新しい経済分析システムの提唱

大　野　拓　行

1. はじめに

パーソナル・コンピューターの普及により，計算機を用いた経済学の研究や教育が身近なものになり，いままで大型計算機に頼ってきたものが研究室で出来るようになってきた。パーソナル・コンピューターの利点を簡単に要約すれば，導入期にある程度の投資さえすれば，いつでも，身近に，自由な形で利用できることであろう。研究面では，分析に用いる文献や資料は研究室にある場合が多いので，分析材料があるところで計算機が利用できることになり研究の効率性が上がる。さらに，研究には試行錯誤を伴うことが多いが，研究室に計算機があることによって，他人に迷惑をかけずに自分が納得できるまで試行を繰り返すことができるという利点も生じてくる。統計学や計量経済学の教育面でもパーソナル・コンピューターを利用すれば単に考え方や手法を教えるだけでなく，いろいろな手法のアルゴリズムを身近なものとして教えることが可能になる。さらに，現実のデータを用いた応用例を手軽に作成でき，しかもそれを視覚的に示すことも可能となる。

パーソナル・コンピューターを研究や教育に利用しようとする場合には目的にあったソフト（プログラム）が必要である。ソフトは自分で作成することも，既存のパッケージソフトを利用することも可能であるが，それぞれに長所と短所を持ち合わせている。自分でプログラムを書けば目的に最もフィットしたソフトを作成できる半面，プログラムを書いたり，テストしたりするのに多くの時間を費やす必要がある。一方，パーソナル・コンピューターのハード面での普及に伴い次第に普及してきた経済分析用のパッケージソフトを利用するとソ

フトを開発するのに時間をかけなくて済むという利点がある。しかし，その半面で研究や教育が使用するパッケージソフトの内容に大きく制限され画一的なものになりがちであるという欠点がある。

研究教育者はあまり時間をかけずに目的に合ったソフトの利用を望んでいるのであるが，一般的には目的が特殊になるほどソフトの開発に時間を費やすことが必要とされる。しかしながら，経済分析の場合，種々の分析手法に使用される基本的演算は共通なものが多く，個々の分析手法は基本的演算の組み合わせが異なるだけといった場合が多い。そこで，このような経済分析に特有な性質を利用すれば，汎用性があり，しかも個性的な研究や教育にも対応可能なシステムを構築できると思うのである。以下の諸節で筆者が提唱する経済分析システムの基本的な考え方を例示を含めながら述べていきたいと思う。

2. 基本的な考え方

一般的なプログラム作成方法はソースプログラムをFORTRANやBASICで書き，それをコンパイル，リンクして実行形式ファイルを作成する。ある目的のためにプログラムを自分で作成する場合には，最初からソースプログラムを書くか，過去に書いたソースプログラムを修正することになる。また，目的に合わせて他人が作成したプログラムやパッケージソフトを修正しようとする場合にもソースプログラムを修正するのが一般的である。しかし，自分が過去に書いたソースプログラムの修正でさえ，プログラムが長くなると困難な場合が多くある。まして，他人のプログラムやパッケージソフトを修正して利用しようとする場合にはかなりの忍耐と努力が要求されるであろう。

書き方を工夫することによって，読みやすく，修正しやすいプログラムを書くことは可能であるがそれにも限度がある。例えば，サブルーチンを多用してメインプログラムをすっきりさせることは可能であるが，そのようなプログラムを修正する場合にも使用されている変数や配列の大きさに注意する必要があるし，サブルーチンプログラムだけを完全に独立した形で使用することはできない。このような意味で，サブルーチンプログラムは独立しているようである

がプログラムの他の部分と密接に関連しているのである。これらのことを経済学における代表的分析である回帰分析を例にとって説明してみよう。

回帰分析における回帰係数を計算する過程を読みやすくプログラムするには，行列の転置，積，逆行列のサブルーチンプログラムを作成し，メインプログラムでは計算順序に従って各サブルーチンを呼び出すようにするのが一般的であろう。次に，このプログラムを利用して産業連関分析の産出水準決定のプログラムを作成する場合を考えて見よう。

この時，回帰分析のプログラムを参考にしようとする研究者は，まずソースプログラムの全体を見て，使用されている変数，配列，サブルーチンプログラムなどを点検する必要がある。サブルーチンプログラムを使用している場合には，個々のサブルーチンは変更しなくてもよいかもしれないが，メインプログラムにおける配列宣言などを修正する必要がある。このような作業も，この例のように，単純な場合は簡単であるがプログラムが大きくなりCOMMON文などを多用している場合には複雑になるであろう。

さらに，単に逆行列を計算したい場合を考えてみると（経済分析ではこのような単純な行列演算を必要とする場合がかなりある），回帰分析のプログラムを修正するより，はじめからプログラムを書いた方が早く目的とするプログラムが作成できるかも知れない。

