数理解析研究所講究録
第551巻 1985年 158-172

Prologによる小型数式処理システムと数値計算

愛媛大工　野田松太郎　（Matu-Tarow NODA）
愛媛大工　戒能　芳弘　（Yoshihiro KAINO）

1．はじめに

最近、数式処理システムへの関心が深まり色々な視点からの研究がなされている。また、REDUCE-3、MACSYMA等の著名なシステムが各種大型計算機上で稼働している。数式処理は、もはや「実験的研究」の段階から「実用」の時代へ突入しているように思われる。これらシステムでは数式を入力することにより、簡素化、微分・積分あるいは微分方程式の求解等を数式のまま遂行する点に特色がある[1]。しかし、良く知られているように工学的分野に出現する大半の問題――その多くは微分方程式によるモデル化により表現される――では、厳密な解析的計算のみで目的が達っせられるものは皆無といっても過言ではない。必然的に近似計算そして数値計算に頼ることになる。この種の要求にこたえるべく、上記各種システムでも技術計算用のFORTRANプログラムの発生

を行うが、十分に利用者の要求を満しているとはいえない。このような数式処理と数値計算の結合の問題に対応するシステムの構築が緊急の課題である。

システム開発のための言語として一般に考えられるのは大半の数式処理システムがよっているLISPである。しかし、利用者との対話環境を考慮すると、LISP的言語はきわめて不適切である。特に高度で複雑な処理を遂行しようとすると利用者がLISPの知識を十分に持つことが要求され各種システムの普及の上での大きな障害となっている。さらに、muMATHあるいはREDUCEの小型版等マイコン・パソコンで使用可能な数式処理システムも開発されているが、上述のような利用者とのインターフェース、移植性等の面に多くの問題をかかえている。

これらに対し、記述言語の変更により既存の数式処理システムの問題点を克服しようとする方向がある。高速な数式処理の遂行のために、言語Cによるシステム SMP[2]、あるいは微分計算に限ってではあるものの数式処理と数値計算の結合に対し効果的なPASCALによるシステム[3]等がある。一方、Bundy等は数式パターンの処理に有効と思われるPrologを用い、簡単な数式の統合あるいは大学入試程度の数学問題の解等について十分な成果を発表している[4]。

そこで、本稿では上記各システムの成果をふまえ、現在の数式処理システムのかかえる諸問題に対処するべき小型数式処理システムの開発について報告する。システム開発のための言語としては、この種システムの基本は数式のパターン・マッチング処理にあること、システムに加える必要のある各種の数式変換用知識（三角関数間の関係その他）、数値計算ライブラリ等のデータベース化を考え、Prolog を採用した。これにより利用の便も大幅に向上することが期待される。また、システムの移植性の良さおよび数値計算との結合を容易にするため、Prolog 処理系自体をCにより新しく作成した。以下、開発した数式処理システムの概容および数値計算との結合を具体例にもとづき述べる。なお、この種システムに必要な言語機能を知るために、作成した Prolog 処理系についても簡単に触れる。

2. Prolog 処理系について

Prologに関する詳しい説明、その応用等は各所で述べられている[5]ので、以下では本稿の一貫性を保つことおよび1.での目的のために、ここで作成した小型Prolog処理系の機能等をまとめる。

i)基本的な機能は、通常の多くの処理系と変化はない。

ii)述語を構成する引数とその表現は次の通りである。

a)変数。記号 $ による（例： $X ）

b)定数。アトムまたは数

c)リスト。（例：（a,b,c）,（$X | $Y））

d)複合項。

iii)組込み述語：下表参照。

operator	symbol	functions
cut	!	freeze certain choices in "Backtracking"
relational	lt le eq ne ge gt	less than less than or equal equal not equal greater than or equal greater than
arithmetic	add sub mul div mod	X+Y=Z add(X,Y,Z) X-Y=Z sub(X,Y,Z) X*Y=Z mul(X,Y,Z) X/Y=Z div(X,Y,Z) XmodY=Z mod(X,Y,Z)
others	read / write atom,car,cdr convert,simpl , etc.	input and output list processing symbolic manipulation

