パーソナルコンピュータ用数式処理システムの拡張(3)

大久保　明伸*

Extension of Symbolic Manipulation System Using Personal Computer

Akinobu OHKUBO

Abstract

The authors have developed symbolic manipulation system for technical or applied mathematics.

In this paper, the authors describe faculties of symbolic manipulation system (UMATH) on personal computer.

This system has been implemented on muSIMP, and has fundamental functions for symbolic manipulation, control faculty of evaluation (EVAL in LISP), the other function using hybrid method.

1．まえがき

パソコン上での数式処理システムは，次第に普及しつつある。パソコンの処理速度と記憶容量の増大により，益々この傾向は広がるものと考えられる。著者は，数式処理システムの利用法，拡張等に関する提案や機能の拡張について発表してきた。

応用数学や工学教育で現れる各種の数学公式やアルゴリズムは演習を通して修得される。演習は数値計算を除いて，手計算によりおこなわれる。このために出題される演習問題は，比較的簡単に解ける問題にならざるをえない。たとえばテイラー展開，微分方程式の級数による解法，要素が数式であるような行列の計算や多項式の四則などは，少し複雑な式でも，実際に計算してみるとかなりの困難を伴う。そのために学生は，教科書に印刷された解答をみて，理解することが要求される。時には，数式を暗記することにもなる。数値計算で電卓を利用するように，数式処理システムの電卓化はこのような問題点を解決するものとおもわれる。また，このシステムを利用することで，計算教育の方法も改善できるはずである。このような電卓的な利用法を一歩進めて，プログラム教育やCAIへの利用なども考えられる。このために必要な機能は，一般の記号処理言語ではまだもちあわせていない。

現在，入手可能な数式処理システムでも，上述したような観点にたって製作されたものではないので，教育用システムとして利用しようとした場合はそのための機能を強化しなければならない。強化された関数群は，既に発表したので，本報告では，このシステムの内部的な構造とプログラム製作に必要な技法について報告する。

2．muSIMPによるLISPインタプリタ

学生に記号処理言語として，最初にLISPを教えると，CARやCDRなどのＳ式処理関数の必要性が理解できないようである。特にLISPの教科書の最初の例題はこれらの関数に関するものが多い。FORTARANやBASICで取り上げられる例題は数値を計算するという意味で，具体的であるが，これに対してLISPの例題は一般に抽象的であ

*宇部工業高等専門学校電気工学科

る。講義する側にとっても，評価関数(EVALやAPPLY)の説明には特に困難を感じる。本来入門者に対しては，記号処理関係の例題でも，具体的な内容の例題が望ましい。たとえば、S式のようなデータ構造は，本来，内部データ構造なので，数式のような具体的なものをどのようにS式に写像すべきかを示すべきであろう。この過程で，CARやCDRの必要性が認識できるはずである。

以上の観点から，muMATHの数式の構造を用いて教育したほうが，S式の理解やLISPの基本関数の必要性を理解しやすい。教える側にとっても，EVALやAPPLYの説明も容易である。

一方，数式処理プログラムを学生自身が製作したり，数式処理プログラムを変更したりするためにも，LISPの理解が必要である。また，LISPは共通の記号処理言語としても重要であるし，インタプリタ本体の動作を認識する必要がある。そこでS式の解釈の原理を示すために，LISPインタプリタを製作した。インタプリタはmuSIMPで記述し，プログラムの例としても利用できるようになっている。また，学生はこのLISPの機能を拡張することもできる。後に示すように，関数の評価の制御機能を使用すれば，評価の過程を，段階的に理解することができる。このLISPの主要部分を図1にしめす。また，この処理系をPASCALで記述したものがあるので，実際には高速に実行することもできる。

