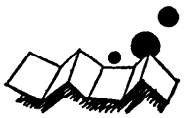
解　説

小型数式処理システムとその応用†



対馬勝英††

1. 序

従来，数式処理はREDUCE，MACSYMAなどに見られるように，大型電子計算機上でのみ利用可能であった．しかし，マイクロコンピュータ技術の進歩により，1979年頃よりmuMATHに代表されるパソコン上で利用できる小型システムが利用可能となった．後者は機能は低いが低価格であり，当初は数式処理の学習や数学教育に使用する簡易システムとしてとらえられていた．

その後，16ビットパソコンの普及に伴い，後者もある程度の規模の処理のできる実用的ツールとしての市民権を獲得し始めた．ところが，最近になり，REDUCEが高位パソコン上でも使用できるようになり，上述の小型システムの市民権が脅かされ始めた．大型数式処理システムがパソコンのレベルに降りてくるという傾向は，今後，さらに増すことが確実視される．この状況のもとで，低コストで利用できるという小型システムの特徴は失われようとしている．

その意味で，現在は，従来の「小型数式処理システム」の意義が問い直されているときである．

本解説では，この現状認識を踏まえて小型システムの現状を分析し（2章），それに基づき小型システムの持つべき特徴を論じる（3章）．研究，教育のために小型システムを変更して使用した体験を述べ，小型システムは特定分野で使われる特殊化した数式処理システムを作る際に有利であることを示す（4章）．

これらより，小型システムにも，高速性，汎用性を目指す大型の数式処理システムとは異なった独自の発展の可能性があることを述べる（5章）．

2. 小型システムの実例

今まで，作成された数式処理システムは60種類くらいあると言われる[1]が，パソコンの上で利用できる商用システムは，muMATHのみであると言って差し支えない．

最近，68000をCPUとする高位パソコン上にREDUCEが移植され，商用システムとして市場に登場した[2]が，これについては2.2節でふれたい．

2.1 muMATH

muMATHの最初のバージョンであるmuMATH 79[3]は，A. D. RichとD. R. Stoutoemyerにより，8080をCPUに持ち，CP/M 80をOSとするパソコンの上に，教育とパーソナルユースを目的として作成された．muMATH-79は式の簡素化，微分，積分，行列，代数方程式の解法，三角関数など，汎用システムと呼べる機能を備えていた[4]．しかし，メモリ空間が64Kバイトしかないため，すべてのパッケジを連結したSYSファイルが作れず，上記の豊富な機能も併用できず利用しにくく，本格的な処理を試みるとメモリ不足のために目的が達成できぬケースが多かった．

その後，パソコンの機能が向上し，8086をCPUとする16ビット機の上で作動するmuMATH-79の改訂版であるmuMATH-83[5]が，新たに供給された．これは256Kバイトのメモリ空間を利用でき，ユーザに開放された作業領域はmuMATH-79に比較して，実質的に十倍近く増加した．また，全パッケジを連結したSYSファイルを利用できるようになった．

新たに増えた数式処理の機能として，常微分方程式の解法，連立一次方程式の解法などがある．また，muMATHの記述言語muSIMPの機能も近似化され，強化された．そのため，muMATH-79とmuMATH-83では内部の記述が相当異なっている．

なお，mu MATH-83は，CP/M 80版のものも供給されている．

機　能

muMATH-83は表-1に示した機能を持つが，処理機能の種類の豊富さという点では，REDUCEと比べてあまり見劣りがしない．REDUCEはLET（図-1

† Small Computer Algebra Systems by Kathuhide Thushima (Osaka Electro-Comunication Univ., Faculty of Engineering).

†† 大阪電気通信大学工学部

表-1 muMATH 83 の標準機能

高桁有理数の計算
有理式，多項式の計算
微分，積分
ベクトル，行列演算
代数方程式，線型連立代数方程式の解法
Σ, Π の演算
浮動小数点表示
常微分方程式の解法
極限値の計算
三角関数，指数関数などの簡単化
など

```
We set the history stack depth (HISTLEN) as 8
0 ? LET A*B=C;
@:  TRUE
1 ? EXPAND (A+B)^4;
@:  4 B^2 C+6 C^2+4 A^2 C+A^4+B^4
2 ? LET DEFINT (X, X, 0, PINF)=PINF;
@:  TRUE
3 ? DEFINT (X, X, 0. PINF);
@:  PINF
4 ? LET DEFINT (#EXP, #EXP, 0, PINF)=PINF;
@:  TRUE
5 ? LET DEFINT (L, L, 0, PINF)=PINF;
@:  TRUE
6 ? DEFINT (Z, Z, 0, PINF);
@:  PINF
```