iv)ちり集め機能（ garbage collection ）

v)大域（外部）変数の設定：本来、Prologの変数はホーン節内でのみ有効であるが、これでは目的のシステム構築

には不適切である。記号#により表現する（例：#A）。

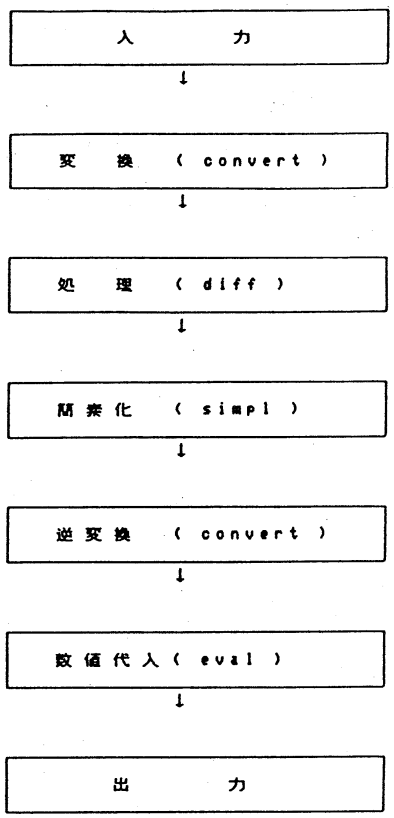
前ページの表より明かなように組込み述語には数式処理に関する述語が多く含まれる。これらを、Cで作成することによりシステムを拡充することは容易である。なお、このProlog処理系はCが稼働する計算機への移植は容易である。現在、DEC・Micro/PDP-11、富士通・FM-11等の小型計算機上で稼働しており、推論速度は約100 lipsである。またPrologが稼働するための必要記憶容量は30 KB程度である。開発の容易さのため単純なコピー方式を採用している。このため推論速度はかなり犠牲にされている。しかし、小型数式処理システム実現のためのある種の「試行システム」作成および検討のためには十分の機能を有している。

3. 数式処理システムとしての概要

数式処理システムを構成する上での要点は数式入力、内部形式（現システムでは演算子の前置形式）への変換、式の簡素化、出力のための式の逆変換、あるいは必要な場合の数値代入、出力等の各段階および対象となる処理ルーチンの効率的な作成にある。たとえば、2次式

$$ax^2 + bx + c$$

の変数 x による微分を数式的に遂行し、かつ $a=1$,



$b=2$, $x=3$ での数値を求めようとすると、上の各処理ルーチンは左図のような制御の流れを構成する。ここで述語 diff は数式微分のための組込み述語である。我々のシステムでは、左図の処理に対して、以下のような Prolog プログラムを書く。使用されている述語の中、convert は入力時の内部形式への変換と、出力時の逆変換の双方に対して用いられる。Prolog 使用の特色の一つに、この種「逆関数」的処理の容易さがある。

```
?- convert(a*x^2+b*x+c,$T1),
   diff($T1,x,$T2),simpl($T2,$T3),
   convert($T4,$T3),
   eval($T4,(a=1,b=2,x=3),$ANS),
   write($ANS)
```

数式処理ルーチンの主なものとして、微分演算の他、行列演算、線型・非線型方程式の求解、多項式の処理、極限操作、Taylor 展開、積分、因数分解、微分方程式の求解等が考

