$

と表すこととする．すなわち，φ と ψ のべき級数の係数は，$(N-1)$ 次まで一致することを意味する．そこで，(10)式の $(N-1)$ 次までの近似式 x_k，つまり

$$f(x_k, \varepsilon)\equiv 0 \mod \varepsilon^N$$

から，

$$x_{k+1}\equiv x_k-[\partial f/\partial x(x_k)]^{-1}f(x_k, \varepsilon) \mod \varepsilon^{2N} \tag{12}$$

によって次の x_{k+1} を決めれば

$$f(x_{k+1}, \varepsilon)\equiv 0 \mod \varepsilon^{2N}$$

となることが示された[15),30)]．すなわち，x_{k+1} は $(2N-1)$ 次の近似式となるのである．

ニュートン反復による摂動計算を，数値処理として実行することはもちろん可能であるが，任意の初期値 x_0 に対して(11)式をえたいときには，数式処理によるべきだろう．しかし，これをちょっと実行してみると，計算の手間が爆発的に増して，とても実用にはならなかった．しかし最近，f が代数方程式系の場合に限るが，新しい算法がえられた[18),19)]，それは $f(x_0, 0)=0$ を x_0 についての代数的制約条件と考え，x_0 の各成分の有理関数となる a_{ij} を，反復過程(12)式でその条件を用いた簡約化を行うものである．これが可能になったのは，イデアルの Gröbner 基底の構成法[24)]が数式処理に組み込まれるようになったからである．

こうしてえられたべき級数解は，数値処理に渡して，近似的な x_0 に対して近似解を求めることもできるし，またそれ自身を注意深く考察することで，新たな知見がえられるだろう．

同様の考え方は，微分方程式

$$f(x, y, dy/dx)=0 \tag{13}$$

にも適用できる．解 $y(x)$ が初期値 (x_0, y_0) において解析的であると仮定すると

$$y(x)=\sum_{j=0}^{\infty} c_j(x-x_0)^j, c_0=y_0 \tag{14}$$

というテイラ展開が可能である．未定係数 c_j は(14)式を(13)式へ代入して，$(x-x_0)$ のべきに整理して決めるが，やはり有限和近似から出発して，ニュートン

反復でべきの次数をふやしていくのがよい．この方面の研究はまだ少なく[9),10)]，また Gröbner 基底の応用には手が着いていないが，発展させれば，数式・数値アルゴリズムとして特徴あるものがえられるであろう．特に，非正規形の微分方程式にも応用可能であることは魅力的である．

6. 特殊関数

応用数学でよく用いられる特殊関数は，次のような非初等的関数の総称である．

（i） Γ 関数と，それから誘導される諸関数．

（ii） フレネル積分，誤差関数，対数積分など，初等関数として表せない，初等関数の不定積分である関数．

（iii） 楕円関数．

（iv） ラプラス方程式を，各種の曲線座標で変数分離してえられる2階線型常微分方程式の解となる関数．

それぞれ由来も定義も異なるので，これらを一緒に考えることは本来無理なのであるが，習慣上まとめて扱っている．特殊関数の計算ルーチンは，計算センターのライブラリのなかで，かなりの比重を占めている．したがって，数式処理でも取り扱うべき対象の候補として有力であるが，現状ではまだ不十分である．

ひとつには，特殊関数を活用する場合，それについてえられているさまざまな公式——パラメータ・変数の変換，異なる特殊関数どうしの関係，漸近展開など——を駆使し，数式の簡約化を行って，最終的な結果をうるのがふつうのやり方だが，その過程の分析が不十分で，アルゴリズムとして書き下せていないことがある．膨大なこれら公式を，数学データベースとして数式処理システムに組み込むことが，（それ自身容易ではないが）成功したとしても，それだけでは宝の持ち腐れとなる可能性がある．

したがって，さまざまな公式の数学的な整理，すなわち相互の代数的な関係を明らかにし，この方向で数式処理にとり入れることが有望であろう．こうした方向で，横尾・小枝・片桐[29)]は誤差関数とその周辺関数に対して，ある程度の成功をおさめている．

誤差関数

$$\operatorname{erf} x = a\int_0^x \exp(-t^2)dt \tag{15}$$

は，上に述べたように特殊関数のひとつであるが，これを含む一般的な関数

$$y(c,l,m,n;x) = c\exp(-lx^2)x^m(\operatorname{erf} x)^n \tag{16}$$

を定義する．ただし c は定数，l は整数，m,n は自然数である．微分に関する漸化関係

$$y'(c,l,m,n;x) = y(-2cl,l,m+1,n;x) + y(cm,l,m-1,n;x) + y(cn,l+1,m,n-1;x) \tag{17}$$

を手がかりに，関数 $y(c,l,m,n;x)$ とその不定積分のなす集合を特徴づけ，それに入る関数については，誤差関数を含む表現式を与える還元則（reduction rule）を考察し，これを微分方程式の境界値問題に応用して，数値処理との結合をはかっている．