図-1 LET の機能

0 で $A*B$ を C に置換することを宣言した．1 で $(A+B)^4$ を展開すると出現した $A*B$ はすべて C に置換された．2 は定積分 $\int_0^\infty XdX$ を ∞ に置換することをシステムに教えた．3 はその結果である．このままでは $\int_0^\infty ZdZ$ を処理しないので 4 で流通変数 #EXP を用いて宣言すると 6 のように，積分変数に関係なく ∞ を得る．この結果は 4.2 に述べるわれわれにより拡張された muMATH を用いて出力した．なお REDUCE では 4 は

FOR ALL L LET DEFINT(L, L, 0, PINF)=PINF

と記述すべきものである．

を参照されたい)，因数分解の機能を持つが，muMATH-83 はこれを持たず，機能的に大きな差となっている．

さらに，細かい点として，FACTOR，HISTORY，係数の取り出し，処理時間表示などの機能のないことがあるが，これらは若干の手直しで比較的容易に muMATH-83 においても実現できる[6) 7)]．

逆に，REDUCE にない機能として，リミットを求める LIM や常微分方程式の解法を行う ODE などが muMATH に備わっている．

記 述

muMATH は，muSIMP と名付けられた LISP に近い関数言語を用いて記述されている．muSIMP でのプログラムの記述方式を理解するために，階乗関数を REDUCE の Procedure に対応する muSIMP の FUNCTION を用いて記述した．

```
FUNCTION FAC(N),
  WHEN N=0, 1 EXIT,
  N*FAC(N-1),
ENDFUN $
```

muSIMP は LOOP が使える，変数を使いやすい，数学的表記法で記述できるなど，LISP を理解しないユーザにも利用しやすい．

muMATH-83 の大部分が muSIMP を用いて記述されているので，ユーザは LISP を知らなくても数式処理システムの構成を理解でき，慣れてくればシステムの持つ知識の追加や変更を行えるようになる．

muMATH のごく一部と muSIMP の多くは機械語で記述されている．muMATH を muSIMP で記述したソースリストが供給されているので，ソースが LISP で記述されている REDUCE や，ソースの供給されない MACSYMA などの大型システムに比較して，ユーザはシステムの動作を容易に理解することができる．

微分や三角関数の簡素化などの数学的知識（ルール）は，property と呼ばれる連想リストにモジュール化して記述されていて読みやすく，機能の追加も容易である（図-2）．

```
PROPERTY DIF 1, *, FUNCTION (EX 1, EX 2),
  EX 1*DIF 1 (EX 2, INDET)+EX 2*
  DIF 1(EX 1, INDET)
ENDFUN $
                    (a)
PROPERTY DIF 1, +, FUNCTION (EX 1, EX 2),
  DIF 1 (EX 1, INDET)+DIF 1 (EX 2, INDET),
ENDFUN $
                    (b)
```

図-2 property の例

(a)は $\frac{d}{dx}(f*g)=g\frac{df}{dx}+f\frac{dg}{dx}$

(b)は $\frac{d}{dx}(f+g)=\frac{df}{dx}+\frac{dg}{dx}$

という微分の簡単化の規則を表している．たとえば

$$\frac{d}{dx}(f*g+h)\xrightarrow{(b)}\frac{d}{dx}(f*g)+\frac{dh}{dx}$$
$$\xrightarrow{(a)}g*\frac{df}{dx}+f*\frac{dg}{dx}+\frac{dh}{dx}$$

のように(a)と(b)は連鎖的に適用される．EX 1，EX 2 が任意複合項であってもルール(a)とルール(b)は引用されて，微分の簡単化が行われる．このルールは，それだけでモジュール化されて記述される．

```
0 ? PUSH ('GIANTS, CENTRAL) $
0 ? PUSH ('TIGERS, CENTRAL) $
0 ? PUSH ('CARP, CENTRAL) $
0 ? CENTRAL &
@: (CARP TIGERS GIANTS)
1 ? POP (CENTRAL);
@: CARP
2 ? POP (CENTRAL);
@: TIGERS
3 ? POP (CENTRAL);
@: GIANTS
```

(a) 標準的なスタックの使用例
PUSH によりスタック CENTRAL に格納されたデータが POP により順次取り出される.

```
2 ? AA: A+B;
@: A+B
3 ? POP (AA);
@: +
4 ? POP (AA);
@: A
5 ? POP (AA);
@: B
```

(b) セットされた定義の取り出し
特に(b)の使用法は, muMATH において多用され記述効率を向上させている.

図-3 PUSH, POP の機能

LISP では, キイ入力した整式のS式への変換, S式の評価, S式の数式への変換を行うトップレベル関数をドライバ, またはREAD-EVAL-PRINT ループと呼ぶ. muMATH ではこれは DRIVER と呼ばれ, PARSE-EVAL-PRINT ループとなる. このドライバや EVAL により起動される整式の簡単化ルーチンは, その記述が数式処理システムの設計方式に関わるので初心者にはやや読み辛い.