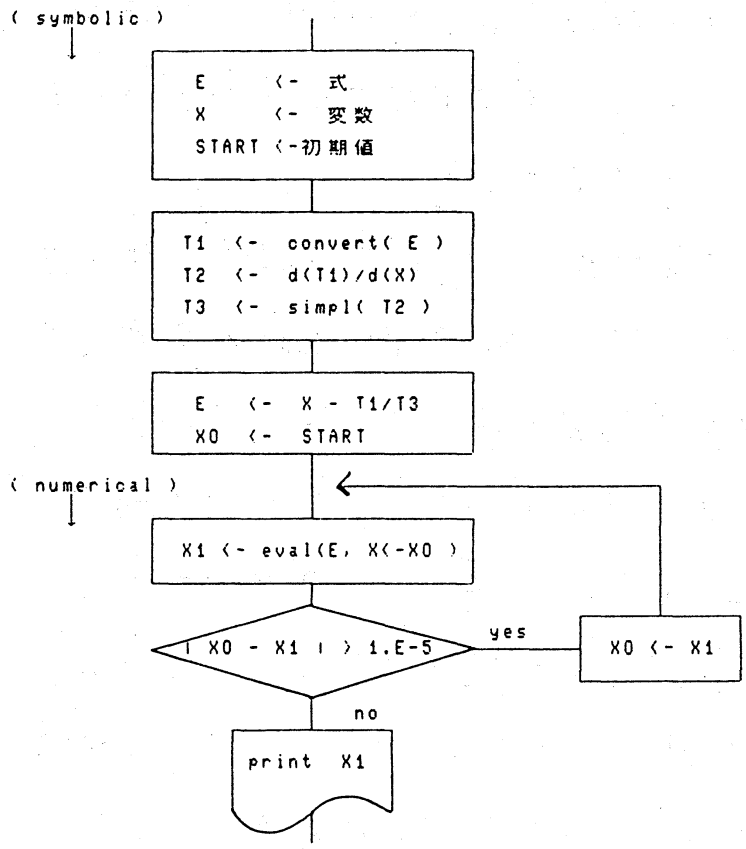
えられる。我々のシステムは、後にまとめるように、現段階でもこれらの多くの部分の処理が可能であるが（5. 参照）、ここでは、特に非線型方程式（代数方程式）の求解処理について詳説する。本システムの特色である数式処理と数値計算の結合が有効に行されていることが示される。

4. 数式処理と数値計算の結合

代数方程式の数式処理部分の制御は、基本的にBundy等により開発されたPRESSの制御方式を踏襲している[6]。以下この点に関し簡単に述る。PRESSでは数式の評価を3段階に分けている。第1段階（isolation）では、方程式等に出現する未知数を抽出する（例：$x-a=b \rightarrow x=a+b$）。第2段階（collection）は未知数の式中での出現回数を1回にし（例：$x\cdot a + x\cdot b \rightarrow x\cdot(a+b)$、$2\sin x\cdot\cos x \rightarrow \sin 2x$）、第3段階（attraction）では形式的には第2段階に類似しているものの、演算の「近い」未知数をまとめる（例：$x\cdot a + x\cdot a\cdot b \rightarrow a\cdot(x + x\cdot b)$）操作を行う。これら3段階は具体的な算術演算を行いはしないが、方程式に対するある種の演算である。そこで、<u>オブジェクト・レベル</u>の演算である通常の算術演算と区別して、<u>メタ・レベル</u>の演算（あるいは推論）と呼ぶ。PRESSにおける数式の処理の

制御は、このメタ・レベルの推論によっている。この方式を採用することにより、制御方式に階層を持たせることができシステム全体の構造化を進めることが容易となる。PRESSでは各段階ごとに生成された数式とのパターン・マッチング処理をほどこしているが、我々の場合、パターン・マッチングの回数を減じることにより、効率化をはかっている。

