JAPLA 研究会資料 2007/6/23

Ｊグラフィックスによる固有値・固有ベクトルの体験ツール

－コンピュータ時代の固有値問題の攻略法－

西川　利男

０．はじめに

固有値問題は、かつて大学の教養課程の線形代数で最後の課題として教えられたという［1]。現在でもそうだろう。しかしこんなことでは困る！　たとえ最初でなくともテキストの中ほどで、じっくりと教えるべきである。ところが線形代数のテキストをひもといてみると、相変わらずたどるのもうんざりする数式が並んでいるばかりである。

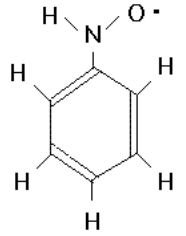
ここで強調したいのは、数式で理論を説明した後、二、三の数値問題で値を求めるだけではなく、固有値がどういう場面で、どう使われるかを具体的な例をあげて、その重要性を示すことがぜひ必要である。なお、簡単な２次の場合にどう計算して値を求めるかは大切だが、もっと高次の場合に固有値をどう求めるかのいろいろな手法（べき乗法、QR 法など）に多くのページをさくことはない。先月、筆者が示したように[2]、固有値は平方根と同じにコンピューター発で値は求められるからである。

今回の発表はＪのグラフィックスにより、あらためて固有値、固有ベクトルとはどんなものかを基本にかえり体験的に理解しようというツールである。

１．固有値問題はどんな場面で現れるか？

実用上、固有値が必要とされる場面はいろいろあるが、とくに微分方程式との関連でさまざまな分野で現れる。例えばストラングの線形代数の教科書[3]では数々の実際例があげられているが、日本の教科書では残念ながらこういった本はない。

・力学 …… バネにつながったいくつかの物体の振動
・パイプの中の物質の拡散の過程
・電気回路における電気振動
・機械、建築、土木などの振動の解析
・量子力学、量子化学における分子、原子、電子の挙動
・統計学の主成分分析
・数理経済学での生産・消費、経済成長のモデル
　　…… など



筆者の場合は、筑波の研究所時代に、左のような Phenyl Nitroxide 誘導体ラジカルのスペクトルの解析にあたって、大型計算機の上で固有方程式を FORTRAN のヤコビ法で数値計算したことなどを思い出す

２．固有値、固有ベクトルとは何か

線形代数で最も基本となる線形変換、つまりマトリックスとベクトルとの以下の演算に対して、J のユーザなら次のように J の用語で理解したらよい。このような考え方についての筆者の試みは以前報告した[4],[5]。ちなみに筆者は行列なる語は動詞というより名詞のイメージが強く、あまり適切ではないと思っている。

$$\begin{pmatrix} \text{マトリックス} \end{pmatrix} \cdot \begin{pmatrix} \text{ベ} \\ \text{ク} \\ \text{ト} \\ \text{ル} \end{pmatrix} = \begin{pmatrix} \text{ベ} \\ \text{ク} \\ \text{ト} \\ \text{ル} \end{pmatrix}$$

名詞　　動詞

動詞　　名詞　　名詞

マトリックス自身は名詞だが、動詞ドット積(J では+/ ・ *)の左引数となり、マトリックス積全体は動詞として、名詞のベクトルに作用して、新しい名詞ベクトルを生成する。このやり方は 1&o. が cos を、2&o. が sin を表すことに慣れている J ユーザには奇異なことではない。

ベクトルはマトリックス積なる操作によって大きくなったり、小さくなったり、また向きも変わる。つまり、マトリックス積は回転、伸縮の操作をおこなう動詞である。

ここで、伸縮はするが向きが変わらないような状態があるだろうかと考える。これに対する解の有無が固有値問題に他ならない。

ここで先の量子化学の固有値問題を、筆者なりに“文学的に”イメージしてみよう。ベクトルで表される“もの”は電子の波動関数であり、これは電子がどう広がっているかを示す。一方、まわりの環境、今の場合は７個の原子の化学結合による構造というワクであり、これはハミルトニアンなる微分方程式で表される[6]。ここで関数に微分方程式を作用させるという意味を、ベクトルにマトリックスを操作させる、と読み替える。そこで固有値を求めるということは、決まったエネルギーを持つ定常状態の解を得ることに相当する。その結果、このワク内で電子がどう広がっているかが分かる。ここでもハミルトニアンとは操作をおこなう動詞と考えると良く納得がゆく。