3．評価（EVAL）の抑制について

普通，LISP系でのS式の評価で，途中の過程を保存することはできない。途中の過程が必要な場合は，トレーサを使用する。しかし，数式処理においては，関数は階層的に定義して，その評価をおこないたいことが多い。また，いつも関数は具体的な定義が与えられるものでもない。計算の途中から，具体的な関数を使用したい場合もある。例えば，F(X)やX＊G(X)などの微分や積分も可能でなければならない。また，F(X)＝SIN(X)とG(X)＝X＾3で定義された関数H(X)＝F(X)＊G(X)を微分する場合，muSIMPでは，

？ DIF(H(X)，X)；

と入力する。結果は，

@：COS(X)＊X＾3+3＊SIN(X)＊X＾2

となる。関数Hの中間構造（F(X)＊G(X)）のレベルで評価を止めることはできない。このため結果が必要以上に複雑になることもある。また途中結果から得られる情報を捨ててしまうことになる。とくに電卓的な使用では，この機能の必要性をかんじる。たとえば，G(X)の評価を抑制することで，

@： COS(X)＊G(X)+SIN(X)＊DIF(G(X)，X)

のような中間結果を得ることができる。H(X)の評価を同時に抑制すれば，使用者はこの関数の微分公式を得るはずである。

このような機能を実現するには，muSIMPのをEVLとAPPLYを書き換えればよい。また，別の方法としては，処理される関数（プログラム）そのものを，一時的にかきかえればよい。本報告では後者の方法で実現した。図3に示すように，評価を抑制する関数名をMASK関数の引数にあたえる。以後抑制された関数は，評価されても自分の関数名と引数をリストとして帰す。関数本体は，別の場所に保存される。評価の抑制を解除するときは，UNMASK関数で解除される。この2つの関数のアルゴリズムを図2に示す。また実行例を図3に示す。図2のPROPERTY INFIX：は関数の定義を，単純化するためのプログラムである。図4は実行例に使用した関数の構造図である。

？ F(X)：SIN(X)+SIN(2＊X)；

とすれば，F(X)が右のように定義される。もちろん，従来の代入の機能はそのままである。

関数の評価の抑制機能は，途中結果を得るという利点ばかりでなく，関数方程式や公式データベースの構築にも必要な機能である。本来，この機能はインタプリタ本体に組込むべきである。次節の技法を用いれば，本体にくみこむことは容易にできる。