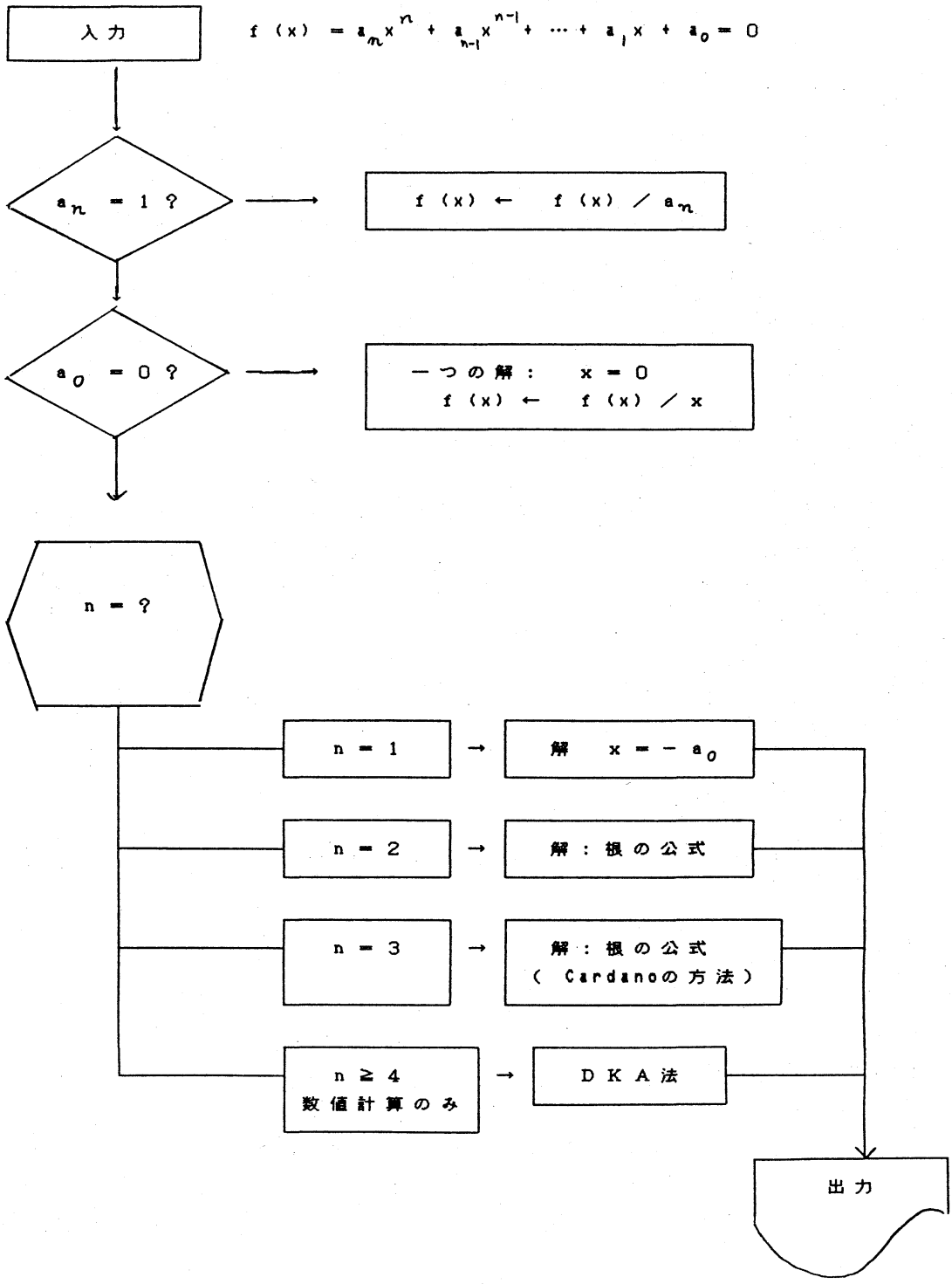
代数方程式の求解に対して、数式処理と数値計算を効果的に遂行する例として左図を考える。大雑把にいえば、良く知られたニュートン法のProlog化ともいえる。対象とする多項式（E）、その変数（X）および二ュートン法の初期値（START）が入力される。Eに対して上で述べたような制御方式による処理が施され、変数による数式