また, 図-3(a) に示す PUSH, POP 関数があり, 特に図-3(b)に示した変数にセットされた値 (定義) をスタックより順に取り出す機能は, muMATH の記述において多用されている.

使 用 法

初歩的なユーザは, 対話的コマンドにより出力を得る. たとえば, $(x+y)^3$ を展開するには次のように入力する.

```
? EXPAND ((X+Y)^3);
@ X^3+3 X^2 Y+3 XY^2+Y^3
```

連続した操作をまとめて行う場合や, システムの持たない機能 (関数) を作り出す場合には, 前述の階乗関数 FAC(N) のように関数を作成して利用できる.

このように, 再帰定義が可能であるし, 見やすい形で関数を登録できる. 一度登録した関数は他の関数より参照でき, muSIMP はその上に乗った muMATH も含め, 言語としての自己増殖性を持っている.

多くのユーザはこの FUNCTION を記述することで所期の目的を達成でき, それ以上の機能を必要としない. 多くのユーザにとり, 式を簡素化するフラグの使用法が汎雑で難しい.

たとえば,

$A*(B+C)$ を $A*B+A*C$

とするには, フラグ NUMNUM を6に, 逆に

$A*B+A*C$ を $A*(B+C)$

とするには, フラグ NUMNUM を−6にする必要がある. (muMATH-83 にはこのようなフラグが 17 種あり, 各フラグの値は 15 個にも及ぶので, その管理は煩しい.)

速 度

muMATH-83とREDUCEの速度の比較を試みた. muMATH-83 は 10 MHz のクロックを持つ 8086 の改良版である V30 を CPU とする NEC 社の PC 9801 VM 2, REDUCE は TSS により京大大型センタの REDUCE 3 を利用した.

まず, 実際的な例における muMATH-83 と REDUCE の比較を試みた.

$(1+X)\wedge N$

を展開するのに要する時間を EXPAND, 展開したもの ANS を,

```
DF (ANS, X);
```

と微分するのに要する時間を DIF, 展開したもの ANS を

```
INT (ANS, X);
```

と積分するのに要する時間を INT とした.

おのおの, muMATH-83, REDUCE について, N=5, 10, 20, 30, 40 において計測し, それらの所要時間と比 (muMATH-83 の所要時間/REDUCE の所要時間) を表-2 に示した.

項数が増加するにつれて, REDUCE の方が相対的に速度が向上する. 特に, N=30, 40 ではメモリが少ないことによるガベージコレクションのために muMATH-83 にとって不利となる. これより, muMATH-83 と REDUCE の速度の比は, 2桁と3桁の間にあると言えよう. (なお, Z80 版の muMATH-83 は, MS-DOS 版より平均して5, 6倍遅い.)

最近, C/M 68K 上で稼動する REDUCE が市販され, 速度の速い LISP が開発されたこともあり, メ

表-2 muMATH-83 の速度
$(1+X)^N$ の展開，微分，積分を行って，muMATH-83 と REDUCE の速度の比較を試みた

N	処理名	muMATH 処理時間	REDUCE 処理時間	muMATH/ REDUCE
10	EXPAND	0.8	0.007	114
	INT	1.3	0.027	48
	DIF	0.4	0.011	36
10	EXPAND	3.2	0.019	168
	INT	3.5	0.041	85
	DIF	1.1	0.016	68
20	EXPAND	16.6	0.045	368
	INT	14.4	0.087	165
	DIF	2.4	0.030	80
30	EXPAND	75.5	0.082	920
	INT	40.0	0.134	298
	DIF	5.2	0.048	108
40	EXPAND	94.8	0.135	702
	INT	82.4	0.204	403
	DIF	8.3	0.076	109

(時間は sec)

インフレーム上の REDUCE との速度差が縮まってきた．将来，68020 のような 32 ビットの CPU がパソコンに登載された時点で，両者の速度差を論議する必要はなくなる．MS-DOS 版の muMATH-83 は 8086 上で動作している割には高速であるが，利用可能なメモリのサイズを 256 K バイトに限ることによりこの速度が得られた点に，限界が感じられる．

その他の機能

muMATH-83 のユーティリティは，muMATH-79 に比較して格段に強化された．ブレークポイントを設定したトレースが可能になり，システムの動作の理解，確認が容易となった．また，システムの持つ任意の関数や PROPERTY，SUBROUTINE をプリティプリントして表示する機能 DISPLAY が用意された．