このように微分方程式を解くということを固有値の問題に置き換えて考えるというやり方は物理学で共通に使われる手法であって、物理学における固有値問題はこのため

にこそあるとさえ言えるのである。

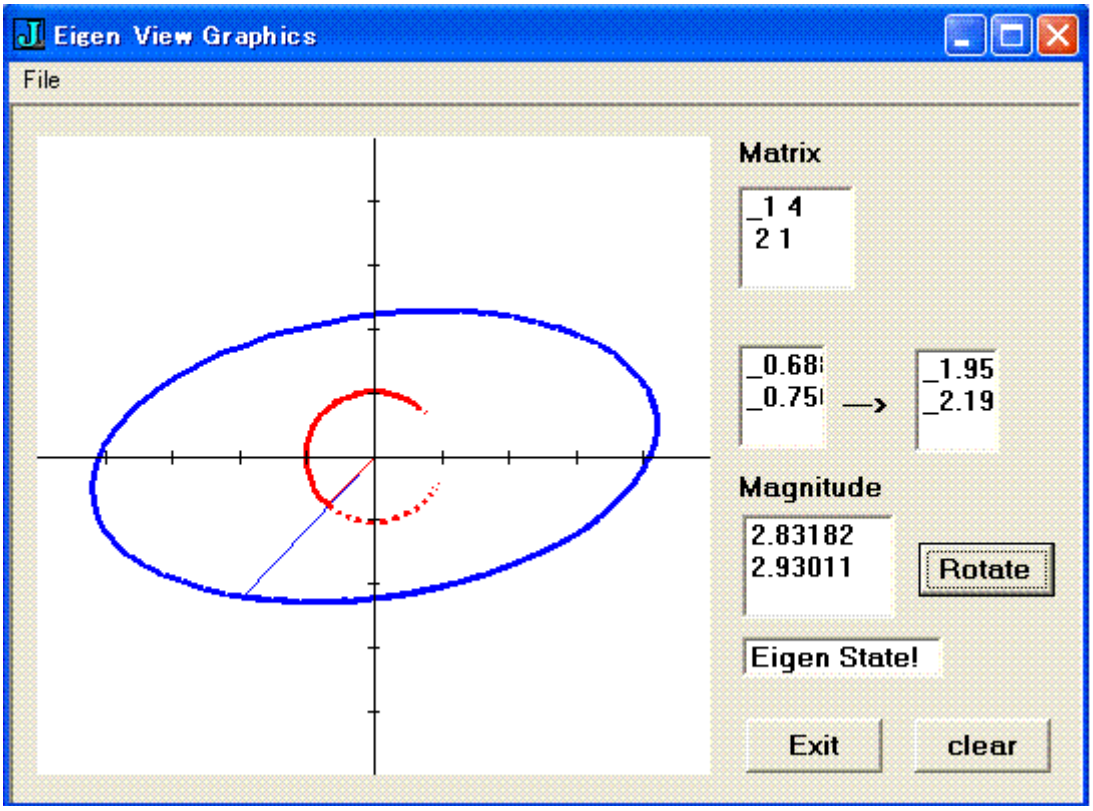
３．固有値表示グラフィックス(Eigen View Graphics)の実際

上のような固有値、固有ベクトルの操作をコンピュータ・ディスプレイの上で体験してみよう、というツールがEigen View Graphics である。なお、グラフィックス画面上での操作ということから、２×２のマトリックスに限らざるを得ないのが残念だがいたし方ない。

プログラムの起動はマトリックスを指定して、例えば次のように行う。

```
    DA1 =: 2 2$_1 4 2 1  NB. 戸田、小野「入門数値計算」オーム社、p.216, A1
    DA1
_1 4
 2 1
    run DA1
```

実際の画面は次のようになる。２つの機能から成っている。



(1) 手動によるマトリックス操作

画面のグラフ上の任意の位置でマウスの左ボタンをクリックすると、その位置と座標原点を結ぶに赤いベクトルが現れる。同時に回転、伸縮のマトリックス操作された結果は青いベクトルとして示される。ふつうは大きさも向きも異なるだろう。入力、出力のベ

クトルの値もそれに応じて表示される。また、下の Magnitude 窓には入出力の拡大率が x 方向、y 方向でそれぞれの値が表示される。

ここで、マウスをいろいろ動かして、いろいろなベクトルでやってみる。そして赤と青とのベクトルが一直線に重なる位置をさがしてみよう。この位置が固有状態で "Eigen State!" と表示される。また、このとき，Magnitude 窓の値は等しくなり、この値が固有値である。さらにベクトルの向きを変えないように、マウスを動かしてみる。固有値は変わらないが、入力ベクトルと出力の固有ベクトルの大きさが変わることが分かるだろう。