微分が行われる。結果に対して再度 PRESS的処理がなされ、簡素化が完成する。以後は制御が数値計算にゆだねられる。すなわち、xの旧値X0に対してのニュートン反復で新値X1を生じ、さらにこの値を旧値とし直し次の反復を行う。数値が求められるべき反復式（E）は数式でしかも未知数（今の場合のx）の出現回数を最小にして得られているため、解は良い精度で少い反復回数で求めることができる。なお、我々のものとは違った視点からではあるが、ニュートン法に対し数式処理（REDUCE）と数値計算（FORTRAN）を併用し、代数方程式の根を単根のみならず多重根まで求めようとする試みが方されていることを付記する[7]。

より総合的にかつ実用性を加味して、代数方程式の根を得るために、我々のシステムの数式・数値混合処理を有効に使用する例を次に示す。簡単な処理の流れは次ページに図示される。入力される代数方程式は、上のニュートン法の場合と同様の処理を受け簡素化された後、$x=0$の根が抽出される。これらの処理後の方程式は、その次数nにより解法が分類される。根の公式は$n=1,2$および$n=3$の特殊な場合に適用され、その他の場合には、全根同時反復法である数値解法の「DKA法」による数値計算が遂行される。DKA法はニュートン法のある種の変形であるが、通常の手法のよう

に減次操作を必要としない。減次操作には、浮動点数による因数分解操作が必要となり、数式処理的手法とは適合しないことは明かである。この場合のシステムへの入力およびシス



テムからの応答は $n=2$ の場合と $n=4$ の場合について、次の通りである。

```
?-algeb(x^2+b*X+c,X,$ANS)
    (((b^2-4*c)^(1/2)-b)/2,-(b+(b^2-4*c)^(1/2))/2)
?-algeb(x^4+2*X^3+3*X^2+4*X+5,x,$ANS)
    (-1.28782+0.857897*i,0.287815+1.41609*i,
     -1.28782-0.857897*i,0.287815-1.41609*i)
```

ここでの数値計算は Prolog を記述している C によっているが、一般に C から FORTRAN のサブルーチンを呼ぶことが可能である。このため、蓄積された豊富なライブラリを使用しうる。また本システムでは数値計算終了後も数式処理の入力状態が継続する。よって計算結果をそのまゝ次の計算に使用することが出来る。

5. 小型数式処理システムとしての機能

3., 4. では、我々の開発したシステムの概要および数式処理と数値計算の結合について述べた。次に既存のマイコン上で稼動する代表的な小型数式処理システムである muMATH との比較・検討をする。muMATH は LISP 系言語である muSIMP で記述されており、最新版では浮動小数点演算の機能も付加されている。簡単な計算の例として数式微分を実

行するための命令は

muMATH　　? DIF(a*x^2+b*x+c, x);

本システム　　?-diff(a*x^2+b*x+c, x, $ANS)

となり、出力変数の指定が異るのみである。各種計算についての比較を下表に示す。

比較項目	muMATH	our's
記述言語	LISP系	Prolog系
使用OS	CP/M, MS-DOS	Cが稼働すればよい (CP/M, MS-DOS, RT-11)
数値計算とのインターフェース	なし	Cにより行う
数式処理機能		
簡素化	スイッチ選択	最簡形
代数方程式	○	○
行列演算	○	○
連立一次方程式	○	○
微分	○	○
極限（部分的）	○	○
Taylor展開	○	○
積分（部分的）	○	×
因数分解	×	×
平均処理速度（現状）	1	7
その他の機能		
連続計算（複雑な計算のプログラム）	LISP (muSIMP)	Prolog
プログラムの容易さ	×	○
浮動小数点演算	muMATH-83 のみ可	○

数式処理機能に関しては、処理可能のものに　○印を、不可能のものに　×印を付した。処理速度は各処理に要する時間の割合（muMATHを1（秒）として）を表わしている。