このようにある特定の分析を1つのプログラムでしようとすると，個々のプログラムに重複する箇所が出てくるし，新たな計算が必要になった場合には常にソースプログラムのレベルで修正が必要となってくる。そこで，考え方を変え，逆行列の計算などの経済分析によく使用される計算を完全に独立したコマンド（以下で詳しく述べる）として作成し，特定の分析は基本的演算コマンドを組み合わせて行うようなシステムを構築すれば，今述べたような欠点を解消し，かつ，新たな利点が生じてくると考えられる。

現在のパッケージソフトの傾向は少数のプログラムにより多くの機能を持たせようとするものであるが，筆者が提唱しようとするシステムは，このような傾向とは異なり，ブロック玩具のようなもので，ある特定の機能しか持たない

小さなコマンド（部品）を組み合わせて目的となる作業を行おうとするものである。このことを最小二乗法を例に述べてみよう。

回帰係数の最小二乗推定量は

$$\hat{\beta} = (X'X)^{-1}X'Y \qquad (1)$$

で与えられ，これは行列の転置，積，および逆行列という3つの基本的演算の組み合わせである。いま，逆行列の計算だけを行う完全に独立したプログラムを作成し，コンパイル，リンク後の実行形式ファイルにMATINVという名前を付けるとしよう。MATINVを完全に独立したコマンドとするためには，そのソースプログラムに，逆行列を計算する機能だけでなく，行列データの入力機能と演算結果の出力機能を含ませる必要がある。逆行列をある特定の目的のため使用しようとする研究教育者は逆行列の計算の細部まで知る必要はなく，そのプログラムの使用法が解れば十分である場合が多い。そこで，ここではソースプログラムの細部には立ち入らないで，その使用法だけを述べることにする。（ソースプログラムについては後の節で述べる）

MATINVの使用法は簡単で，

MATINV　A　B　　(2)

である。機能は，ファイルAの行列データを読み込んで逆行列を計算して，その結果をファイルBに出力するだけである。実際の実行例を図1に示す。このようにMATINVはMS-DOS[1]付属の外部コマンドのFORMATやDISKCOPYと同様に利用できるのである。(2)式においてMATINVがコマンドで，AとBがそのパラメーターである。個々のコマンドをMS-DOSの外部コマンドとして利用できるのがこのシステムの第1の利点である。

逆行列と同様に行列の転置と積のコマンドを作成する。それぞれの使用法は，

MATTRN　A　B　　(3)

MATPRO　A　B　C　　(4)

---

(1) MS-DOSは米Microsoft社の商標である。MS-DOSには，システム自体に含まれているコマンド（内部コマンド）とコマンドの実行形式のファイルが必要なコマンド（外部コマンド）とがある。

図1 逆行列計算コマンドMATINVの使用例

$A = \begin{pmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{pmatrix}$ として A^{-1} の計算を実行する。

ファイルAの内容表示

```
A > type A
4 4
5  7  6  5
7 10  8  7
6  8 10  9
5  7  9 10
```

(注) typeはファイルの内容を表示させるMS-DOS付属コマンド。最初の4は行数，次の4は列数を表し，それ以後が行列Aの内容。行列データの記述法は図4参照。

MATINVの実行

```
A > MATINV A B
```

ファイルBの内容表示

```
A > type B
4 4
6.800000000000E+01 -4.100000000000E+01 -1.700000000000E+01 1.000000000000E+01
-4.100000000000E+01 2.500000000000E+01 1.000000000000E+01 -6.000000000000E+00
-1.700000000000E+01 1.000000000000E+01 5.000000000000E+00 -3.000000000000E+00
1.000000000000E+01 -6.000000000000E+00 -3.000000000000E+00 2.000000000000E+00
```

である。(3)式は行列の転置のコマンドでAの行列を読み込んで，その転置行列をファイルBに出力する。(4)式は行列の積のコマンドでAとBの積を計算し，結果をファイルCに出力する。(2), (3), (4)のコマンドを組み合わせることによって(1)式は以下のように記述できる。

```
MATTRN  X      X'                    (5)
MATPRO  X'     X      X'X            (6)
MATINV  X'X    IX'X                  (7)
MATPRO  X'     Y      X'Y            (8)
MATPRO  IX'X   X'Y    B              (9)
```

(5)から(9)までコマンドを順番に実行することによってファイルBに回帰係数が出力されることになる。このようなシステムを用いることによって以下のような利点が生じてくる。

a．個々のコマンドを単独にMS-DOSの外部コマンドとして使用できる。

b．コマンドの組み合わせ方法は利用者の自由である。

c．利用者は個々のコマンドの内部構造に注意する必要はない。他人が作成したコマンドでも計算が正しく，入出力の形式が統一しておれば自由に利用可能である。

d．作業内容が簡潔に記述できる。このことは研究面でも教育面でも便利である。

e．作業の途中結果がファイルに保存されるので，計算のチェックや途中結果を用いた別の分析が簡単にできる。不要のファイルは最後にMS-DOS付属コマンドで消去すればよい。

f．プログラムの作成や修正がコマンド単位なので，保守や機能拡大が簡単にできる。

g．新しい作業に対する対応がコマンドの組替えや少数の新規コマンドの作成で済み汎用性がある。過去に作成したコマンドの蓄積を生かすことができる。

h．作業は個々の小さなコマンドを連続的に実行することにより為されるためCPUのメモリーが少なくても大きな作業をすることが可能である。

一方，このシステムの一番の欠点は，コマンドを連続的に実行したり，常にファイルに対して入出力を繰り返すことによる計算時間の長さである。しかし，最近急速に普及してきたRAMディスクやハードディスクを利用すればこの問題は解消することができるであろう。