J. Calmet[7)] らも，漸化式を手がかりに，数式処理による直交多項式の識別を行っている．

7. 有限差分法・有限要素法コードの自動生成

数式処理システムによって，偏微分方程式の境界値問題を解くための有限差分法あるいは有限要素法コードを自動生成し，数値処理システムで数値計算をするという過程は，数式処理と数値処理の界面として最も早く使われ，最も広範に実用化しているものの一つである．

有限差分法についていえば，たとえばいくつかの数学公式集[1)]には，かなり高い次数の差分公式がのせられている．しかしさらに高い次数の差分公式を用いる場合，あるいは非線型性を含む偏微分作用素を差分化する場合，さらには線型偏微分作用素であっても，空間領域が不規則であるので，正規形への変換を行い，そこでの差分化をもとの領域に逆変換する場合など，数式処理が用いられる．もちろん差分化のアルゴリズムはあらかじめ決められているものとし，長大で誤りやすい，数値処理のためのプログラミングの主要部を自動化しようとするものである．この点では，すでにいくつかの成功例も報告されている[14),26),27)]．

有限要素法の場合は，静的問題でも動的問題でも，剛性行列（stiffness matrix）と質量行列（mass matrix）を決めることが肝心な点である．どちらも，区分的多項式とその導関数の積の，有限領域での積分によるから，数式処理を応用することができる．ただ，有限差分法の場合もそうであるが，2章で述べたように，数式処理システムで生成した数値計算プログラム（たとえば FORTRAN プログラム）は，数式表現が長く複雑になりすぎる傾向があり，いかに簡約化して表現するかが課題となっている．

有限差分法・有限要素法については，数式処理システムから数値処理言語（多くは FORTRAN）のプロ

グラム形式が出力ができることが不可欠な道具である．これが狭い意味での両者の界面といえよう．

8. むすび——今後の課題

以上とりあげたテーマのほかに，たとえば数値積分と数式積分との界面とか，区間演算 (interval arithmetic) を数式処理によって解析し直し，区間解析 (interval analysis) を，より現実的なものにしていく試みであるとか，最近さまざまなテーマが展開されるに至っている．この小論でふれることのできなかったテーマについては，参考文献 4), 6), 20) を参照していただきたい．

数式処理と数値処理の界面の発展のためには，両者の側で他を意識した新しいアルゴリズムの開発が強く望まれるところであるが，そのためにも両方の処理システムをスムーズに渡ることのできる計算機環境が切望される．

筆者は，小鹿丈夫・渡部敏とともに REDUCE と FORTRAN の両方を使いながら，連立非線型方程式を解く NAES というパッケージの開発にあたり，このような処理を混合処理 (hybrid manipulation) と名づけることとした[21]．今のところ，混合処理は OS に依存しながらファイルのやりとりを通じて，REDUCE と FORTRAN の間で結果の受け渡しをする形式をとっているが，その経験を通じて混合処理のできる新しいシステムの必要を痛感する．

すなわち，問題解決過程のある局面 (phase) では数値処理を実行し，別の局面では数式処理を実行し，両者の間でデータの受け渡しがスムーズにでき，かつできるだけ両者が高速で演算される新しいシステムである．数値処理言語の多くはコンパイラ型であるのに対して，数式処理言語はほとんどインタプリタ型で，しかも LISP に基礎を置くものが多い．また FORTRAN には，過去の膨大なライブラリの蓄積があるなど，かなり根本的な相違点があるが，それを克服した新たなシステムが望まれる．MACSYMA や REDUCE には数値計算 (numerical evaluation) 機能がついているが，数値処理言語の実行に比べれば格段に遅く，しかも内容がまだ貧しい．しかし，とにかくその機能があるということは，混合処理システムの実現に希望を抱かせてくれる．

「大量の数値計算が容易に行われるようになればなるほど，非数値的解析を通じて，現象を支配している本質を見出す努力が一層重要となってくる[27]」と指摘されている．かつて R. W. Hamming は，「計算 (computing) の目的は，数ではなく洞察 (insight) にある」と彼のテキスト[11]の冒頭に記したが，これからは「数や式ではなく洞察にある」といいかえるべき時代に入っていくであろう．それを期待しつつ，拙い解説を終ることとする．

最後に，小論の執筆にあたって有益な刺激を与えて下さった，科研費総合A「数式処理の学際的応用への総合的研究」研究班 (代表者 後藤英一氏) の方々に，深甚なる謝意を表したい．

