ＡＰＬ　から　Ｊ　へ　再帰法・ 行列式計算

固有値問題シリーズ　（その８）

中野嘉弘　（札幌市・８５才）

FAX 専　011-588-3354　yoshihiro@river.ocn.ne.jp

行列式の求値を、Ｊ言語の　基本的関数　det =: -/ . * を用いずに、やる方法の一例である。　APL 言語でのそれ（特に再帰法）を J 言語でも実現したものだ。著者は、今は、Dyalog APL101 と　J601版　を使用している。

０．　は　し　が　き

Ｊ研の１１月例会報告「行列式の計算法異考」（文献１）の補足・延長である。それは、今まで見逃されて来た固有値問題の直接法（文献　２ａ～２ｆ）、即ち、我らの Naigen 法、さらに、最古参の Frame 法、あるいは、同じ内容かもしれないが、仏露の大家 ルヴェリエ・ファデーエフ　Leverrier-Faddeev 法と格闘している内に、気が付いた事柄であった。

行列式の計算では、Ｊ言語は断然、有利である。　それは、Ｊ言語の基本的関数の det =: -/ . *　のおかげである。　同じような基本関数を提供するのは、 J 言語の他には、SHARP APL 位かも知れぬ（文献２）。　小生愛用の英国製の　Dyalog APL では残念ながら、この流儀は出来なかった。　かの有名な　APL2 では、確認テストの暇がなかった。　数年前に折角、インストールして置いた教育版を動かせないので、専門家からの情報を期待したい。

一般的に、行列式の計算法には、昔、 VECTOR誌で　Joseph De Kerf （文献２）が APL 言語について述べて居る如く、
１）　Recursive 再帰法　と　２）　Elimination　とが可能である。
J 言語については、　２）消去法について、前稿で述べたばかりである（文献１）。今回は、前回（少々難物と見えたので！）触れなかったが、１）　再帰法　（Recursive Method）の方について述べる。

１．　再帰法　の　ＡＰＬ　ＤＥＴ関数

APL Font のワープロ組版は難物なので、下記の遺漏分は、稿末のスキャナー版を参照されたい。

● recursive

```
[0]  D  DET1  M;N;I;  IO
[1]   ((2   M)  (1  =/ M)  (1  ,M)=  1   ,M)/0
[2]   (2 > N)/0, D  (+/1  ,M)+0=N  1   M
[3]  D  N  I   IO  1
[4]  LAB: D[I]  DET1  M{1   N;(I   N)/  N]
[5]    (N  I  I+1)/LAB
[6]  D -/M[1;]xD
```

テスト行列：　前稿（文献１）と同じもの

```
ya2 ← 2 2 r 6 2  2 3
yb3 ← 3 3 r 3 1 4  1 5 9  2 6 5
yc5 ← 5 5 r (i 5),((i 5)*2), ((i 5)*3), ((i 5)*4), ((i 5)*5)
```

APL 言語での計算結果：

```
DET1  ya2  → 14
DET1  yb3  → -90      （APLでの負数表示は、本来はアッパー・バーだ。）
DET1  yc5  → 34560
```

２．　再帰法　の　Ｊ　ＤＥＴ関数J 言語プログラム

（J601版　中野）

```
    detn =: 3 : 0
      n =. # y
  if.  n = 2  do.  det2  y   goto_0.  end.
    - / (0{y) * (> detn each cofact0b y)
 label_0.
    )

    det2 =: 3 : 0
-/ (0{y) * |.1{y
    )

   cofact0b =: 3 : 0
 n=. # y
 d =. < (n,n) $ 0
 y0 =. 0 }. y
   i =. 0
 while. i < n do.
   ci =. < (<<<i) {"1 y0
   d =. d, ci
   i =. i + 1
 end.
  d =. }. d
   )
```

テスト解：　データ行列は、前節と同じ

```
   detn  ya2
 14

   detn  yb3
_90

  detn yc5
34560
```

３．　時間計測関数　Tdetn　では

APLの計時関数は、ラフ（msec 級には無効）なので、Ｊ　言語のみの話である。

```
Tdetn =: 3 : 0

wr  y
  t =. 6!:1''

 d =.  detn  y

wr (':(6!:1'') - t ), ' sec '
 d
  )
```

データ行列例

```
rot=: 3 : 0
  (}. y), {. y
)

latin =: 3 : 0
rot ^:(i. y)  a =. >: i.y
  )
```