また，システムの持つ関数，PROPERTY，SUBROUTINE や変数の値を，個別にファイルに書き込むユーティリティ WRITE が付加された．また，muSIMP 上で利用できるエディタがあり，さらに，TIME 関数が使用でき，処理時間の計測が容易に行える．

muSIMP

muSIMP はシンタクスシュガをまぶした LISP にほかならないが，ユーザは LISP に関する知識を持たなくてもこれを利用できる．PUSH，POP を用いてスタックが利用でき，スタックは大域変数の役割を果たすので，プログラムが相当書きやすくなった．特に，数式処理のアルゴリスム記述にとり，PUSH，POP は自然な記述を与えるようだ．図-4 に代入を行う SUB と関数の変数依存性を定義する DEPEND 1 を，PUSH，POP を使用した場合とそうでない場合について示した．

```
FUNCTION SUB (EX 1, EX 2, EX 3),
  WHEN EX 1=EX 2, EX 3 EXIT,
  WHEN ATOM(EX 1), EX 1 EXIT,
  ADJOIN (SUB (FIRST (EX 1), EX 2, EX 3),
  SUB (REST (EX 1), EX 2, EX 3)),
ENDFUN $
```

(a)

```
FUNCTION SUB (EX 1, EX 2, EX 3),
  WHEN EX 1=EX 2, EX 3 EXIT,
  WHEN ATOM (EX 1), EX 1 EXIT,
  ADJOIN (SUB (POP (EX 1), EX 2, EX 3),
  SUB (EX 1, EX 2, EX 3))
ENDFUN $
```

(a′)

```
FUNCTION DEPEND 1 (EX 1, LEX 2),
  LOOP
    WHEN ATOM (LEX 2) EXIT,
    PUT (EX 1, FIRST (LEX 2), 'DEPENDS),
    LEX 2: REST (LEX 2),
  ENDLOOP,
ENDFUN $
```

(b)

```
FUNCTION DEPEND 1 (EX 1, LEX 2),
  LOOP
    WHEN ATOM (LEX 2) EXIT,
    PUT (EX 1, POP (LEX 2),' DEPENDS),
  ENDLOOP,
ENDFUN $
```

(b′)

図-4 muSIMP-83 の記述と PUSH，POP
(a)は再帰的記述のうちで FIRST，REST (LISP の CAR，CDR) を用いて記述した場合で，(a′) は再帰的記述のうちで POP を利用した記述である．POP はスタックより破壊的取り出しを行う．
(b)は繰り返し (LOOP) のうちで first，rest を (b′) は繰り返しのうちで POP を用いた記述である．
(a)より (a′)，(b)より (b′) の方が易しい記述であり，かつ高速である．

一方，使用できる制御構造は，

WHEN～EXIT

BLOCK～ENDBLOCK

LOOP～ENDLOOP

のみであり，豊富とは言えない．ELSE のないことによる分岐の記述のしにくさが，muSIMP の欠点であり，構造的プログラミングは行えない．

ほかに，CATCH，THROW も使える．また，現在

の環境をそのまま保存する SAVE と，それを再びメモリにロードする LOAD がある．

また，mu SIMP から脱出しないで OS レベルの処理を行う EXECUTE が用意され，従来のような mu SIMP 上の作業と OS 上の作業の分離という難点が解消された．

OS の持つエディタを数式処理システム上より利用する，インタプリタを含む他言語で数値計算を行って mu MATH に復帰するなどの，OS 上の多面的機能が数式処理システムの上から利用できるようになり，小型数式処理システムも「環境」としての条件が整ってきた．さらに，I/O ポートと交信する機能 PORTIO が用意されているので，種々の外部機器の制御や第二のパソコンとの交信などが可能となり，パソコンの特徴を活かした利用方式が開発できる．特に，グラフィックスは，数式の柔軟な入出力の可能性という点で重要である．

2.2 その他のシステム

大型機，スーパミニコンでしか利用できなかった REDUCE をパソコン上で利用できるようにしたシステムが，国内外で市販され始めた．これを小型システムと呼ぶべきか否かためらわれるが，この大型機上のものをパソコンへという傾向が，パソコンの機能の向上とともに拡大することは確実である．

これらは 68000 を CPU とする最高位パソコン上の CP/M 68 K の上で利用できるが，REDUCE 3 の最大の特徴である因数分解を実用的に利用するには，1 M バイト以上のメモリ容量が必要とされる．メインフレーム上のシステムに比べて処理速度は遅くとも専有して使用できることにより，研究のツールとしては大きな役割を果たしているようだ[8]．また，低位オフコンレベルの UNIX システム上でも REDUCE が使用できるがパーソナルユースとしてはコスト上難点がある．

ほかに，代数計算を行う BASIC で記述した専用システムがある[9]．BASIC を用いる以上，汎用システムの構築には無理がある．パソコンの上の PASCAL を用いて汎用に近いシステムを記述することは可能であり，数値計算との複合利用を目指せる[10]．

また，パソコン上の prolog で記述されたシステムとして，愛媛大[11]，神戸大[12]，大阪電通大[13]などで開発された小型システムがある．野田たち[11]はC言語の上に prolog を作成し，さらにその上に数式処理システムを構築している．速度的に問題はあるが，彼らのシステムは数式処理と数値計算の結合や，制御方式の設計に関して，種々の試行錯誤を行う「試行システム」として機能し，3 章に述べる小型システムに固有のいくつかの特徴を持っている．また，村上ら[12]はパソコン上の prolog を用い，数式の処理の中間結果を表示することにより，教育効果を期待する教育用数式処理システムを作った．われわれのシステム[13]については 4 章で述べる．