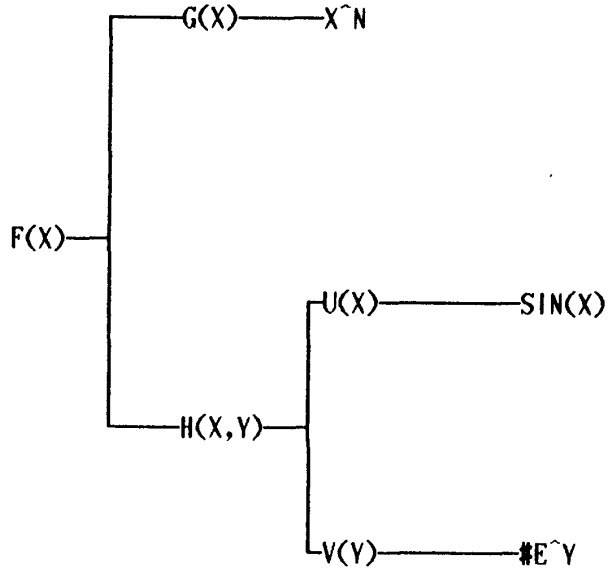

図4 階層構造関数の例

```
FUNCTION PLAPPLY(FN,ARGS,PLIS),
 WHEN ATOM(FN),
  WHEN FN='CAR,  FFIRST(ARGS),                        EXIT,
  WHEN FN='CDR,  REST(FIRST(ARGS)),                   EXIT,
  WHEN FN='CONS, ADJOIN(FIRST(ARGS),SECOND(ARGS)).EXIT,
  WHEN FN='ATOM, ATOM(FIRST(ARGS)),                   EXIT,
  WHEN FN='EQ,   FIRST(ARGS) EQ SECOND(ARGS),         EXIT,
  WHEN FN EQ PLEVAL(FN,PLIS),FALSE,                   EXIT,
  PLAPPLY(PLEVAL(FN,PLIS),ARGS,PLIS),
 EXIT,
 WHEN FIRST(FN)=LAMBDA,
   PLEVAL(THIRD(FN),PAIRLIS(SECOND(FN),FIRST(ARGS),PLIS)),
 EXIT,
 WHEN FIRST(FN)=LABEL,
   PLAPPLY(THIRD(FN),ARGS,ADJOIN(ADJOIN(SECOND(FN),THIRD(FN)),PLIS)),
 EXIT,
ENDFUN;

FUNCTION PLEVAL(FORM,PLIS),
 WHEN ATOM(FORM),PLASSOC(FORM,PLIS),EXIT,
 WHEN FUNC(FIRST(FORM)),
      PLAPPLY(FIRST(FORM),EVLIS(REST(FORM),PLIS),PLIS),
 EXIT,
 WHEN FIRST(FORM)='QUOTE,SECOND(FORM),EXIT,
 WHEN FIRST(FORM)='COND,EVCON(REST(FORM),PLIS),EXIT,
 WHEN FIRST(FORM)=PLEVAL(FIRST(FORM),PLIS),EVLIS(FORM,PLIS),EXIT,
 WHEN NOT(NAME(FIRST(FORM))),EVLIS(FORM,PLIS),EXIT,
 PLEVAL(ADJOIN(PLEVAL(FIRST(FORM),PLIS),ADJOIN(REST(FORM),FALSE)),PLIS),
ENDFUN&;

RDS();
```

図1　LISPインタプリタ（一部）

```
#MASK:FALSE:
FUNCTION MASK(X,Q),
ASSIGN(COMPRESS(LIST('#,X)),GETD(X))
  Q:GETD(X),
  PUTD(X,LIST('FUNCTION, SECOND(Q),ADJOIN(LIST ,ADJOIN(X,SECOND(Q)))))
ENDFUN&

FUNCTION UNMASK(X),
PUTD(X,EVAL(COMPRESS(LIST('#,X)))),
ASSIGN(COMPRESS(LIST('#,X)),COMPRESS(LIST('#,X))),
ENDFUN&

SUBROUTINE QUOTE LEX1,
 LEX1,
ENDSUB$

PROPERTY INFIX,:,COND(
  WHEN NAME(EX1),
   LIST(':,EX1,PARSE(SCAN(),20)) EXIT,
  WHEN NOT(SUM(EX1)AND PRODUCT(EX1)AND POWER(EX1)),
   PUTD(FIRST(EX1),LIST('FUNCTION ,REST(EX1),PARSE(SCAN(),20))) EXIT,
 SYNTAX());



RDS();
```

図2　評価の抑制関数

```
@: PRINTLINE

?
F(X,Y):G(X)*H(X,Y);
@: X (Y)

?
G(X):X^N;
@: X ()

?
H(X,Y):U(X)+V(Y);
@: X (Y)

?
U(X):SIN(X);
@: X ()

?
V(Y):#E^Y;
@: Y ()

?
F(A,B);
@: A^N #E^B + A^N SIN A

?
F(0,0);
@: 0^N

?
DIF(DIF(F(X,Y),X),Y);
@: #E^Y X^(-1 + N) N

?
MASK(G,H);
@: FUNCTION (X),
     LIST (G, X)
   ENDFUN

?
DIF(F(X,Y),X);
@: #E^Y DIF (G (X), X) + DIF (G (X), X) SIN X + COS X G (X)

?
UNMASK(H);
@: #H

?
DIF(F(X,Y),X);
@: #E^Y DIF (G (X), X) + DIF (G (X), X) SIN X + COS X G (X)

?
MASK(U);
@: FUNCTION (X),
     LIST (U, X)
   ENDFUN

?
DIF(F(X,Y),X);
@: #E^Y DIF (G (X), X) + DIF (G (X), X) U (X) + DIF (U (X), X) G (X)

?
TAYLOR(F(X,Y),X,0,4);
@: X DIF (G (0), 0) U (0) + X DIF (U (0), 0) G (0) + #E^Y X DIF (G (0), 0) + #E
^Y X^2 DIF (G (0), 0, 0)/2 + #E^Y X^3 DIF (G (0), 0, 0, 0)/6 + #E^Y X^4 DIF (G
(0), 0, 0, 0, 0)/24 + #E^Y G (0) + X^2 DIF (G (0), 0) DIF (U (0), 0) + X^2 DIF
(G (0), 0, 0) U (0)/2 + X^2 DIF (U (0), 0, 0) G (0)/2 + X^3 DIF (G (0), 0) DIF
(U (0), 0, 0)/2 + X^3 DIF (G (0), 0, 0) DIF (U (0), 0)/2 + X^3 DIF (G (0), 0, 0
, 0) U (0)/6 + X^3 DIF (U (0), 0, 0, 0) G (0)/6 + X^4 DIF (G (0), 0) DIF (U (0)
, 0, 0, 0)/6 + X^4 DIF (G (0), 0, 0) DIF (U (0), 0, 0)/4 + X^4 DIF (G (0), 0, 0
, 0) DIF (U (0), 0)/6 + X^4 DIF (G (0), 0, 0, 0, 0) U (0)/24 + X^4 DIF (U (0),
0, 0, 0, 0) G (0)/24 + G (0) U (0)
```