3．バッチファイル(2)の利用

MS-DOSのバッチファイルを利用することにより定型的な作業を1つのコマンドとして実行することが可能となる。さらにMS-DOS付属コマンドを利

(2) バッチファイルとは，そのファイルを呼び出すとファイルに記述されているコマンドを順番に実行していくような特別なファイルである。他のファイルとはファイル拡張子BATによって区別される。

リスト1 最小二乗法 OLS.BAT

```
ECHO OFF
CLS
-------------------------------------------------------------------------------
  OLS.BAT  最小二乗法

                使用法  OLS %1 %2
                        %1 説明変数行列
                        %2 被説明変数ベクトル

    出力 X'X.@  積率行列              IX'X.@ 積率行列の逆行列
         B.@    係数推定値ベクトル    E.@    残差ベクトル
         VAR.@  分散共分散行列
-------------------------------------------------------------------------------

 ファイルチェック

OLS$ %1 %2
IF ERRORLEVEL 1 GOTO END

ECHO 最小二乗法
ECHO ON

  係数推定値を求める。

MATTRN      %1          X'.@
MATPRO      X'.@        %1          X'X.@
MATINV      X'X.@       IX'X.@
MATPRO      X'.@        %2          X'Y.@
MATPRO      IX'X.@      X'Y.@       B.@

  分散共分散行列の計算

MATPRO      %1          B.@         XB.@
MATADD      %2          XB.@        E.@          —
```

```
40 :
41 : VARCOR        E @          IX'X @        VAR @
42 :
43 :   一時的ファイルの消去
44 :
45 : FOR %%X IN(X' X'Y XB) DO DEL %%X @
46 :
47 :   END
```

用することにより複雑な作業も記述可能となる。

バッチファイルの例として前節の最小二乗法を考えてみよう。回帰係数を計算する度に(5)式から(9)式のコマンドをキーボードから入力するのは大変であり，入力ミスによる計算間違いも起こりがちである。そこで，リスト1に示すようなバッチファイルOLS.BATを作成することによって，単に

OLS　X　Y　　(10)

と入力することによって回帰係数と係数の分散共分散行列を計算することができる。OLS.BATの内容を詳しく見てゆこう。このバッチファイルにおいて使用されているMS-DOS付属コマンド(ECHO, CLS, IF, FOR)以外のコマンドはOLS$, MATTRN, MATPRO, MATINV, MATADD, VARCORの6つである。MATTRN, MATPRO, MATINVはそれぞれ行列の転置，積，逆行列のコマンドで前節でその使用法と機能は説明したので，ここでは残りの3つについてその使用法と機能を説明していく。

OLS$: OLS.BATのためのファイルチェック

使用法　OLS$　X　Y

X　説明変数行列のファイル名

Y　被説明変数ベクトルのファイル名

機能　ファイルX, Yの存在のチェックとX, Yの行数，列数が最小二乗法を適用するのに適切かどうかをチェックする。このコマンドはOLS.BATに特有なもので汎用性はない。

MATADD：行列の加減演算

使用法　加算　MATADD　A　B　C　+

減算　MATADD　A　B　C　−

機能　行列AとBの加算あるいは減算を行い，結果をファイルCに出力する。

VARCOR：係数推定値の分散共分散行列の計算

使用法　VARCOR　A　B　C

A　残差ベクトル

B　積率行列の逆行列

機能　残差ベクトルAと積率行列の逆行列Bを用いて分散共分散行列を計算し，結果をファイルCに出力する。

OLS.BAT の内容の説明の前にバッチファイルの引数について説明をしておく。バッチファイルの実行時に必要な値はパラメーターとして与えることができる。(10)式においてOLSがバッチコマンドで，XとYがOLSに対するパラメーターとなる。[3] バッチファイル内ではパラメーターは％0～％9で表し，コマンド実行時にパラメーターと置き換えられる。例えば，(10)式のような使い方をするとOLS.BATにおける％1と％2が自動的にXとYに置き換えられて実行される。

OLS.BATの1行目と2行目はMS-DOS付属コマンドの実行である。1行目のECHO OFFは以下ECHO ONが表れるまで画面表示を中止する命令であり，2行目のCLSは画面消去の命令である。3行目から16行目までは注釈であり実行に関係しない。17行目でファイルチェックコマンドOLS$を実行し，もしエラーがあれば18行目で47行目のENDに飛んで作業を終わる。20行目は単に"最小二乗法"という文字列を画面に表示させるMS-DOS付属コ