小型システムの場合の大型システムと同様に，ユーザをどれだけ獲得できるかがシステムの成長，発展の鍵となる．muMATH はパソコン上で作動するので，すでに多くのユーザを獲得している．4 章に示すような，個々の研究分野のニーズに応じて mu MATH の機能を拡張する試みは，その意味で，各研究分野，応用分野において大きな可能性を持つ．

3. 小型システムの特徴

従来の数式処理システムと比較して小型システム，それもパソコン上の小型システムの特徴を述べたい．さらに，小型システムのみたすべき特徴について述べることで，今後の小型システムの設計への指針としたい．

（1） 経 済 性

小型システムの特徴として経済性がある．数式処理システムの利用に関する費用には，導入コストとランニングコストがあり，またランニングコストは純然たる計算費用と，ソフトウェアメンテナンスに要する費用に分けられる．一応実用的な計算ができる MS-DOS 上の muMATH-83 を利用すると，ハードウェア，ソフトウェアあわせて数十万円のコストがかかる（ソフトウェアのみなら十万円弱）．しかし，この場合，大型センタの TSS 利用のように計算費用で悩まされることはない（特に，数式処理は数値計算と異なり計算量の事前の予測が難しいので，計算費用に対する心理的な不安が生じやすい）．大型システムでは，ソフトウェアメンテナンスの費用が必要なものもあるが，小型システムではそれは必要ではない．パソコン上の REDUCE を利用するためには，現時点ではハードウェア，ソフトウェアを含めて百万円から二百万円のコストがかかる（ソフトウェアのみなら，高速 LISP をあわせて 30 万円強）[14]．

（2） 「閾」の問題

我が国において，大型センタのサービスする数式処理システムが利用できるようになって十年近い年月が

経過したが，その間のユーザの増加は，数式処理の潜在性可能性の割には著しいものではない．

この理由として，使用に際しての「閾」の高さがある．

数式処理システムの全体的な機能がつかみ切れず，ユーザが巧く使いこなせないことやレベルの高いプログラミングを行う際に再帰を含めたプログラミング技法が難しいことが閾を高くしている．また，計算時間を節約するためには，システムがどのようにLISPを用いて記述されているかを知って，プログラムを記述すると優利であるという側面がある．さらに，システムが内部に組み込まれた知識をどのように用いてその答えを出力したかがよく判らず，「ブラックボックス」としてしか使えぬためにシステムを把握しにくいという面がある．

一方，トラブルの生じた際に，大型機のように資料調べが拡散してゆくといった問題の生じない点は，パソコンを利用した小型システムの利点である．ソースリストを含めた資料が公開されていてコンパクトで，かつ手近にあることが，利用に関する閾を下げる効果を持つ（この点では，CP/M 68K 上の REDUCE も同様の特徴を持つので，今後のユーザの増加が見込まれる）．

さらに，muSIMP などの関数言語においては，トレース，デバッグの機能の充実が重要であるが，muMATH-83 はブレークポイントの設定もできるなどよく工夫されており，初心者にとり閾の低いシステムとなっている．

（3） 機能の改善の容易さ

muMATH の項で述べたように，ユーザの利用に関するレベルを

（a） 対話レベル

（b） FUNCTION レベル（REDUCE の procedure レベル）

（c） property を変更するレベル

（d） 組込関数まで変更するレベル

に大別できる．（REDUCE では，（a）で LET 機能を使えるので，特に（a）の比重が高い．）通常のユーザは，大型，小型システムを問わず，（a），（b）のレベルの利用に終始している．

しかし，muMATH では数学的知識，運用的知識が可読な形でモジュール化して与えられているので，REDUCE に比較してシステムの持つ知識（特に数学的知識）を手直ししやすい．（b）のレベルで記述しにくいアルゴリスムも，（c），（d）のレベルでは記述しやすい場合があるが，（c），（d）のレベルの作業は，通常のシステムでは LISP を理解せずには不可能である．muMATH では，このレベルも muSIMP を用いて作業が行える．これが muMATH の特徴の一つである．このように情報処理の専門家でないユーザにも，内部の手直しが可能なようにシステムをモジュラに記述することは，小型システムの設計理念として望ましい．

（4） 利用の専有性

従来，一部の恵まれた研究者を別にすると，数式処理システムを四六時中専有して使うことは夢でしかなかった．しかし，パソコン上の小型システムの普及や，REDUCE のパソコンへの移植などにより，上記の夢が実現しようとしている．