図 3 関数評価抑制の実行例

4．基本グラフィック関数の埋め込み

数値計算の結果をグラフ化することは，よくおこなわれることである。数式処理においても，得られた結果の式をグラフ化することは必要である。たとえば，ある関数を多項式に展開し，グラフ化したとする。学生は，より関数展開の意味を理解できるであろう。また，従来の応用数学で講義される微分方程式の級数による解などは一か二つの例しか体験していないのが普通である。これをグラフ化することもまれであろう。展開の中心の意味付けも，グラフ化することで，より具体的に理解できる場合が多いものと考えられる。

さらに，この機能はプログラム教育においても必要である。とくに再帰技法を修得させるための例題には，再帰図形を描かせることが多い。学生は，この図形を描くことで再帰技法を，比較的容易に修得できる。また，ロボットなどのAI的なシミュレーション実験等にも必要になろう。

muSIMPでの図形の処理方法はつぎの方法が可能である。

(1)PRINT関数またはPRIMATH関数で，X-Yプロッタに出力する。

(2)ハイブリッド処理の技法を用いて，BASICやFORTRANで間接的に図形出力をおこなう。

(3)muSIMPに図形処理関数を付け加える。

(1)と(2)に関しては，すでに文献に述べた。本報告では(3)についてのべる。原理はパソコン上のROMBASICにあ

```
GLINE:  CALL EXPLODE；ｍｕＳＩＭＰ（GLINE(A1,A2,A3,...,A9)）からの引数を作業番地
                    ；ARG1,ARG2,...ARG9 に格納する。
        MOV   AX,CS:[ARG1]；ARG1（１ワード）をLINE命令の１番目の実引数とする。
        MOV   CS:[LINE],AX
        MOV   AX,CS:[ARG2]；ARG2（１ワード）をLINE命令の２番目の実引数とする。
        MOV   CS:[LINE+2],AX
        MOV   AX,CS:[ARG3]；ARG3（１ワード）をLINE命令の３番目の実引数とする。
        MOV   CS:[LINE+2],AX
              .
              .
              .
        MOV   AL,BYTE PTR CS:[ARG9] ；ARG9（１バイト）をLINE命令の９番目の実引数
        MOV   CS:[LINE+2],AX        ；とする。
        SAVESIMP                    ；muSIMPの現在のレジスタを保存する。
        MOV   BX,OFFSET LINE        ；LINEの引数のオフセットを設定する。
        INT   0A7H                  ；ROM のLINE命令を実行する。
        RETURNSIMP
        IRET
LINE    DW    0,0,0,0               ；実引数（ワード）
        DB    0,0,0,0,0             ；実引数（バイト）
```