(3) 使用法から見ると，(10)式のバッチコマンドは(2)式の一般コマンドと同じであるがファイル拡張子が一般コマンドの場合EXEかCOMなのに対してバッチコマンドはBATである点が異なっている。さらに重要な違いは一般コマンドがその内容を簡単に見ることができないのに反して，バッチコマンドは一般コマンドを特定の順番に実行するコマンドでその内容を内部コマンドtype，copy等で容易に見ることができる点である。

図2　OLS.BAT の使用例

付表に示した統計データを用いて，次式を計測する。

LN(GNP) = b0+b1*LN(K)+b2*LN(L)　　(i)

説明変数行列，被説明変数ベクトルをそれぞれ，X，Yとすると，それぞれのファイルの内容は以下のようになる。データの記述法は図4を参照のこと。

A > type X

```
20 3
1  11.196  8.477
1  11.264  8.494
1  11.355  8.506
1  11.458  8.522
1  11.587  8.536
1  11.713  8.525
1  11.806  8.553
1  11.891  8.554
1  11.991  8.541
1  12.072  8.556
1  12.143  8.562
1  12.208  8.575
1  12.265  8.585
1  12.326  8.599
1  12.393  8.610
1  12.459  8.622
1  12.519  8.636
1  12.575  8.641
1  12.627  8.650
1  12.686  8.657
```

A > type Y

```
20 1
11.535
11.637
11.758
11.871
11.944
11.993
12.082
12.126
12.122
12.159
12.204
12.256
12.307
12.358
12.397
12.430
12.462
12.498
12.546
12.587
```

ファイルXとYに対して最小二乗法を適用するには

A > OLS X Y

と入力すればよい。係数推定値はファイル B.@に保存されている。その内容は，

A > type B.@

```
3 1
-4.261314713396E+00 4.980334117863E-01 1.217637737747E+00
```

となっているので，b0 = −4.261，b1 = 0.498，b2 = 1.218 であることがわかる。また，推定値の分散共分散行列は，

A > type VAR.@

```
3 3
2.526702197474E+01 3.768484106729E-01 -3.477131149497E+00 3.768484106734E-01
6.124045686370E-03 -5.256684277569E-02 -3.477131149499E+00 -5.256684277562E-02
4.794997459184E-01
```

となっている。

マンドである。21行目で画面表示を復活させる。23行目は注釈である。25行目から33行目で回帰係数を計算している。ここはファイル名が異なるだけで(5)式から(9)式までと同じである。37, 39行目で残差($\hat{\varepsilon} = Y - X\hat{\beta}$)の計算をして，41行目で分散共分散計算コマンドVARCORを実行している。45行目は不要ファイルX'.@，X'Y.@，XB.@を消去するMS-DOS付属コマンドの実行である。結局，このバッチファイルの本質的な部分は25, 27, 29, 31, 33, 37, 39, 41の8行にすぎないのである。このバッチファイルを実行することによって，10行目から12行目に記述したような結果ファイルを得ることができる。OLS.BATによるコブ・ダグラス生産関数の計測例を図2に示しておく。

このようにバッチファイルを作成することにより入力ミスによる計算間違いをなくすることができる。さらに，リスト1のように，作業内容を簡潔に記述できるためバッチファイルをプログラムファイルとドキュメントファイルを兼用したものとして利用することが可能となる。

次に，このシステムの汎用性を示す例をあげてみよう。回帰分析を行う場合，経済理論からの要請で回帰係数の間にある種の制約を置いて推定を行いたい場合がよくある。その目的のために利用される手法は線型制約付最小二乗法であるが，研究者の利用可能なパッケージソフトにそれがない場合には自分でプログラムを組むか，その手法の使用を諦めるしかない。FORTRANなどに慣れている研究者は別として，自分でプログラムを組むのは大変であるが筆者の提唱するシステムを利用すれば数式どうり作業手順を記述するだけで簡単に目的とする手法が利用可能となる。

線型制約付最小二乗法の係数推定量とその分散共分散行列は以下の式で与えられる。

$$\tilde{\beta} = \hat{\beta} - (X'X)^{-1}A'[A(X'X)^{-1}A']^{-1}(A\hat{\beta} - r) \tag{11}$$

$$\mathrm{Var}(\tilde{\beta}) = \Sigma - \Sigma A'(A\Sigma A')^{-1}A\Sigma \tag{12}$$