専有使用が実現すると，長大な処理の実行のみでなく，規模は小さいが数式処理システムとの多数回の対話が行われ，従来，紙と鉛筆で行っていた操作の多くが数式処理システムに移行する．その際に，柔軟な入出力機能の作成，数学知識データベースの引用機構の作成が重要になる．後者にはパーソナルなデータベースを構築することで，小型システムを特殊化するという側面がある．なお，数学公式のデータベースについて文献 15) がある．

（5） システムの特殊化

システムの専有的利用が増加すると，自らの研究上の要求にあわせてシステムを特殊化したいという要求も高まる．また，数式処理システムの普及を妨げている要因の一つとして，研究者の広範な要求に応えるだけの数学知識が，数式処理システム上に集積されていないことがあげられる．

ユーザが独自に数学知識を数式処理システム内に書き込もうとするとき，本章（3）の（a），（b）のレベルでは書き切れぬ場合があり，大型システムの場合，そこで挫折してしまう．しかし，muMATH のような小型システムでは（c），（d）のレベルでの書き込みが容易であり，システムを特殊化しやすい．4.2 節で述べる ICAS も小型システムの特殊化の一例である．

（6） 豊富な入出力

現状のパソコンは，マウス，タブレットなどの入力装置やグラフィックディスプレイ，プロッタやプロッタプリンタ，カラープリンタなどの出力装置を持ち，大型機の標準端末に比較して，多彩な入出力機能が実現されている．

これを生かして，数式処理システムに種々の付加価値をつけ得るが，OS の公開されているパソコンにはその種の開発を効率良く行える長所がある．パソコン上の小型システムはこの種の機能を強化する方向で，存在意義を主張できる可能性がある．これについて，5 章で述べる数式処理ワークステーションの試みを参照されたい．

4. 小型数式処理システムの記述

4.1 Prolog

2 章でも述べたように，Prolog を用いて記述された本格的システムとして，エジンバラ大の PRESS がある[16]．一方，我が国で作られた小型システムとしては，愛媛大[11]，神戸大[12]のシステムとわれわれの CASP[13] がある．数式処理システムは，LISP または，LISP 系言語で記述されるのが普通であり，あえて prolog を用いて記述する理由が問われる．これに関しては，文献 17) に詳しい．

prolog で記述する利点として

①知識をモジュール化された形で並列に並べることで，式の簡素化，微分など数式処理のアルゴリスムを記述できる．

②prolog の持つパターンマッチング機能は，本来，数式処理になじみやすい．

③システムがバックトラックを行うので，制御的な記述をほとんどしなくても済む．

などがあげられる．

①の特徴は，3 章の(3)で述べたシステムの拡張(ルールの付加) が容易であることを意味し，特に小型システムに対して prolog を用いる利点と言える．

②も，記述されたシステムの理解が容易であることを意味し，小型システムに望まれる特徴である．

しかし，システムの規模が大きくなると制御的なルールが増加し，①，③の特徴はそのままでは成立しなくなる．また，prolog における制御的なルールは，必ずしも記述しやすくなく，下手に記述すると，探索空間の規模が大きくなるし，バックトラックの頻度にも影響する．

小型システムを prolog で記述することは比較的容易であるが，それを大規模化してゆく技術には相当な困難が伴うと言える．筆者は CP/M 68 K 上に，CASP と呼ばれる prolog で記述したシステムを開発したが，muMATH の持つ機能を特定のパッケジに分割して再現することはできたが，それらを一体化した大きなパッケジは，2 M バイトのメモリでは有効に動作しなかった．

現状では，prolog で記述する小型数式処理システムは，汎用システムではなく，prolog の特徴を利用した個別の特殊システムを目指すものであろう．

4.2 ICAS

筆者らは，muMATH の上にオペレタ，関数，独立変数を独立した内部構造として持つ数式処理機能 ICAS を作成し，さらにその上に量子力学的散乱理論におけるオペレタ，変数を扱うパッケジ OAP を構築した[18]．ICAS は 1) オペレタを前置型で扱える，2) オペレタ代数が行える，3) オペレタ，関数を含む整式を任意のレベルでその評価を抑止して表示できるなどの特徴を持ち，この分野の研究者が従来，紙と鉛筆で行ってきた種々の操作が，数式処理システム上で行える．図-5 に上記の機能の一端を示すが，詳細な内容については文献に依られたい．

さらに，これらの機能を十分に利用するため，muMATH に LET，FOR，係数生成，SHOWTIME などの大型システム上でのみ利用できる機能を付加し，実用性を高めた．また，MAKERULE と呼ばれる対話的ルール登録機能を作成し，従来は 3 章の(3)の property，組込関数の変更によらねばシステムに組み込めなかった，構造的なルールをパターンマッチン

```
0 ? DEFINE U(x), x^3, ENDDEF;
    Function U(x) accepted
@:  U(x)
1 ? DEFOP Dx, DIF (#EXP, x), ENDOP;
@:  Dx
4 ? OP: (Dx+A)^3;
@:  (A+Dx)^3
time: 0 sec
5 ? OP. U(x);
@:  6+18Ax+9A^2x^2+A^3x^3
time: 1 sec
6 ? DISABLE (U) $
6 ? OP. U(x);
@: 3AUxx(x)+3A^2Ux(x)+A^3U(x)+Uxxx(x)
0 ? (Dx+A)^3. U(x);
@: 6+18Ax+9A^2x^2+A^3x^3
```

