ＡＰＬ　と　固有値問題

固有値問題シリーズ（その９）

中野嘉弘　（札幌市・８５才）

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

固有値問題の解法をＡＰＬ言語で行う。　直接法が主であり、今までのＪ言語を主にしたシリーズの　８篇　等に続くものである。
著者は Dyalog APL101 と J601版　を使用している。

は し が き

我らの固有値問題シリーズの前回（８篇目）は「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」であった。　そこで、懐かしい ＡＰＬ言語も再登場した。　（文献１ａ－１ｇ）、（文献　２）
その関係で、昔の資料の中に、１９８１年当時「ＡＰＬで固有値問題をと張り切った頃の思い出」を再発見した。
１９８１年秋、ＡＰＬ学会がキャリフォーニアの　パロアルト Palo Alto であった。その時、知り合った IBM 藤沢研の 森脇幸生氏から恵贈された、彼の訳書「ＡＰＬ微積分のアルゴリズム的方法」（文献３）と固有値問題解法の資料であった。
資料例題は実対称行列で、次数は高々、２次、３次まで。（文献３－ａ）。
計算時間の例では、IBM5100 マシンで、次数　5, 7, 10, 15 で Time 75 、135 、270 、770 (sec) があった。　また、なんと　２次のデータ行列　-3 4 -2 1 でも、収束解は得られなかったともあった。　報告者はスエーデンの大学の Axel Ruhe 先生でした。
この頃は、こんな簡単なもので、一喜一憂だったのか！

今ではＪ言語で、固有値問題を、直接法で容易に解ける御時勢になったのだから、ＡＰＬで復習して見るのも意味があろう。

１．　直接法　を　ＡＰＬ　でも

直接法で解ける限り、他の近似的手法に頼る理由は無かろう！
それには、与行列の固有多項式（の係数）を求める。　今まで、Ｊ言語の例で、繰り返し述べたように、ルヴェリエ・ファデーエフ法（文献１－ｄ）またはＦｒａｍｅ法（文献１－ｅ）のアルゴリズムを利用する。　Ｊ言語で関数 Feigen として、散々扱って来たから、ＡＰＬでも簡単である。　例示する。

```
     ▽ D ←FEIGEN  A ;  X ;  E ;  K ;  N ;  C ;  D
[1]    →( 1 = N ← 1 ↑ A) / 0
[2]  X ←  E ← (N,N)  1, N  0  ◇  K ← 1  ◇  D ← 1
[3]  AG  :    X ← A +.x  X
[4]        C ←  (+/(1 1  X))  K x  ~1
[5]        D ←  D, C
[6]        X  ←   X  +  C x E
```

```
[7]       → ( N ≧ K ← K + 1 )/ AG              ▽
```

APL Font　でのワープロ作業は、難物なので、遺漏の場合は、稿末の　スクリプト（スキャナー版）を参照下さい。　また、APL の負数の負号は、Ｊ言語のアンダーバーと反対にトップバーであるから、今は　tilde チルデ　波記号で代用表示している場合もあるので、御了承下さい。

```
例題            A1 ← 3 3  1 0 5  0 8 0  5 0 1
        FEIGEN  A1
    1  ~10  ~8  192    （固有多項式の係数、左端が最高の３次項）
       8  6  ~4        （固有値は３ケ、　固有多項式の解より）
```

２．　固有多項式の解法

Ｊ言語の基本関数　ｐ．　に対応するものが、ＡＰＬには無いので、別途、その関数　PSUM を作成した。

```
      ▽ P ← C  PSUM  M ;  K ;  N ;  I ;  E
  [1]     E ←  ~1 +  (  C))
  [2]   K ← 1 ↑ M
  [3]      P ←  + ／  C x ( K * E)
  [4]      N ←  1 ↓ M
  [5]      I ←  K + 1
  [6]   AG  :  P  ← P , +／C x ( I *  E)
  [7]      → ( N ≧ I ← I + 1 )/ AG
  [8]      → 0
         ▽
```

演算例：　途中経過をも示せば

```
              与行列　A1、      Feigen  A1
              固有多項式    C  ←  1  ~10  ~8  192

          □  ←    P ← C   PSUM  ( ~5, 10 )      （最小、最大　予想値）
~143  0  99  160  189  192  175  144  105  64  27  0  ~11  0  39  112
       （零点は３ケ処）

          □  ←    Q  ←  ( - iota 5, 0, iota 10 )
  ~5  ~4  ~3  ~2  ~1   0   1   2   3   4   5   6   7   8   9   10  （対応する）

          □  ←    X  ←     Q  x  ( P = 0 )
  0  ~4   0   0   0   0    0   0   0   0    0   6   0   8   0   0
```

０　を消去したい。

```
               ( X ≠  0 ) ／  X          より
             ~4   6   8
```