ここで，$\hat{\beta}$: 制約なしの最小二乗推定量

Σ: $\hat{\beta}$ の分散共分散行列

A, r: 線型制約 $A\beta = r$ におけるA行列と r ベクトル

リスト2　制約付最小二乗法 LCOLS.BAT

```
ECHO OFF
CLS
:----------------------------------------------------------------------
:  LCOLS.BAT  線形制約付き最小二乗法
:
:    使用法  LCOLS %1 %2 %3 %4
:                          %1 : 説明変数行列
:                          %2 : 被説明変数ベクトル
:                      %3, %4 : 線形制約 Ab = r における A 行列と r ベクトル
:
:    出力 : B.@        無制約の場合の係数推定値ベクトル
:           E.@        無制約の場合の残差ベクトル
:           VAR.@      無制約の場合の分散共分散行列
:           LB.@       線形制約の場合の係数推定値ベクトル
:           LE.@       線形制約の場合の残差ベクトル
:           LVAR.@     線形制約の場合の分散共分散行列
:----------------------------------------------------------------------
:
: ファイルチェック
:
LCOLS$ %1 %2 %3 %4
IF ERRORLEVEL 1 GOTO END
:
ECHO 制約付き最小二乗法
ECHO ON
:
: OLS.BAT を呼び出す。
:
COMMAND/C                OLS          %1           %2
DEL X'X.@
:
: 係数推定値を求める。
:
MATTRN       %3          A'.@
:
MATPRO       IX'X.@      A'.@         XA.@
:
MATPRO       %3          XA.@         AXA.@
:
```

```
40 MATINV      AXA @      IAXA @
41
42 MATPRO      %3         B @        AB @
43
44 MATADD      AB @       %4         AB_B @     —
45
46 MATPRO      XA @       IAXA @     T @
47
48 MATPRO      T @        AB_B @     T1 @
49
50 MATADD      B @        T1 @       LB @       —
51
52 : 残差ベクトルの計算
53:
54 MATPRO      %1         LB @       XLB @
55
56 MATADD      %2         XLB @      LE @       —
57
58 : 分散共分散行列の計算。
59
60 MATPRO      %3         VAR @      AS @
61
62 MATPRO      T @        AS @       TAS @
63:
64 MATADD      VAR @      TAS @      LVAR @     —
65
66 : 一時的ファイルの消去
67
68 FOR %%x IN(A' IX'X XA AXA IAXA AB AB_B T T1 XLB AS TAS) DO
   DEL %%x @
69
70 END
```

(11), (12)式を計算するバッチファイルLCOLS.BATはリスト2のようになる。LCOLS.BATにおいて新たに作成したコマンドはLCOLS$だけであり，これも先のOLS.BATにおけるOLS$と同様なもので作業の本質的な部分ではない。LCOLS.BATにおいて注意すべき所は29行目である。(11), (12)式を計算するためには前もって制約なしのOLS推定値とその分散共分散行列を得て

図3　LCOLS.BAT の使用例

コブ・ダグラス生産関数に1次同次を課すことを図2の(ⅰ)式の係数で表現すると，

$$b1+b2=1 \qquad (ii)$$

となり，(ⅱ)式を行列表示 $A\beta=r$ で表現すると，

$$A=(0\ 1\ 1),\ r=1$$

となる。よって，ファイル A, r の内容は，

```
A > type A
  1 3
  0 1 1
B > type r
  1 1
  1
```

となる。(ⅰ)式を(ⅱ)式の制約の下で推定するには，

```
A > LCOLS X Y A r
```

と入力すればよい。係数推定値はファイル LB.@に保存されている。その内容は，

```
A > type LB.@
3 1
1.570066645203E+00 5.853885609784E-01 4.146114390222E-01
```

となっていて，b1+b2=1 が成立していることがわかる。

おく必要があるので，ここでバッチコマンド OLS を呼び出している。他の一般コマンドと異なり，バッチファイルの中で他のバッチコマンドを呼び出す場合には，このように

```
COMMAND/C　バッチコマンド
```　(13)

とする必要がある。LCOLS.BAT の他の内容はコマンドの組み合わせ方とファイル名が異なるだけなので作業内容の把握は簡単にできると思う。先のコブ・ダグラス生産関数に同次性を課した場合の計測例を図3に示す。

このように，このシステムでは過去に作成したコマンドやバッチファイルの蓄積を利用して，新しい要請に最小の努力で対応することが可能となるのである。研究教育者はこのような基本演算のコマンドをより多く集めれば集める程，多くの局面に対応できることになる。なお，ここでは行列演算を用いた例をあげたが，基本的演算コマンドは行列演算に限られないことは明白である。このシステムでは入出力のデータの形式が統一しておれば個々のコマンドの内容は

どのようなものでもよいのである。

経済分析では収束判定を含んだ繰り返し計算を行う場合もよくある。そこで，最後に，このような場合を考えてみよう。いま，次のような，3つのバッチファイルと1つのコマンドがあるとしよう。

INT.BAT：初期値計算の作業内容を記述したバッチファイル。初期値はファイルAに出力されるものとする。

REPEAT.BAT：1回の繰り返し計算においてなされる作業内容を記述したバッチファイル。計算結果はファイルBに出力されるものとする。

CHECK.EXE：収束判定用のコマンド。使用法はCHECK A B Cで，AとBは比較するファイルで，Cは収束判定基準値である。機能は基準値Cによって A と B の内容を比較し，収束していたらERRORLEVEL 0を返し，収束していなかったらERRORLEVEL 1を返す。

