3X-3

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

吉田　利信　　山下　稔
千葉大学工学部

1．はじめに

数理的な問題を解くことは、問題を定式化し、その解の満たす方程式あるいはなんらかの関係を表す式を導出することに始まる。理論的な問題では、これら方程式あるいは関係を表す式自体が得られればよい。このような目的にはREDUCEなどの数式処理システムはかなり強力な道具である。しかし、数値的な解を求める必要がある問題では、さらに数値計算のための計算式を導出し、それを用いて数値計算をしなければならない。このような場合、REDUCEなどを用いて導出した式は手計算で求めた式に比べて一般には非常に効率が悪い。このように数式処理と数値計算の間の距離は大きく、これらの間の界面が議論されるようになって来た［三井 1986］。

このような界面の一つとして偏導関数の計算がある。数値計算のための種々の算法には関数とその導関数の値を必要とするものが多い。これらに対しては、数式としての導関数を得にくいことから、数値微分法を利用したり間接的に導関数値を利用する算法が工夫されている。最近、数値計算の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［Rall 1981］,［Iri 1984］,［伊理他 1985］、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値を求める方法が示された。また、この方法を用いて得た偏導関数値を利用する算法も開発されている［伊理他 1986］。この偏導関数の自動計算法を利用するシステムとして、関数をFORTRAN の式あるいはプログラムで入力するプリプロセッサ方式のシステムが作成されている［Rall 1981］,［岩田 1984］。

本文では、数式処理と数値計算の界面を考えるという立場から、“計算グラフとして表される数値計算過程の操作”という形で数式を扱う数式処理システムを試作したので報告する。このシステムにおいて、偏導関数の自動計算法は計算グラフのある拡張操作として実現される。

2．試作システムの構造

このシステムは、数式を入力しその数式を計算する過程を計算グラフとしてシステム内部に作成する部分と、その計算グラフの部分グラフで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計算過程を計算グラフに追加する部分、部分グラフで表される複数の関数の計算手順を出力する部分、および、不用になった部分グラフを削除する部分からなり、これらはLISPの関数で記述されLisp09とUTILISP 上で稼働している。

2．1　計算グラフの内部表現

計算グラフの各節点はLISPのアトムで表現され、その性質リストに演算子、被演算子のリスト、その節点を被演算子とする節点のリストが記述される。たとえば、F=A*B*(A*B+C) という計算過程は図1に示すように表され、節点V1の性質リストには演算子が＊であること、被演算子がAとB であること、V2とV3でこの節点が使われることが記述される。このように、計算過程は双方向のポインターを持つグラフで表される。

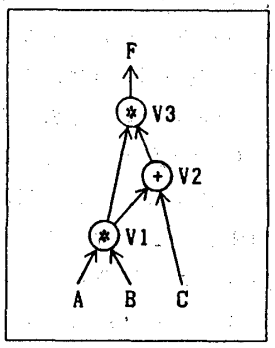

図1 計算グラフの例

2．2　数式の解釈および計算グラフの作成

数式を入力するLISP関数は、与えられた数式を構文解析し、単項または二項の基本演算に分解し、演算順序に従って基本演算ごとに計算グラフの節点を生成していく。

節点を生成する際、V+0, V*1, V-V, V/Vなどは簡約化を行い節点を生成しない。また、U+(-V), (-U)-(-V) に対しては、U-V, V-Uなどを表す節点を生成する。また、可換な演算に対しては、被演算子を辞書の順に並べることにより、たとえば、B*A に対し

Formula Manipulating System for Numerical Computation
Toshinobu YOSHIDA, Minoru YAMASHITA
Chiba University

てA*B を表す節点を生成する。

節点を生成するときに、その基本演算に対応する節点が既に作られているか調べ、存在する場合には新たに節点を生成することをしない。たとえば、F=A*B*(A*B+C) 中のA*B に対して図1のように節点V1のみを生成する。既に節点が作られているかどうかを調べることは、被演算子の節点にある“どの節点でその節点が使われているか”という記述を用いて高速に行われる。しかし、たとえば(A+B)+C とA+(B+C) のように、結合法則により等価であっても、別々の節点を生成する。