エル２　　l2=:　latin 2
エル３　　l3=:　latin 3
エル４　　l4=:　latin 4
エル５　　l5=:　latin 5
エル６　　l6=:　latin 6
エル７　　l7=:　latin 7
エル８　　l8=:　latin 8
エル９　　l9=:　latin 9
エル１０　l10=:　 latin 10　 等々

```
   Tdetn  l5
1 2 3 4 5
2 3 4 5 1
3 4 5 1 2
4 5 1 2 3
5 1 2 3 4
  0 sec
1875

   Tdetn  A88
 0 1 0 0  0  0  0 0
_2 0 1 1  0  3  0 0
 1 1 0 0  0 _1  0 0
 1 0 0 0  0 _1  0 0
_1 0 4 2 _1  1  0 0
 0 1 0 0  0  0  0 0
 0 0 0 0  0  0  1 0
 0 0 0 0  0  0  0 2
  1.703  sec
```

0　（行列式値）
1
以上で、前稿に対する「再帰法」の補足は済んだ。　以下は、その後の知見である。

４．　その他の　ＡＰＬ　ＤＥＴ　関数

　ＡＰＬ言語には、行列式値を計算する簡便な基本的関数、例えば　Ｊ言語に於ける **det =: -/ . *** の類は、一部の例外（**Sharp APL**）を除けば無いので、それに代わる関数が、数多く工夫されている（文献２）。　その幾つかは、前節までに話題にしたが、その後、気が付いたものを、取り上げて置こう。
例えば、 **Maurice Dalois** 著　**" APL*PLUS PC " p.134** （文献３）等である。
「消去法」と「再帰法」のそれぞれがある。　トライされたい。
　APL Font のワープロ組版は難物なので、下記の遺漏分は、付録のスキャナー版を参照されたい。

１）内容的には「消去法」と思われるもの。

```
        ▽  Z ← PCDET M ; K
[1] → 0  IF 1 = 1 ↑ rho Z ← M
[2] → OK  IF  0 ≠ Z ← M[1;1]
[3] → IF 0 = K ← 1 ↑ ( 0 ≠ M[;1]／iota ↑rho M
[4] M[1,K;] ← M[K,1;]
[5] Z ← M[1;1]
[6] OK: M ← M - M[;1]。.x M[1;] ÷ M[1;1]
[7] Z ← Z x PCDET 1 1 ↓ M
[8] ▿
```

検算データ　**M0 ← 4 4 rho 2 _1 0 3 3 1 4 6 _2 4 0 0 0 1 2 _3**
演算例　　**PCDET M ← M0**　から　行列式値は　**84**

２）　上例を、本質的部分のみに簡単化した下記でも演算結果は同じ。

```
        ▽  Z ← PCDET1  M ; K
[1] → 0  IF 1 = 1 ↑ rho Z ← M
[2] Z ← M[1;1]
[3] OK: M ← M - M[;1]。.x M[1;] ÷ M[1;1]
[4] Z ← Z x PCDET1  1 1 ↓ M  ▿
```

３）　再帰法のもある。

```
        ▽  Z ← PCDET2  M ; I ; SIGN
[1] → ( 1 = 1 ↑ rho Z ← M)／0
[2] Z ← 0  ◇ I ←1  ◇ SIGN ← 1
[3] AG: Z ← Z + SIGN x M[I;1] x PCDET2  0 1 ↓ (I ≠ iota 1 ↑ rho M) ? M
[4]    SIGN ← _1 x SIGN
[5]  →((1 ↑rho M) ≧ I  ←  I + 1)／ AG  ▿
```

演算結果　　**PCDET2 M**　から　**84**

　これら今回追加の　ＡＰＬでのプログラム例を、Ｊ言語に　コンバートするのも、面白い問題かもしれぬ。

文　　献

１）中野嘉弘・山下紀幸：「行列式の計算法異考　固有値問題の脇き道」
固有値問題・シリーズ　（その７）　**JAPLA 2007 Nov 24,　pp.10**

２）　**Joseph De Kerf: " APL Defined Functions for the Calculation of Determinants ", VECTOR 1994, Vol.10 No.3, pp.21-22**