CHANGE.BAT：収束しなかった場合，次の繰り返し計算に入る前になされるパラメーターの変更などの作業を記述したバッチファイル。

以上のような部品を揃えれば，リスト3に示したようなバッチファイルで目的とする作業は記述できる。バッチファイルの内容を説明すると，1行目で初期値を計算し，3行目で最初の繰り返し計算を実行する。そして，7行目で収束判定をして収束していると判定されたら19行目に飛んで作業を終わる。もし収束しなかったら，13行目に飛び，15行目で変更作業を施し，17行目で13行目に返る。

この例で示したように，このシステムではコマンド，バッチファイル，MS-DOS付属コマンドを組み合わせることによって複雑な作業も記述可能となるのである。

リスト3　収束判定を含むバッチファイルの例

```
1  INT
2
3    START
4
5  REPEAT
6
7  CHECK A B 0.01
8
9  IF ERRORLEVEL 1 GOTO LOOP
10
11 GOTO END
12
13   LOOP
14
15 CHANGE
16
17 GOTO START
18
19   END
```

4. 基本的演算のプログラムの作成

ブロック玩具で遊ぶ子供が個々のブロックの内部構造を知らなくてもいいように，このシステムの利用者が全て個々の基本的演算コマンドの内部構造を知る必要はない。このことは我々が何の疑問もなくMS-DOS付属コマンドを使用していることでも解る。しかし，他人の作成したコマンドだけでは目的とする作業ができない場合もあるし，プログラムが書ける研究者は自分でコマンドを作成したい場合もある。そこで，この節では筆者が作成したコマンドを例にプログラムの作成方法を述べることにする。

このシステムにおける基本的特質は入出力におけるデータ構造の統一性である。これはブロック玩具において，接続用の穴の大きさが統一されているのと同じである。データ構造が同一であればコマンドの内容がどんなものであろうとシステムの他のコマンドと一緒に使用することが可能である。データ構造の

一例として筆者が使用しているものを紹介する。

a．数値データは全て行列として扱う。

すなわち，スカラーは1行1列の行列として扱い，ベクトルは行か列が1の行列として扱う。経済分析にはベクトルや行列が多く使用されるのでデータを行列に統一しておくと扱いやすくなるためである。

b．ファイルにおけるデータの記述はアスキー形式とする。

これは，バイナリ形式だとファイル容量が節約できる半面，ファイルの内容を見るために特別なコマンドを通す必要があるためである。

c．ファイルにおいては最初の数値が行数を，2番目の数値が列数を表す。

図4　データの記述法

1. スカラーの場合

A＝5の場合，ファイルの内容は

1 1 5

となる。最初の1が行数で，次の1が列数である。

2. 縦ベクトルの場合

$A=\begin{pmatrix}1\\2\\3\end{pmatrix}$ の場合，ファイルの内容は

3 1 1 2 3

となる。最初の3が行数で，次の1が列数である。

3. 横ベクトルの場合

A＝(1 2 3)の場合，ファイルの内容は

1 3 1 2 3

となる。最初の1が行数で，次の3が列数である。

4. 行列の場合

$A=\begin{pmatrix}1 & 2 & 3 & 4\\5 & 6 & 7 & 8\\9 & 10 & 11 & 12\end{pmatrix}$ の場合，ファイルの内容は

3 4 1 2 3 4 5 6 7 8 9 10 11 12

となる。最初の3が行数で，次の4が列数である。

以上であるが，ファイルにおいて1行にいくつ数値を並べるかは任意である。すなわち，A＝5の場合，次の3つの記述とも同じである。

(a) 1 1 5　(b) 1
1 5　(c) 1
1
5

3番目からが数値データで行列の行ごとに記述される。記述例を図4に示す。

データを以上のように記述すれば筆者が作成したコマンドが自由に使用できるようになる。また，データ構造変換プログラムは簡単なので，データ構造変換コマンドを利用すれば，異なったデータを持ったシステム間のコマンドの混合使用も簡単に可能だと思われる。

次に，基本的演算プログラムはコマンドにできるだけ汎用性を持たせるために，以下の点に留意して作成すべきである。

a．1つのコマンドに多くの機能を持たせないようにする。

使用者に自由な使い方を許すためにもコマンドの構造は単純な方がよい。また，コマンドの保守や修正にとってもこのことは大切である。

b．入出力ファイル名はコマンドのパラメーターで与えられるようにする。

特定のファイル名しか対応しないようなコマンドは汎用性に乏しいし，コマンドの実行時にファイル名を聞いてくるようなプログラムではコマンドを連続的に実行する際の使用者の負担が大きくなりすぎるからである。

c．配列を実行時に動的に宣言できること。

配列を動的に割り当てることにより，種々の大きさのデータに柔軟に対応でき，配列の上限を利用するハードの状態に適合させることができる。このようにすると，使用するハードのメモリーの状態に応じて配列の大きさを変更するという厄介な作業から解放される。

以上のような点に留意すれば，どのような言語プログラムを作成してもよいが，筆者はC言語でプログラムを作成することを推奨したい。C言語は経済分析のプログラム言語としてはFORTRANやBASICのように馴染みはないが，FORTRANやBASICでできることは全て可能で，かつ，それ以上の機能をも持ち合わせた言語である。