図-5 オペレタの取り扱い

われわれの開発した ICAS 機能におけるオペレタの取り扱いについて述べる．0 は関数 $U(x)=X^3$ の定義，1 はオペレタ (d/dx) の定義，4 は $O=(d/dx+A)^3$ なるオペレタ O の生成，5 は O の作用例，6 は関数 $U(x)$ の評価の抑止の宣言．そして $OU(x)$ を求めると微係数 $Uxxx$，Uxx，Ux を用いた出力を得る．0 は直接オペレタを前置して作用させた例で 5 と同じ結果を得た 4，5 では所要時間を表示させた．

グ技法を用いて，対話レベルで登録できるようにした．(現時点で，ICAS，OAP は約 100 K バイトのサイズを占める)．

このようにして開発した ICAS，OAP は理論物理学，数学，回路理論，CAI などを専攻する 20 名以上の muMATH を利用する研究者をユーザとして獲得し筆者を中心とした組織により保守，運営している．この体験を通じて，3 章に述べた小型システムの特徴を積極的に生かせば，種々の分野において，小型システムなりの独自の発展が期待できると考えている．

4.3 その他の言語による記述

Cで記述されたシステムとして SMP があるが，これはスモールシステムとは呼べない[19]．

Cで記述することにより，LISP で記述するよりは高速性が得られるが，一方ではすでに公開された LISP のアルゴリスムの遺産を利用したいという事情がある．われわれの体験でも，Cの上に LISP の関数と等価な関数を構築し，その上で数式処理システムの記述を行うことは容易にできるが，それだとニモニクで記述した LISP 上の数式処理システムよりも遅く高速性を獲得することにつながらない．パソコン上のCで開発したシステムを最終的に，メインフレームか 68000 や 68020 上のCに移殖するなら，上記のやり方もそれなりのメリットがある．

むしろ，Cの持つポータビリティの良さを生かして，機種の異なる種々のパソコン上で，同じ小型数式処理システムを稼動させることが有望であり，Cの役割もその辺にある．

汎用システムを目指さない特殊な用途の専用システムであれば，LISP で記述するのとは異なったCの特徴を生かしたアルゴリスムを採用することは可能である．

また，FORTH を用いて一層の高速性を得ることは可能であるが，FORTH を用いた本格的な数式処理システムの構築の例を，筆者は知らない．

PASCAL に関しては，数値計算との複合利用が可能である点は興味深いが，記述の冗長性が気になる．教育的な課題として数式処理システムを作成させる場合を別にすれば，Cで記述する方が妥当性が高い．

5. 展 望

小型数式処理システムが独自の発展をしてゆくためには，3 章で述べた特徴を伸ばしてゆくことが必要だが，ここでは，ハード，ソフト面のヒューマンインタフェースの改良について具体的に述べたい．

(A) ハードウェア面での改良

ハードウェアの側面よりの改良として，キーボードのみの入力方式からの脱却，グラフィックディスプレイを用いた柔軟な入出力の表示と，それらのハードコピー出力などがある．

最近の高レベルパソコンは，マウス，マルチウインドウ機能，高度のグラフィック機能を持ち，上記の数式処理システムの改良を行うだけのハードウェアの成熟が実現している．

筆者の研究室で稼動しているパソコンを利用した数式処理専用システムを紹介し，ハードウェア改良の概念を具体的に説明したい．本システムは，8086 を CPU とする高位パソコン (NEC 9801 VM 2) と 2 M バイトのメモリを持つ 68000 システムの複合システムであり 2 種のディスクドライブと，20 Mバイトの容量のハードディスクを装備している．また，マウスを入力装置として持つ．8086 上の MS-DOS で muMATH が利用できさらにその上で ICAS を利用できる．CP/M 68 K 上で prolog で記述されたわれわれの研究室で開発した CASP，独自に開発した OECU-LISP で記述した ICAS 68 が稼動している．さらに CP/M 68 K 上で REDUCE 3.2 を利用している．

また，MS-DOS 上で稼動するCで記述した小型数式処理システムを，マウスとアイコンとキーボードを使用して利用する，「数式処理ワークステーション」を開発中である．これは，上述の入力，出力におけるヒューマンインタフェースを改善する試みである．整式の取り扱いのみならず，数学データベースよりの入力，数学データベースへの出力，グラフィック画面の利用，整式スタックへの書き込み，数式処理システム上からの OS コマンド実行などをマウスとアイコンにより制御し，操作性の著しい改善をはかることを試みている．

