参 考 文 献

1) Iri, M. (1984). Simultaneous computation of functions, partial derivatives and estimates of rounding errors — complexity and practicality --, *Jpn. J. Appl. Math.*, **1**, 223-252.
2) 伊理正夫・小野令美・戸田英雄（1986）. 合成関数の高速微分法とその導関数を含む Runge-Kutta 系の常微分方程式数値解法公式への応用，情報処理学会論文誌，**27**，223-252.
3) 小野令美（1986）. Runge-Kutta 系の 6 段 6 次極限公式及び 6 段で数値的に 6 次の公式，情報処理学会論文誌，**27**，掲載予定.
4) 戸田英雄（1980）. Runge-Kutta 系のある極限公式の打切誤差について，情報処理学会論文誌，**21**，285-296.

微分方程式を数値的に解くための数式の前処理について

千葉大学工学部 吉 田 利 信・中 野 泰 男・山 下 稔

物理的な問題を数値的に解く場合，数学的な微分方程式の記述から，数値的に解を求めるための具体的な $y'=f(x, y)$ の形の記述を得る必要がある．この作業は大きな問題に対してはかなり労力が必要である．そこで，この f の導出が容易に記述でき，また，f の効率のよい計算過程が得られるような数式処理システムが望まれる[2]．

本発表では，微分方程式を解くための具体的な数式を導出したり，それらの数式に対して微分方程式を解く算法で必要となる前処理を施したりするのに望ましい数式処理システムについて，REDUCE と我々が試作した数式処理システム[3] とを用いて検討した．

まず，2 重振子の運動方程式を求めるという簡単な力学の問題を記述し，運動方程式を導出し，その大きさを比較した．REDUCE の出力は FORTRAN 形式で 1022 行となるが，試作システムでは，演算総数が 542 個，行数が 180 行とかなり効率のよいプログラムが得られる．しかし，これも手で最適化することによって，演算総数が 120 個，行数が 45 行のプログラムを得ることができる．この例から，REDUCE はこのような数値計算のための数式の導出には不向きであること，また，我々の試作システムも，さらに効率のよい数式を導出できる余地が存在していることから十分なものとはいえないことがわかった．また，これらの数式処理システムへの入力（プログラム）の記述が大変煩わしいことから，計算過程を容易に記述することができる言語インターフェイスが望まれる．

次に，微分方程式 $y'=f(x, y)$ の数式による具体的な記述が得られているとき，微分方程式を解く Runge-Kutta 法，Taylor 法における計算過程の導出について比較した．その結果，次の結果が得られた．

(1) 微分方程式を数値的に解くために広く Runge-Kutta(-Gill)法が用いられ，数値計算用副プログラムライブラリ中のサブルーチンが使われている．このようなライブラリでは，連立型の微分方程式に対して，関数の計算を関数副プログラムで定義し，複数の関数を別個に計算している場合が多いようである．しかし，関数の計算をサブルーチン副プログラムで定義し，複数の関数を同時に計算し重複した計算をしないことが望ましい．計算グラフを用いた我々の試作システムは，複数の関数に現われる重複した計算を見つけ出し，効率的な計算過程を作り出すことが不完全ではあるが可能である．

(2) 偏導関数を用いる Runge-Kutta 2 段 4 次法[1] や Taylor 法において，我々の試作システムは効率のよい偏導関数の計算過程を導出することができる．また，6 個の例題に対して，5 次

のTaylor法は，Range-Kutta 4段4次法と同程度から1/2の演算時間で同程度の精度を得ることができた．

参考文献

1) 伊理正夫，小野令美，戸田英雄（1986）．合成関数の高速微分法とその導関数を含むRunge-Kutta系の常微分方程式数値解法公式への応用，情報処理学会論文誌，**27**，389-396．
2) 三井斌友（1986）．数式処理と数値処理との界面，情報処理，**27**，422-430．
3) 山下 稔，吉田利信（1986）．計算グラフを用いた数値計算のための数式処理システムの設計，研究集会「グラフ理論の数値計算への応用」予稿．

計算グラフを用いた数値計算のための数式処理システムの設計

千葉大学工学部 山下 稔・吉田利信

現在知られている数式処理システムとしてREDUCE, MACSYMAなどがある．これらのシステムは，入力された数式に対して微分などの処理を行い，その結果を数式として出力する．しかし，これらのシステムでは，数式を入力変数まで展開してしまうので出力される数式が膨大になる事がよく起こる．また，その出力には，共通な因子が含まれている事も多い．したがって，計算が効率よく行えるような数式を出力する数値計算のための数式処理システムの実現が望まれる．

最近，計算過程を表す計算グラフを用いた偏導関数の自動計算法が提案され[2),4)]，関数自身を計算する手間の定数倍の手間で，正確な偏導関数の値が求まる方法が示された．この方法は，計算グラフで表される計算過程に対して，偏導関数は“入力変数と出力変数とを結ぶ経路上に現れる要素的偏導関数の積を，そのようなすべての経路について足し合わせる”ことにより得られるという考えに基づいている．また，偏導関数の自動計算法を用いたプリプロセッサ方式のシステムが作成されている[3),4),5)]．

そこで，数値計算のための数式処理システムとして，計算グラフを用いたシステムを試作した．この構造は，(1) 数式を入力してその計算過程を計算グラフとしてシステム内部に蓄える部分，(2) その部分グラフにより表される複数の関数に対してそれらの偏導関数を求める過程を計算グラフに追加する部分，(3) 計算グラフから複数の関数の計算手順を出力する部分，及び (4) 不要になった部分グラフを削除する部分からなり，UTILISP [1)] で記述されている．(1) (2) (3) の部分の特徴は，

(1) 数式を，単項または，2項の基本演算に分解して節点を作成する．その際，基本演算に対応する節点が既に作られていれば，新たに節点を作り出さずに，その節点を対応させる．したがって，同じ基本演算については，1つの節点が対応することになる．また，$V+0$, $V*1$, $V-V$, V/V の基本演算についてはそれぞれ V, V, 0, 1 が対応するので節点を作り出さず，$U-(-V)$, $(-U)-(-V)$ に対してはそれぞれ $U+V$, $V-U$ を表す節点を作り出すなどの簡略化を行なう．これらのことより，出力されたプログラムを実行する時の計算効率はよくなる．しかし，結合，分配の法則に基づいた等価な節点でも，それらを異なる2つの節点として作り出すという問題がある．例えば，$(X1*X2)*X3$ と $X1*(X2*X3)$ は数学的に等価な節点であるのにそれぞれ異なる節点として作り出される．

(2) この試作システムにおいて，偏導関数の自動計算法は“システムに蓄えられている計算