プログラム作成の例として第2節で使用した逆行列計算コマンドMATINVのソースプログラムをリスト4に示す。プログラムの内容を簡単に説明しておくと，プログラムはメインプログラム(22行目から44行目まで)と

4つの関数からなっている。4つの関数はそれぞれ，

行列入力の関数（47行目から70行目まで）

62行目で行列の行数，列数に応じてメモリーを確保している。

逆行列計算の関数（73行目から96行目まで）

逆行列計算の中心部分は82行目から91行目である。

計算結果の出力の関数（99行目から121行目まで）

110行目でファイルの1行目に行数と列数を出力している。

111行目から118行目で結果を出力している。

エラー処理の関数（124行目から130行目まで）

である。メインプログラムの最初を22行から24行までのように記述すると，このプログラムを実行するときのパラメーターの数が変数argcに，パラメーターの文字列が配列argv[]に格納され，プログラム内でパラメーターを参照可能となる。37行目で行列入力の関数を呼び出し，41行目で逆行列計算の関数を呼び出し，43行目で結果の出力の関数を呼び出している。

5. おわりに

以上，筆者の提唱する経済分析システムについて基本的な考え方を述べてきたが，それを要約すると以下のようになる。ある固定的な箱の中に多くの機能を詰め込もうとするとそれ自体に限界があるし，利用者の自由な使用を制限することにもなる。そこで，考え方を変えて，単純な機能しか持たない小さな箱を多く作成して，その組み合わせ方を替えることによりいろいろな目的に対応するシステムを構築すると次のような利点が生まれてくるのである。

a．個々のコマンドを単独にMS-DOSの外部コマンドとして使用できる。

b．コマンドの組み合わせ方法は利用者の自由である。

c．利用者は個々のコマンドの内部構造に注意する必要はない。他人が作成したコマンドでも計算が正しく，入出力の形式が統一しておれば自由に利用可能である。

d．作業内容が簡潔に記述できる。

e．作業の途中結果がファイルに保存されるので，計算のチェックや途中結果を用いた別の分析が簡単にできる。

f．プログラムの作成や修正がコマンド単位なので，保守や機能拡大が簡単にできる。

g．新しい作業に対する対応がコマンドの組替えや少数の新規コマンドの作成で済み汎用性がある。過去に作成したコマンドの蓄積を生かすことができる。

h．作業は個々の小さなコマンドを連続的に実行することにより為されるためCPUのメモリーが少なくても大きな作業をすることが可能である。

さらに，このような発想の下でパッケージソフトを開発すると，利用者はパッケージソフトの持つ利点を享受できると同時にそのパッケージソフトを構成している個々のコマンドを自由に使用して，特定の目的に応じたソフトを簡単に作成することが可能となる。

参考文献

[1] 内田幸夫「経済データの統計解析用計算機言語—COLSA—」『国民経済雑誌』，第151巻第4号，1985年

[2] 内田幸夫「統計解析用計算機言語COLSA Σ」『国民経済雑誌』，第155巻第3号，1987年

[3] アスキー書籍編集部編『MS-DOS 3.1ハンドブック』，アスキー，1986年

[4] B. W. カーニハン，D. M. リッチー共著『プログラム言語C』，共立出版　1981年

[5] ピート・メテヴェーレス著『MS-DOSファイル整理学』，アスキー，1986年

付表　生産関数推定のための統計データ

	GNP	K	L	LN(GNP)	LN(K)	LN(L)
1965	92027	—	4709	11.430	—	8.457
1966	102210	72836.4	4801	11.535	11.196	8.477
1967	113182	77956.1	4883	11.637	11.264	8.494
1968	127709	85367.1	4945	11.758	11.355	8.506
1969	142994	94609.8	5025	11.871	11.458	8.522
1970	153915	107655	5095	11.944	11.587	8.536
1971	161688	122134	5038	11.993	11.713	8.525
1972	176628	134077	5181	12.082	11.806	8.553
1973	184569	145915	5187	12.126	11.891	8.554
1974	183798	161372	5121	12.122	11.991	8.541
1975	190875	174884	5198	12.159	12.072	8.556
1976	199630	187840	5230	12.204	12.143	8.562
1977	210234	200313	5297	12.256	12.208	8.575
1978	221243	212196	5353	12.307	12.265	8.585
1979	232878	225539	5425	12.358	12.326	8.599
1980	242131	240989	5488	12.397	12.393	8.610
1981	250159	257585	5550	12.430	12.459	8.622
1982	258241	273581	5633	12.462	12.519	8.636
1983	267782	289140	5657	12.498	12.575	8.641
1984	281102	304655	5709	12.546	12.627	8.650
1985	292838	323256	5752	12.587	12.686	8.657