これが最終的な　「根」　即ち　「固有値」　である。
勿論、途中経過を示さないような、関数作成も出来る。

３．　データ行列　ラテン　ＬＡＴＩＮ　から

Ｊ言語での固有値問題の例題に盛んに登場した行列作成法　**Latin n**　の例を採る（文献６）。

```
        ▽ D ←  LATIN  N;A;I
[1] → (2>N) / 0  ◇  I ← 1
[2]  D ← A ← 0 phi iota N
[3]  AG: D ← D, A ← 1 phi A
[4]    → (N ≧ I ← I + 1)/AG
[5]    D ← (N,N) rho D
[6]   → 0
      ▽
```

演算例：　　**C ← FEIGEN L3　← LATIN 3**

```
        C              L3
  1 ~6  ~3  18        1 2 3
                      2 3 1
                      3 1 2
```

固有値の解は今は、整数ではないので、簡単でない（Ｊ言語　ｐ．関数の解では　**6、 ± 1.73205**）。

　□　←　**P ←　C　PSUMA　M ← (~2, 10, 1)**　即ち（最低、最高、増分）と置いて、結果の　**P**　値は

~8 14 18 10 ~4 ~18 ~26 ~22 0 46 122 234 388

従って、解（値０の項）は、最大値（解は最後の項）から「４つ」手前の　６　である。

１）　これは、解の最初である。　６　を明示したいならば、

```
  □  ←   Q ← (( - iota  2), 0, iota 10)
 ~2 ~1  0  1  2  3  4  5  6  7  8  9  10
         Q x (P = 0)
  0  0  0  0  0  0  0  0  6  0  0  0  0
```

或いは　　**(X　≠　0) ／ X ← Q x (P = 0)**　から　６　を得る。

２）　次の解　（小数）

解の近似値が判っていれば、楽である。

捜査範囲の指定は　　**M ← (1.732, 1.7321, 0.00001)**　とするのが良い。

```
    P ← C  PSUMA  M
    B ← |P      （絶対値を採る）
    B  >  0.0001
  1  1  1  1  1  0  1  1  1  1  1
    Q ← (1.732 +  0.00001 x (0, iota 10))
    Q  x  ( B ≦ 0.0001 )
 0  0  0  0  0  1.73205  0  0  0  0  0
```

さらに、上記の　**X**　表記を利用すれば、肝心の　**1.73205**　のみを選び、表示出来る。

３）　第３の解　（負の小数）

捜査範囲の指定は　　**M ← (~1.7321, ~1.7320, 0.00001)**　とする。

```
    P ← C  PSUMA  M
```

```
        B ← |P      （絶対値を採る）
        B < 0.0001
    0 0 0 0 0  1  0 0 0 0 0
        Q ← (~1.7321 +  0.00001 x (0, iota 10))
        Q  x  ( B ≦ 0.0001 )
 0 0 0 0 0  ~1.73205   0 0 0 0 0
```

かくて、固有値の３根、即ち　６、 **±1.73205**　が　ＡＰＬ言語で確認された。

この演算から、多項式の　根の発見　（**ROOT FINDER**）には、更に工夫が要る事が判る。

４．　ＲＯＯＴ　ＦＩＮＤＥＲ　の工夫

１）前節の求根問題は、原３次多項式を、最初の根による１次式（**X - 6**）で除せば、残りの因子（今は２次式）は簡単に求められる筈である。　この流儀をトライしよう。

A ← 1 ~6 ~3 18　（原３次式）、　　**B ← 1 ~6**　（除式、１次）　として
２つの　多項式の商は、暗算でも可能な位だが、敢えて関数　**POLDV** でも作れば
C ← A POLDV B　から　　**C** は２次式で、係数は　**1 0 ~3**　となる。
この２次方程式の根は、殆ど暗算か、簡単な筆算でも解けるが、敢えて関数 **ROOT2** でもつくれば
ROOT2 C　より、２根は、　**1.732050808** と　**~1.732050808**。
それと以前の根　６　とで、固有値の全てが求められる。

用いる関数　**POLDV** や　**ROOT2** は簡単であるから、読者の御自作にまかせたい。
しかし、負数の平方根など（虚数）は許されぬ（**dominant error**）。
また、複素数相当は、２成分ベクトルとして扱う。　要注意！　つまり、**APL** 言語では、虚数を「数」として認めておらぬらしいので、複素数根は、当初から、諦めねばならぬ。
従って、固有値問題では、**APL** は最初から、重大な制限を覚悟せねばならぬ！

２）　**General Root Finder** 関数　**GRF**　（**K.E. Iverson** 作成、文献４）の利用

```
        ▽ Z ← B GRF X
   [1] Z ← .5x+／B
   [2] B[1+ X < F Z] ← Z
   [3] → .00001 < |X - F Z ▿
```