参 考 文 献

1) Abramowitz, M. and Stegun, I. A.(ed.): Handbook of Mathematical Functions, Dover, New York pp. 883-885 (1965).
2) Арайс, Е. А. и Яковлев, Н. Е.: Автоматизация Аналитичесних Вычислений В Научных Исследованиях, «Наука» Сибирское отделение, Новосибирск (1985). (Arais, E. A. and Yakovlev, N. E.: Automatization of Analytic Computation in Scientific Research, "Nauka" Siberian Branch, Novosibirsk.)
3) Bencze, Gy.: Analytic Solution of the Schrödinger Equation with Optical Model Potential for S-wave Neutrons, Comment. Physico-Mathematicae, Vol. 31, N◦ 9, pp. 1-4 (1966).
4) Brown, W. S. and Hearn, A. C.: Applications of Symbolic Algebraic Computation, Computer Physics Comm., Vol. 17, pp. 203-215 (1979).
5) Butcher, J. C.: The Non-existence of the Ten Stage Eighth Order Explicit Runge-Kutta Methods, BIT, Vol. 25, pp. 521-540 (1985).
6) Calmet, J. and Van Hulzen, J. A.: Computer Algebra Applications, Computing, Suppl. 4, pp. 245-258 (1982).
7) Calmet, J. and Cohen, I.: Symbolic Manipulation to Recurrence Relations, An Approach to the Manipulation of Special Functions, Symbolic and Algebraic Computation by Computers, World Scientific Publ., Singapore, pp. 55-65 (1985).
8) Chudìk, J.: Data Flow Computer Architecture, Algorithm, Software and Hardware of Parallel Computers, Springer, Berlin, pp. 323-358 (1984).
9) Geddes, K. O.: Convergence Behaviour of the Newton Iteration for First Order Differential Equations, LN in Computer Sci,. Vol. 72, FUROSAM 79, pp. 74-87 (1979).
10) 藤瀬哲朗：連立1階常微分方程式の形式級数解の近似計算法，数理解析研究所講究録 486「数式処理と数学研究への応用」, pp. 49-66 (1983).
11) Hamming, R. W.: Numerical Methods for

Scientists and Engineers, McGrow-Hill, New York (1962).
12) 伊理正夫，土谷 隆，星 守：偏導関数計算と丸め誤差推定の自動化の大規模非線形方程式系への応用，情報処理，Vol. 26, No. 11, pp. 1411-1420 (Nov. 1985).
13) Канторович, Л. В.: Об Одной Математической Символке, Удобной при Проведении Вычислений на Машинах, ДАН СССР, Том 113, No. 4, 738-741 (1957). (Kantorovič, L. V.: On a Mathematical Symbolism Convenient to Machine Computation, Dok. Akad. Nauk a USSR, Vol. 113, pp. 738-741).
14) Keller, H. B. and Pereyra, V.: Symbolic Generation of Finite Difference Formulas, Math. Comp., Vol. 32, No. 144, pp. 995-971 (1978).
15) Lipson, J. D.: Newton's Method: A Great Algebraic Algorithm, Proc. 1976 ACM SYMSAC, pp. 260-270 (1976).
16) 三井斌友：数式処理のすすめ(1)，(2)，京都大学大型計算機センター広報，Vol. 16, pp. 331-343 (1983)，Vol. 17, pp. 19-28 (1984).
17) 三井斌友：数値解析入門一常微分方程式を中心に一，朝倉書店，東京 (1985).
18) Moritsugu, S., Inada, N. and Goto, E.: Symbolic Newton Iteration and its Application, Symbolic and Algebraic Computation by Computers, World Scientific Publ., Singapore, pp. 105-107 (1985).
19) 森継修一：記号的ニュートン反復法の拡張，数式処理通信，Vol. 3, No. 3, pp. 14-21 (1985).
20) Ng, E. W.: Symbolic-Numeric Interface: A Review, LN in Computer Sci., Vol. 72, EUROSAM 79, pp. 330-345 (1979).
21) 小鹿丈夫，渡部 敏，三井斌友：連立非線形代数方程式解法用パッケージ NAES とニュートン法，数理科学，No. 218, pp. 23-30 (1981).
22) 小野令美，戸田英雄：6個の関数計算による実質的6次の Runge-Kutta 法，情報処理学会論文誌，Vol. 23, No. 6, pp. 599-607 (1982).
23) Rall, L. B.: Automatic Differentiation: Technique and Applications, LN in Computer Sci., Vol. 120, Springer (1981).
24) 佐々木建昭，古川昭夫：コンピュータ環論，情報処理，本特集号 (1986).
25) 島田俊夫：数値計算向きデータフロー計算機，情報処理，Vol. 26, No. 7, pp. 780-796 (July 1985).
26) Steinberg, S. and Roache, P.: Using VAXIMA to Write FORTRAN Code, Application of Computer Algebra, Kluwer Academic Publ. pp. 1-22 (1985).
27) 竹田辰興，常松俊秀：核融合計算への応用，数理科学，No. 242, 特集「数式処理」，pp. 29-38.
28) 戸田英雄，小野令美：5個の関数計算による実質的に5次の Runge-Kutta，情報処理学会論文誌，Vol. 22, No. 2, pp. 85-98 (1981).
29) 横尾英俊，小枝祐司，片桐理和：誤差関数とその周辺関数の数式処理と応用，情報処理学会論文誌，Vol. 25, No. 5, pp. 797-803 (1984).
30) Yun, D. Y. Y.: Algebraic Algorithm Using p-adic Constructions, Proc. 1976 ACM SYMSAC, pp. 248-259 (1976).

(昭和60年12月9日受付)