(注)　出所　ECONOMATE マクロデータファイル

GNP：実質国民総生産（昭和55年基準）　単位　10億円

K：実質民間企業設備資本ストック（昭和55年基準）　単位　10億円

L：就業者数　単位　万人

ただし，データは年度データで，Kは期首，Lは期末データである。

リスト4　逆行列コマンドのソースプログラム

```
1  /**
2      matinv.c  逆行列
3
4
5      使用法  MATINV A B
6                              A : 入力行列
7                              B : 出力行列  B = INV(A)
8
9                                        作成者  大野拓行
```

```
10 :                                                作成日  02/05/1987
11 : **/
12 :
13 : #include <stdlib.h>
14 : #include <stdio.h>
15 : #define TYPE double                  /*TYPEをdoubleと定義*/
16 :
17 : typedef struct {                     /*構造体MATRIXの定義*/
18 :     int n, m;                        /*   n:行数   m:列数   */
19 :     TYPE*a;                          /*   a:行列データへのpointer*/
20 :     }MATRIX;
21 :
22 : void main(argc, argv)                /*main program*/
23 : int argc;
24 : char *argv[];
25 : {
26 :     void error(), mat_out3();        /*使用関数の宣言*/
27 :     int mat_inpl();
28 :     MATRIX *mat_inv();
29 :
30 :     MATRIX *a;                       /*使用変数の宣言*/
31 :
32 :     allmen();                        /*メモリの確保*/
33 :
34 :         if (argc != 3)               /*エラーチェック*/
35 :           error ("使用法に誤りがあります。",NULL);
36 :
37 :         mat_inpl(argv[1], a);        /*行列の入力*/
38 :         if(a->n != a->m)
39 :           error ("正方行列ではありません。", NULL);
40 :
41 :         a = mat_inv(a);              /*逆行列の計算*/
42 :
43 :         mat_out3(a, argv[2]);        /*結果の出力*/
44 : }
45 :
46 :
47 : int mat_inpl(a, b)     /*行列入力の関数*/
48 : char*a;                /*第1引数:入力データファイル名*/
49 : MATRIX*b;              /*第2引数:構造体MATRIXへのpointer*/
```

```
{
    FILE*fp_r, *fopen();                    /*使用変数の宣言*/
    long siz = sizeof(TYPE);
    int i, n, m;

    if ((fp_r = fopen(a,"r")) == NULL)      /*ファイルの open*/
        error ("ファイルが open できません %s", a);

    fscanf(fp_r, "%d", &b->n);              /*行数と列数の読み込み*/
    fscanf(fp_r, "%d", &b->m);
    n = b->n; m = b->m;

    if((b->a = (TYPE*)getml(n*m*siz)) == NULL)  /*メモリの確保*/
        error("out of memory", NULL);

    for(i = 0; i<n*m; i++)                  /*行列データの読み込み*/
        fscanf(fp_r,"%lf", &*(b->a+i));

    fclose(fp_r);                   /*ファイルの close と return*/
    return(0);
}


MATRIX*mat_inv(a)     /*逆行列計算の関数*/
MATRIX*a;             /*引数 : 構造体 MATRIX への pointer*/
{
    int n, i, j, k;                 /*使用変数の宣言*/
    TYPE p, q, *x;

#define X(i, j) *(x+(i)*n+j)        /*X(i, j) の定義*/

    n = a->n; x = a->a;             /*逆行列の計算*/
    for(k = 0; k<n; k++){
        p = X(k, k);
        X(k, k) = 1.0;
        for(j = 0; j<n; j++) X(k, j) = X(k, j)/p;
        for(i = 0; i<n; i++){
            if(i == k) continue;
            q = X(i, k);
            X(i, k) = 0.0;
```

```
90                  for(j = 0 ; j<n ; j++)
91                      X(i, j) = X(i, j)-q*X(k, j);
92              }
93          }
94  # undef X
95          return(a);              /*構造体 MATRIX への pointer を返す*/
96  }
97
98
99  void mat_out3(a, b)             /*行列印刷の関数 (type III)*/
100  MATRIX *a ;
101 char *b ;
102 {
103         int i, j, l = 0 ;                   /*使用変数の宣言*/
104         int n, m ;
105         FILE *fp_w, *fopen() ;
106
107         fp_w = fopen(b, "w") ;              /*ファイルの open*/
108
109         n = a->n ; m = a->m ;
110         fprintf(fp_w, %d %d¥n", n, m);/*行数, 列数の出力*/
111         for(i = 0 ; i<n ; i++){             /*行列データの出力*/
112             for(j = 0 ; j<m ; j++){
113                 fprintf(fp_w, %2.12E", *(a->a+i*m+j));
114                 ++l ;
115                 if(l == 4){fprintf(fp_w, "¥n") ; l = 0 ;}
116             }
117         if(l) {fprintf(fp_w, "¥n") ; l = 0 ; }
118         }
119         fclose(fp_w) ;                      /*ファイルの close & return */
120         exit(0) ;
121 }
122
123
124 void error(s1, s2)              /*エラーメッセージの表示と実行停止*/
125 char *s1, *s2 ;
126 {
127         fprintf(stderr, s1, s2) ;
128         fprintf(stderr, "¥n") ;
129         exit(1) ;
130 }
```