意味は、**B** は　根の探査範囲、　**X** は　多項式値の設定

問題対象の多項式自身の設定は、今は係数　**1 ~6 ~3 18** を　逆順に代入して

```
        ▽ Z ← F X
 [1] Z ← 18 + (~3 x X) + (~6 x X*2) + (1 x X*3) ▿
```

演算：　変数 **X** の範囲　**5 〜 6**　で、

```
        5 6 GRF 0      から　根の値は    6
        2 1 GRF 0                 1.732050896
       ~2 0 GRF 0                ~1.732050896
```

（ 注意：　**0 ~2** の範囲指定は不可であった。）

大変、うまく行った！

なお、森脇氏の著書（文献３、ｐ．274）の練習問題　７．５７(c)に　「固有値問題、特性多項式の係数ベクトルを、ドミノ関数を使わずに求めよ」　があり、フレーム法の名前（文献７）も、脚注に、書かれている。　「知る人ぞ知る」事柄だったのだ。

５．　む　　す　　び

全く、「温古知新」と云うか、大昔の先人のお知恵に、敬服する事が多々あるものです。　　当時のAPL学会の報告を、コンピュータ誌「ｂｉｔ」に寄稿した記憶がある。　その旧聞が再発見されれば、面白い話題が得られるかも。　懐かしいな！

文　　　献

１－ａ）中野嘉弘「Ｊ言語と高等数学　（固有値問題直接法）」
JAPLA報告　2007.4.2 pp.9

－ｂ）中野嘉弘「Ｊ言語と固有値問題（その２）直接法の発展」
JAPLA報告　2007.5.26 pp.13

－ｃ）中野嘉弘「Ｊ言語と固有値問題（その３）VECTOR誌投稿原稿」
JAPLA報告　2007.6.23 pp.3

－ｄ）中野嘉弘「Ｊ言語と固有値問題（その４）チュートリアル等　LAPACKまで」
JAPLA報告　2007.6.23 pp.9

－ｅ）中野嘉弘「Ｊ言語と固有値問題（その５）　ファデーエヴァ法へ
JAPLA報告　2007.9.22 pp.8

－ｆ）中野嘉弘・山下紀幸「Ｊ言語と固有値問題（その６）　**frame** 法を主に
JAPLA報告　2007.10.27 pp.8

－ｇ）中野嘉弘・山下紀幸「行列式の計算法異考：固有値問題（その７）脇き道
JAPLA報告　2007.11.24 pp.8

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

３）森脇幸生 訳・日本ＡＰＬ協会 編：「ＡＰＬ 微積分のアルゴリズム的方法」
工学図書、１９８０、昭和５５年１１月２０日
原著：　**D.L.ORTH　" CALCULUS in a new key "　1976, APL Press**

－ａ）　**APL Quote Quad 1980**　より抜き書き
Axel Ruhe "Eigenvalues and Eigenvectors by Rayleigh Quotient Iteration March 1980 pp.29-30 and its Correction (June 1980 p.18)

４）**Kenneth E. Iverson: " ALGEBRA An Algorithmic Treatment ", pp. 137-138 APL PRESS Pleasantville New York, 1971,1972, 1977**

５）**Kenneth E. Iverson: " Elementary Analysis " , 1976, APL PRESS Swarthmore, PA, USA**

６）山下 **FAX (2007. 6. 3 11:00)** : データ行列作成関数　**latin** と **rot**

７）古屋　茂「行列と行列式」培風館、**1957** 初版、**1959** 増補版、**1998** 増補第**65**刷

付録：　スクリプト　スキャナー版

```
      ⎕←FEIGEN
     ∇ D←FEIGEN A;X;E;K;N;C;D
        →(1=N←1↑⍴A)/0
        X←E←(N,N)⍴1,N⍴0 ⋄ K←1 ⋄ D←1
      AG:X←A+.×X
        ⎕←C←(+/(1 1⍉X))÷K×¯1
        D←D,C
        X←X+C×E
        →(N≥K←K+1)/AG
     ∇
```

```
      ⎕← PSUM
     ∇ P←C PSUM7 M;K;N;I;E
        E←⌽(¯1+⍳(⍴C))
        K←1↑M
        P←+/C×(K*E)
        N←1↓M
        I←K+1
      AG:P←P,+/C×(I*E)
        →(N≥I←I+1)/AG
        →0
     ∇
```

```
      ⎕←LATIN
     ∇ D←LATIN N;A;I
        →(2>N)/0 ⋄ I←1
        D←A←0⌽⍳N
      AG:D←D,A←1⌽A
        →(N≥I←I+1)/AG
        D←(N,N)⍴D
        →0
     ∇
```