(B) ソフトウェア面での改良

数式処理システムのソフトウェア面でのヒューマンインタフェースの改善については，いくつかの提言がなされている (本特集の永田「数式処理技術の利用法」によりその概容がつかめる)．

①研究者に対する問題解決のサポート，②入出力時における外部手作業のような副次的作業の軽減．③システムの思考過程の明示機能，④教育，CAI への応用．⑤HELP 機能の強化，などがヒューマンインタフェースに絡んで重要である．

そのうちで，多人数教育における④の利用は，小型システムにとり有望である．教育用の小型システムはいくつか作成されているが，実践的研究は今後の課題と思われる．これに関して筆者らは，大阪電気通信大学情報処理教育センターに設置した48台のパソコンを連結したLAN上での数学教育の小型システムによる実践を準備している[20]．その過程で，学習者の数式処理システムの利用に関する誤まりを認知するプロダクションシステム「SMDIAG」[21]を作成したが，この他にも解決すべき多くの課題があると考える．

最後に，小型数式処理システムの進展は，より本格的な数式処理の高速アルゴリズムの研究[22]，大型システムの開発研究[23]に支えられて，可能となることに注意したい．

小型システムの限界を超えたところに，多くの応用研究上の問題解決がある．したがって，小型システムと大型システムの併用，複合的利用が必要となるケースは多い．大型システムが，大型コンピュータでもパソコンでも利用できるようになってこの問題が解決されるのか，それとも筆者が本解説で述べたように，大型システムと小型システムはおのおの独自の発展をしてゆくのか，予測し難い面がある．

参考文献

1) Buchberger, B. (ed.): Computer Algebra Symbolic and Algebraic Computation, pp. 221-224 (1982).
2) 山本，戸島，村田：LISP 68 K の開発と RED. UCE の移植，bit, Vol. 7, No. 12, pp. 64-82 (1985).
3) Rich, A. D. and Stoutemyer, D. R.: Capabilities of the MUMATH-79 Computer Algebra System for the Intel-8080 Microprocessor, Proceedings of EUROSAM '79 (Lecture Note in Computer Science 72), Springer-Verlag (1979).
4) The muMATH-79 Symbolic Mathematics System Reference Manual, the SOFT WARE-HOUSE, (1979).
5) The muMATH-83 Symbolic Mathematics System Reference Manual, The SOFT WARE-HOUSE, (1983).
6) 対馬：数式処理システム OECU-MATH の作成，大阪電通大科学論集 (1982).
7) 対馬：数式処理システム OECU-MATH の拡張，大阪電通大科学論集 (1983).
8) 広田：「もっと数式処理を使おう」，物理学会秋の分科会特別講演 (1985).
9) 増田：BASIC による代数計算の数式処理用のサブルーチン・パッケージ，数理解析研究録 551, pp. 136-144 (1985).
10) Rall, L. B: Differentiation in PASCAL-SC (Type GRADIENT), ACM Trans. Math. Softw. pp. 161-184 (1984).
11) 野田，戒能：PROLOG による小型数式処理システムと数値計算，数理解析講究録 551, pp. 158-172 (1985).
12) 村上他：数式処理システムを用いた新しい数学教育，教育工学関連学協会連合全国大会 (1985).
13) 対馬：数式処理システムにおける知的対話環境の改善，数理解析研究録 551, pp. 116-135 (1985)
対馬：Prolog 上の数式処理システム CASP，ソフトウェアコンフェレンスプロシーディングス (1985).
14) これらの情報に関しては雑誌『数式処理通信』が詳しい．
15) 増永他：GAL における数学データベースについて，情報処理学会第28回全国大会 (1984).
16) Bundy, A and Welham, B.: Using Meta-level Inference for Selective Application of Multiple Rewrite Rule Sets in Algebraic Manipulation, Artificial Intelligence, Vol. 16, No. 3, pp. 189-212 (1981).
17) 永田，数式処理における Prolog，情報処理，Vol. 25, No. 12, pp. 1404-1509 (Dec. 1984).
18) 対馬，佐藤：知的数式処理システム ICAS の開発，(数理解析研『数式処理と数学研究への応用』研究集会) (1985).
19) Cole, C. A., and Wolfram, S.: SMP-A Symbolic Manipulation Program. SYMSAC, 20-22 (1981).
20) 対馬：「対話型情報処理教育のパソコン LAN への展開」，第8回私情協シンポジウム (1985).
21) 対馬，阪野：数式処理テストにおける誤答診断システム SMDIAG の作成，CAI 学会誌 Vol. 4 No. 1, pp. 5-13 (1984).
22) 佐々木：数式処理，情報処理学会 (1980).
23) 米国は REDUCE，MACSYMA などを生み数式処理に大きな影響を与えた．我が国でも GAL のような独自のシステムが生まれつつある．これらの成果はやがて小型システムにも反映されるだろう．

(昭和60年12月2日受付)