(2) ベクトルを回転させながらのマトリックス操作

はじめに適当な位置で赤と青とのベクトルを表示させておく。次に Rotate ボタンを押してみる。すると赤いベクトルの先端は少し回転し、同時に対応した青いベクトルも動く。さらに Rotate ボタンを押すたびに、赤いベクトルは回転し、円弧を描く。変換された青いベクトルは円弧ではなく、楕円を描くだろう。また、その途中では赤、青ベクトルが一直線になった固有状態が見つかるだろう。

４．おわりに

固有値、固有ベクトルの考え方および算法は、理工科はもちろん経済学など文科系にとっても今や欠かすことのできない数学リテラシーといえよう。

かつて固有値、固有ベクトルを多少は扱ってきたものの、筆者自身、真に理解していたとは言えなった。量子化学の本の中でハミルトニアンの波動方程式を解くといった記述に続いて、固有値、固有ベクトルという語が出てきたとき[3]、違和感なくすぐ分かったというばウソになる。実は、J グラフィックスを使ったこのようなツールは筆者自身の固有値理解の確認のためのものでもある。

今回の Eigen View Graphics の実験により、自分の目と手を通して体験することによって、固有値、固有ベクトルとはどういうものかを知り、別に恐れることなし、と誰にでも身近に感じていただければありがたいと思っている。

文献

[1] 中野嘉弘「J 言語と固有値問題」JAPLA 研究会 2007/5/26

[2] 西川利男「J の正規表現プログラミング III－記号処理により行列式を直接展開して固有値を求める」JAPLA 研究会 2007/5/26

[3] G.ストラング、山口昌也監訳「線形代数とその応用」産業図書(1986).