2．3　偏導関数計算のための計算グラフの拡張

関数f の変数x に関する偏導関数∂f/∂x は、計算グラフ上でx とf を結ぶ経路上の要素的偏導関数の積をすべての経路について加えることによって求められる。変数 x1,...,xmに関する関数f1,...,fn が計算グラフの部分グラフとして表現されているとき、この方法を用いて、LISP関数DIFF-UP および DIFF-DOWNはそれらの関数の変数に対するヤコビ行列の計算過程を計算グラフに追加する。また、ヤコビ行列にベクトルを掛けてできるベクトルの計算過程は、LISP関数DIFF-UPVによって計算グラフに追加される。

DIFF-UP は、計算グラフ上の半順序に従って、つまり、計算グラフの変数側から関数側に向かって、次のようにヤコビ行列の計算過程を計算グラフに追加する。(1)変数の節点xiに∂xi/∂xi=1 を蓄える。(2)節点viの演算子がf 、被演算子がvk,vl のとき、つまり、vi=f(vk,vl) のとき、要素的偏導関数 ∂vi/∂vk, ∂vi/∂vlを表す節点を生成する。これらの節点と既に得られている節点∂vk/∂xj,∂vl/∂xj から ∂vi/∂xj を表す節点をすべてのj について生成し、節点viに蓄える。この操作を関数側に向って繰返す。(3)関数の節点fiに蓄えられている∂fi/∂xj を表す節点に dfidxjという名前を付け、すべての節点に蓄えられている偏導関数の情報を消去する。以上の手順により計算グラフを拡張すると、その節点の数 L(f,∇f) はもとの計算グラフの節点の数L(f)の1+3m倍以内になる。ただし、m は変数の数である。

DIFF-DOWN は、計算グラフの関数側から変数側に向かって、DIFF-UP と同じようにヤコビ行列の計算過程を計算グラフに追加する。この手順により計算グラフを拡張すると、その節点の数L(f,∇f) はもとの計算グラフの節点の数L(f)の1+3n倍以内になる。ただし、n は関数の数である。

DIFF-UPVは、ヤコビ行列そのものは必要なく、ヤコビ行列にベクトルを掛けてできるベクトル zi=Σ_j (∂fi/∂xj)*yj が必要なときに用いる。手順は、DIFF-UP の(1)において、変数の節点xiに(∂xi/∂xi)*yi=yi を蓄え、(2)と同様に、Σ_j (∂vi/∂xj)*yj を表す節点を生成し、節点viに蓄える。(3)において、関数の節点fiに蓄えられている節点にziという名前を付け、すべての節点に蓄えられている偏導関数の情報を消去する。以上の手順により計算グラフを拡張すると、その節点の数L(f,∇f*y) はもとの計算グラフの節点の数L(f)の4 倍以内になる。

3．まとめ

この試作システムとREDUCEを用いて、3次元空間中の2重振子の運動方程式を導出した。REDUCEではFORTRAN 形式で1022行となったが、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た。しかし、このプログラムはさらに演算総数が 120個、行数が 45 行まで簡約化できる。今後の課題としては、計算グラフの簡約化の操作、計算グラフと計算グラフの合併の操作、計算グラフ操作言語の設計などがある。

参考文献

三井斌友 (1986) ：数式処理と数値処理との界面. 情報処理, Vol. 27, No. 4, pp. 422-430.

L. B. Rall (1981) : Automatic Differentiation: Techniques and Applications. Lecture Notes in Computer Science, Vol. 120, Springer-Verlag, Berlin.

M. Iri (1984) : Simultaneous Computation of Functions, Partial Derivatives and Estimates of Rounding Errors - Complexity and Practicality. Japan Journal of Applied Mathematics, Vol. 1, No. 2, pp. 223-252.

伊理正夫，土谷隆，星守 (1985) ：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.

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, Vol. 27, No. 4, pp. 389-396.

岩田憲和 (1984) ：偏導関数計算の自動化. 東京大学大学院工学系研究科情報工学専門課程修士論文.