図５　MASMによるGLINEの定義例

```
FUNCTION FIBO1(N,X,Y,HX,HY,P,Q),
  WHEN N=1 ,
   GLINE(X,Y,X,Y+HY,RED,0,0),
   GC(X,Y+HY,1,1, RED ,0),1,
  EXIT,
  WHEN N=2,
   GC(X,Y,1,1,GREEN,0),1,
  EXIT,
  GLINE(X,Y,X+HX ,Y,GREEN,0,0),
  P:FIBO1(N-1,X+HX ,Y,HX,HY),
  GLINE(X,Y,X,Y+ L(N-1)*HY,RED,0,0),
  Q:FIBO1(N-2,X,Y+L(N-1)*HY,HX,HY),
  P+Q,
ENDFUN&

FUNCTION FIBO(N,X,Y,%LOCAL:%HX,HY),
HX:QUOTIENT(630-X,N-2),
HY:QUOTIENT(390-Y,L(N)),
BLOCK WHEN HY=0,HY:1,EXIT,ENDBLOCK,
FIBO1(N,X,Y,HX,HY),
ENDFUN&

FIBO(10,3,3);

RDS();
```

図6 グラフィック関数の使用例

るグラフィックルーチンを使用することである。

muSIMPでは200番地から3番地おきに,2DD番地までにINT 81からINT 8F命令が格納されている。これはユーザの機械語ルーチン定義用の番地である。全部で16個の関数が定義できる。UMATHでは，この割り込み命令にROM上のグラフィックルーチンを割り当てるメモリ常駐プログラムを製作した。

このなかのINT 81をN88BASICのLINE命令に割り当てる方法について概略する。図5にグラフィック関数GLINEの流れとmuSIMPのなかでの定義法を示す。

muSIMPのアトムセルやノードセルのメモリ内の配置はマニュアルを参照してほしい。これらの関数のmuSIMP側のプログラムを図6に示す。

この手法は，図形ばかりでなく数値計算やファイル処理などLISP系の言語では弱い機能を付加するのに,応用できる。

6．まとめ

この報告では，記号処理の教育や数式処理を使用した教育をおこなうための道具類について述べた。わずかな改造でmuSIMPは教育用システムとして利用可能である。近い将来，高専においても，本格的にAI関係の講義がおこなわれるであろう。このための基礎教育用として，このシステムは意味がある。共同利用の計算機と違ってパソコンは益々安価で高機能化し，このシステムを利用できる環境が整いつつある。

今後は,数式処理を利用した図形処理や知的なCAIシステムの開発を行う予定である。

謝 辞

日頃，お世話になる電子計算機室関係者に謝意を表します。

参考文献

1) muSIMP/muMATHはCP/MまたはMSDOSで動作する数式システムである。The Soft Warehouseに所有権がある。

2) 大久保：BASICによる構造化プログラミング,宇部高専研究報告，58年3月29号

3) 大久保：数式処理システムの電卓的な利用法について，高等専門学校，情報処理教育研究協議会，58年8月3号

4) 大久保：パソコンによる数式処理システムの利用,電気四学会中国支部連合大会講演論文集，58年

5) 大久保：パソコンによる数式処理システムの利用(2),電気四学会中国支部連合大会講演論文集，59年

6) 大久保：パーソナルコンピュータによる数式処理,宇部高専研究報告，59年3月30号

7) 大久保：パーソナルコンピュータ用数式処理システムの改善について，情報処理学会第28回全国大会講演論文集，59年

8) 大久保：パソコンによる数式処理システムの利用(3),電気四学会中国支部連合大会講演論文集，60年

9) 大久保：パーソナルコンピュータ用数式処理システム（UMATH）の拡張，宇部高専研究報告，60年3月31号

10) 大久保：パーソナルコンピュータによる数式処理(その1)，数式処理通信（Vol.1-No.3）サイエンティスト社

11) 大久保：パーソナルコンピュータによる数式処理(その2)，数式処理通信（Vol.1-No.4）サイエンティスト社

(昭和62年9月20日受理)