[4] 西川利男「線形代数を APL/J 思考から理解する－その１」J 研究会資料 2005/1/22

[5] 西川利男「線形代数を APL/J 思考から理解する－その２」J 研究会資料 2005/2/26

[6] 米沢貞次郎ら（京都大学福井謙一研究室共同執筆）「量子化学入門（上、下）」化学同人(1963). など

プログラム・リスト

```
NB. Eigen Problem Viewing Graphics
NB.    by Toshio Nishikawa 2007/5/26

wr=: 1!:2&2

require 'gl2'

NB. Problem Data
DA1 =: 2 2$_1 4 2 1          NB. 戸田、小野「入門数値計算」オーム社、p.216, A1
DA2 =: 2 2$_4 1 _1 _2        NB.    〃   〃   A2
DA3 =: 2 2$1 4 _2 1          NB.    〃   〃   A3
DA4 =: 2 2$4 1 1 _2          NB.    〃   〃   A4
DM2 =: 2 2$3 2 4 1           NB. From Nakano, 戸川「数値計算入門」オーム社

NB. eg. run DA1, run DM2, ..
run=: eigen_run

EIGEN=: 0 : 0
pc eigen closeok;pn "Eigen View Graphics";
menupop "File";
menu new "&New" "" "" "";
menu open "&Open" "" "" "";
menusep ;
menu exit "&Exit" "" "" "";
menupopz;
xywh 226 97 34 12;cc ok button;cn "Rotate";
xywh 183 136 34 12;cc cancel button;cn "Exit";
xywh 6 7 168 142;cc eiggraph isigraph;
xywh 181 18 29 26;cc mda listbox;
xywh 181 7 39 10;cc ma static;cn "Matrix";
xywh 181 53 22 26;cc invec listbox;
xywh 225 54 21 31;cc outvec listbox;
xywh 207 63 15 10;cc arrow static;cn "--->";
xywh 182 91 38 30;cc mag listbox;
xywh 181 81 39 10;cc tmag static;cn "Magnitude";
```

```
xywh 182 118 50 16;cc eigmsg listbox;
xywh 225 136 34 12;cc clear button;
pas 6 6;pcenter;
rem form end;
)


NB. Main Program
eigen_run=: 3 : 0
DA =: y.        NB. Input Matrtix Data, global
if. 0 = #DA do. DA =: DA1 end. NB. default data
wd EIGEN
NB. initialize form here
grid ''
wd 'set mda ', ; (inmat DA) ,&. > LF
glshow ''
wd 'pshow;'
)

inmat =: 3 : 0
DA0 =. ": y.
DA1 =. '"',"(1) DA0,"(1) '"'
DA2 =. <"1 DA1
)

eigen_cancel_button=: 3 : 0
wd 'pclose;'
)

NB. Set Vector by Mouse Left Down, then View Eigen Vector
eigen_eiggraph_mbldown=: 3 : 0
d=. ". sysdata
x0=: (0{d) * 1000 % (2{d)
y0=: (1{d) * 1000 % (3{d)
glclear ''
grid ''
```

```
X0 =: ((<.x0) - 500) % 100
Y0 =: ((<.y0) - 500) % 100
Z0 =: 2 1$X0, Y0
Z1 =: DA +/ . * Z0
X1 =: {. , Z1
Y1 =: {: , Z1
DX =: X1 % X0
DY =: Y1 % Y0
x1 =. (X1 * 100) + 500
y1 =. (Y1 * 100) + 500
glrgb 0 0 255
glpen 10 0
glmove x1, y1
gllines x1, y1, 500, 500
glrgb 255 0 0
glpen 10 0
glmove x0, y0
gllines x0, y0, 500, 500
NB. gltextalign TA_BOTTOM
NB. gltext '(', (": X0), ' ,', (": Y0), ')->(', (": X1), ' ,',(": Y1),') : ',
(": DX), ' ,', (": DY)
wd 'set invec ', (":X0), LF, (":Y0)
wd 'set outvec ', (":X1), LF, (":Y1)
wd 'set mag ', (":DX), LF, (":DY)
NB. gltextalign TA_BOTTOM
NB. gltext (": DX),',',(":DY),'=',(": DX-DY)
if. (|DX - DY) < 0.2
  do. wd 'set eigmsg ', '"Eigen State!"'
  else. wd 'set eigmsg ', '"Not Good!"'
end.
glshow ''
X0A =: X0
Y0A =: Y0
X1A =: X1
Y1A =: Y1
)
```

```
NB. Axis and Grid Lines
grid =: 3 : 0
glrgb 0 0 0
glpen 1 0
gllines 0, 500, 1000, 500      NB. x-axis
gllines 500, 0, 500, 1000      NB. y-axis
NB. x-grid
x_grid =. <"1 ((100*>:i.9),.490),"(1) (100*>:i.9),.510
gllines L:0 x_grid
NB. y-grid
y_grid =. <"1 (490,. 100*>:i.9),"(1) (510,. 100*>:i.9)
gllines L:0 y_grid
)

NB. Rotate Vector & View
eigen_ok_button=: 3 : 0
NB. -- input vector ------------------
('r0';'t0') =. *. X0A j. Y0A
tt0 =: t0 + 0.1
('X0B';'Y0B') =. +. r0 r. tt0
'x0a y0a x0b y0b' =. 500 + 100 * (X0A, Y0A, X0B, Y0B)
NB. -- output vector -----------------
Z0A =. 2 1$X0A, Y0A
Z1A =. DA +/ . * Z0A
X1B =: {. , Z1A
Y1B =: {: , Z1A
'x1a y1a x1b y1b' =. 500 + 100 * (X1A, Y1A, X1B, Y1B)
NB. -- output curve ------------------
glrgb 255 255 255
glpen 10 0
gllines x1a,y1a,500,500
glrgb 0 0 255
glpen 10 0
gllines x1a,y1a,x1b,y1b
glpen 1 0
gllines x1b,y1b,500,500
```

```
NB.  --- input curve -----------
glrgb 255 255 255
glpen 10 0
gllines x0a,y0a,500,500
glrgb 255 0 0
glpen 10 0
gllines x0a,y0a,x0b,y0b
glpen 1 0
gllines x0b,y0b,500,500
grid ''
glshow ''
NB. --- show values -------------
DXB =. X1B % X0B
DYB =. Y1B % Y0B
wd 'set invec ', (":X0B), LF, (":Y0B)
wd 'set outvec ', (":X1B), LF, (":Y1B)
wd 'set mag ', (":DXB), LF, (":DYB)
if. (|DXB - DYB) < 0.1
  do. wd 'set eigmsg ', '"Eigen State!"'
  else. wd 'set eigmsg ', '"Not Good!"'
end.
X0A =: X0B
Y0A =: Y0B
X1A =: X1B
Y1A =: Y1B
)

eigen_clear_button=: 3 : 0
glclear ''
grid ''
glshow ''
wd 'set invec ', '"     "'
wd 'set outvec ','"     "'
wd 'set mag ', '"      "'
wd 'set eigmsg ', '"     "'
)
```