３）　**Maurice Dalois : " Introduction to APL*PLUS PC ", p.134**
1993, Educ APL Inc., Quebec, CANADA

```
      ⎕←DET1
     ∇ D←DET1 M;N;I;⎕IO
[1]    →((2≠ρρM)∨(1≠=/ρM)∨(1↑,M)=¯1↑,M)/0
[2]    →(2>N)/0,D←(+/1↑,M)+0=N←1↑ρM
[3]    D←NρI←⎕IO←1
[4]   LAB:D[I]←DET1 M[1↓⍳N;(I≠⍳N)/⍳N]
[5]    →(N≥I←I+1)/LAB
[6]    D←-/M[1;]×D
     ∇
```

```
      ⎕←PCDET
     ∇ Z←PCDET M;K
[1]    →0 IF 1=1↑ρZ←M  ⍝ EXIT IF M IS 1 BY 1 MATRIX
[2]    →OK IF 0≠Z←M[1;1]
[3]    →0 IF 0=K←1+(0≠M[;1])/⍳1↑ρM ⍝ ALL 0 DET 0
[4]    M[1,K;]←M[K,1;] ⍝ CHANGE LINE 1 AND K
[5]    Z←-M[1;1]  ⍝ CHANGE SIGN
[6]   OK:M←M-M[;1]∘.×M[1;]↓M[1;1]
[7]    Z←Z×PCDET 1 1↓M  ⍝ DET OF SUB MATRIX
     ∇
      ⎕←  PCDET1
     ∇ Z←PCDET1 M;K
[1]    →0 IF 1=1↑ρZ←M
[2]    Z←M[1;1]
[3]   OK:M←M-M[;1]∘.×M[1;]↓M[1;1]
[4]    Z←Z×PCDET1 1 1↓M
     ∇
      ⎕←  PCDET2
     ∇ Z←PCDET2 M;I;SIGN
[1]    →(1=1↑ρZ←M)/0
[2]    Z←0 ⋄ I←1 ⋄ SIGN←1
[3]   AG:Z←Z+SIGN×M[I;1]×PCDET2 0 1↓(I≠⍳1↑ρM)≠M
[4]    SIGN←¯1×SIGN
[5]    →((1↑ρM)≥I←I+1)/AG
     ∇
```

```
      ⎕←IF
     ∇ S←A IF B
[1]    S←B/A
     ∇
```

【付録： Scanner 版 の Scripts 類 】

```
      ⎕←DET1
     ∇ D←DET1 M;N;I;⎕IO
[1]    →((2≠ρρM)∨(1≠=/ρM)∨(1↑,M)=¯1↑,M)/0
[2]    →(2>N)/0,D←(+/1↑,M)+0=N←1↑ρM
[3]    D←NρI←⎕IO←1
[4]   LAB:D[I]←DET1 M[1↓⍳N;(I≠⍳N)/⍳N]
[5]    →(N≥I←I+1)/LAB
[6]    D←-/M[1;]×D
     ∇
```

```
      ⎕←PCDET
     ∇ Z←PCDET M;K
[1]    →0 IF 1=1↑ρZ←M   ⍝ EXIT IF M IS 1 BY 1 MATRIX
[2]    →OK IF 0≠Z←M[1;1]
[3]    →0 IF 0=K←1↑(0≠M[;1])/⍳1↑ρM ⍝ ALL 0 DET 0
[4]    M[1,K;]←M[K,1;] ⍝ CHANGE LINE 1 AND K
[5]    Z←-M[1;1]  ⍝ CHANGE SIGN
[6]   OK:M←M-M[;1]∘.×M[1;]↑M[1;1]
[7]    Z←Z×PCDET 1 1↓M  ⍝ DET OF SUB MATRIX
     ∇
      ⎕←  PCDET1
     ∇ Z←PCDET1 M;K
[1]    →0 IF 1=1↑ρZ←M
[2]    Z←M[1;1]
[3]   OK:M←M-M[;1]∘.×M[1;]↑M[1;1]
[4]    Z←Z×PCDET1 1 1↓M
     ∇
      ⎕←  PCDET2
     ∇ Z←PCDET2 M;I;SIGN
[1]    →(1=1↑ρZ←M)/0
[2]    Z←0 ⋄ I←1 ⋄ SIGN←1
[3]   AG:Z←Z+SIGN×M[I;1]×PCDET2 0 1↓(I≠⍳1↑ρM)⌿M
[4]    SIGN←¯1×SIGN
[5]    →((1↑ρM)≥I←I+1)/AG
     ∇
```

```
      ⎕←IF
     ∇ S←A IF B
[1]    S←B/A
     ∇
```