17ü

数値計算との結合（インターフェース）は、既に述べたように数式処理システムの一般化、普及のための重要な機能であるが、muMATH 等では考えられていない。また、連続計算の例として、上記の ax^2+bx+c の導関数を $a=1$, $b=2$ に対し、$x=3$ と $x=3.1$ で評価しその差を求める計算を考える。muMATH（新版の muMATH-83 でしか浮動小数点の計算が不可能だが）では多段の対話的処理、あるいは muSIMP により変数の束縛を考慮した複雑なプログラムをする必要がある。これは LISP 系言語に不慣れな利用者に対し、数式処理システムの使用を困難ならしめる大きな要因にもなっている。一方、我々の場合には

```
?-diff(a*x^2+b*x+c,x,$F),eval((a=1,b=2),$F,$G),
  eval(x=3,$G,$T1),eval(x=3.1,$G,$T2),
  sub($T1,$T2,$ANS),write($ANS)
```

のプログラムのみである。プログラムの複雑さは減少し、かつプログラム言語に対する特別な知識もさして必要としない。なお、数式処理の実行速度は、現在では muMATH の方が平均で約7倍高速であるが、structured sharing 方式等 Prolog 処理系の高速化により、我々のシステムでも muMATH 以上の演算速度は十分に見込むことができる。

6. まとめ

以上に述べたように、我々の開発した小型数式処理システムは、

数値計算との有効なインターフェースを持ち、

プログラムが容易であり、

マイコン上で稼働し移植性が良い

という特色を持つ。特に、マイコン上で稼働するような小型数式処理システムでは記述言語としての Prolog の採用が有効であることがわかった。数値計算とのインターフェースは数式処理システムの普及の面からも必須不可欠であるが、これはその制御を C によっていることに依存している。

今後はこのシステムをより高度なものにするため数式処理システムの機能として積分、因数分解等の付加がある。また、現在は数式をシステム内で演算子の前置形式に変換して処理しているため、記憶領域縮小の演算の必要上、各段階で簡素化の述語 simpl を用いている。これに対し、入力時に「順序付け」処理をすることによってもシステムの高速化が可能である。さらに、代数方程式の求解のような数式・数値混合処理の拡充のために、必要な数値計算アルゴリズムの取捨選択あるいはそのデータベース化等が重要である。数値計算との結合を考える限り数式処理に適合する数値計算アルゴリ

ズムの開発、あるいは数値計算に適合する数式の簡素化等への対応の課題がある。

参考文献

1) Hulzen, J.A. and Calmet, J. : Computer Algebra Systems , in "Computer Algebra Symbolic and Algebraic Computation" ed. Buchberger, B. et al., pp.221-243, Springer-Verlag(1982).

2) Cole, C.A. and Wolfram, S. : SMP - A Symbolic Manipulation Program , SYMSAC 1981, pp.20-22(1981).

3) Rall, L.B. : Differentiation in Pascal-SC (Type GRADIENT) ACM Trans. Math. Softw., vol.10, No.2, pp.161-184(1984).

4) Bundy, A. and Welham, B. : Using Meta-level Inference for Selective Application of Multiple Rewrite Rule Sets in Algebraic Manipulatin , Artificial Intelligence, vol.16, No.3, pp.189-212(1981).

5) 多くの関連文献および解説書がある。その一部を示すと、

- Clocksin, W.F. and Mellish, C.S. : Programming in Prolog , Springer-Verlag(1981).
- 永田守男：数式処理におけるＰｒｏｌｏｇ，情報処理，vol.25 No.12, pp.1404-1509(1984).
- 後藤滋樹：Ｐｒｏｌｏｇの言語機能詳説，情報処理，vol.25, No.12, pp.1319-1328(1984).
- 中村克彦：Ｐｒｏｌｏｇ処理系，情報処理，vol.25, No.12, pp.1329-1335(1984).

6) 次の文献でも同様の手法が用いられている。

- 武脇敏晃，宮地泰造，国藤進，古川康一：Ｐｒｏｌｏｇによる数式処理システム試作の構想，日本ソフトウェア科学会第１回大会論文集，1B - 4, pp.29-32(1984).

7) Ojika, T., Watanabe, S. and Mitsui, T. : Subroutine Package NAES (Nonlinear Algebraic Equation Solver) , Computer Programming Laboratory, RIMS(1981).