

固有値計算の QR 分解法 (Gram-Schmidt 処理) の J プログラム J の 3D グラフィックスによる図形説明とともに

西川利男

2016/03/12

目次

1	QR 分解法と Gram Schmidt 計算－処理のながれ [2],[3],[4]	2
2	Gram-Schmidt 処理の具体例	2
3	Gram-Schmidt の J プログラムとその実行	4
4	J-3D グラフィックスによる図解	7
5	終りに	7
6	J のプログラムリスティング	8

固有値、固有ベクトルは、理系文系を問わず真に使える道具（＝リテラシー）として身に付けるべき線形代数の中心をなす大切なテーマである。しかし、その理解のしかたは、理工学と経済数学とでは違うのではないかと私なりの考えを問うてみた。

特に実用の意味から、数式を使った理論より、数値計算のプログラミングを通して理解を行ってみた。前回 [1] は Power 法、Deflation 法の J プログラムで行った。

[1] 西川利男「固有値計算の Power 法と Deflation 法の J プログラム

主成分分析からの理工学とは違った固有値の理解」JAPLA 研究会資料 2016/1/16

今回は、もう一つの計算法として、Gram Schmidt 処理を用いた QR 分解法をとりあげた。なお、J では言語プリミティブ 128!:1 としてシステム常備されているが、あえてプログラミングしてみることにした。なお、前には気付なかったが、山下氏による報告 [2] があった。さらに J の 3D グラフィックスを用いて、その仕組みを示した。

1 QR 分解法と Gram Schmidt 計算－処理のながれ [2],[3],[4]

まずは、処理の流れを要点のみで記す。数学については文献 [4]などを参照のこと。
正則な（正方行列で、行列式の値が 0 でない） n 行 n 列の行列 A を取り上げる。

1. QR 分解法とは

行列 A を 2 つの行列 Q と行列 R の積に分解する。この方法が Gram-Schmidt である。
ここで、

- 行列 Q は正規直交行列
- 行列 R は右上三角行列

つぎに、

- 行列 Q と行列 R とを入れ換えて
- 行列 $R \cdot Q$

の行列の掛け算を行い、これを行列 A_2 とする。

つぎには、この行列 A_2 を元にさらに正規直交行列と右上三角行列への分解をおこなう。これを何回もおこなうに従って、行列 A の固有値は行列 R の対角要素へと収束して行き、求められる。

2. Gram Schmidt 処理とは（正規直交行列 Q を求める方法）

ここでの処理では、 J の見方からすると、前にも [1] で示したように、行列は動詞というより名詞と見たほうが良い。つまり n 行 n 列で値が並んだ配列 A とみる。そして、これを n 個の列ベクトルがヨコに並んだ集合と見る。それぞれの列ベクトルは n 個の値から成る。Gram Schmidt 処理では、とくに正射影の係数を求める次のようなベクトルの間の内積を用いた計算がポイントになる。

(行ベクトル) ip (列ベクトル) \rightarrow スカラー (単なる数)、 ip は内積

J では $ip =: +/ @: *$ で求められる。また、上の演算は、 J では 1 行*1 列の配列が返されるので、スカラー値にするには、`array_to_scalar =: {.@,` で行う。

[2] 山下紀幸「QR 法により固有値を求める (手計算と J との比較)」JAPLA, 1997/5/4.

[3] 戸田英雄、小野令美「入門数値計算」p.196-203、オーム社 (1983).

[4] Bill Jacob, "Linear Functions and Matrix Theory", Springer(1995).

2 Gram-Schmidt 処理の具体例

ここでは数式ではなく、具体的な数値例を用いて、Gram-Schmidt の計算過程を示す。
配列 A を列ベクトル a_1, a_2, a_3 として扱う。

$$A = \begin{pmatrix} 3 & 1 & 1 \\ 4 & 5 & 2 \\ 0 & 0 & 6 \end{pmatrix} \quad a_1 = \begin{pmatrix} 3 \\ 4 \\ 0 \end{pmatrix} \quad a_2 = \begin{pmatrix} 1 \\ 5 \\ 0 \end{pmatrix} \quad a_3 = \begin{pmatrix} 1 \\ 2 \\ 6 \end{pmatrix}$$

2.1 配列 Q を求める (分解過程)

- k = 1 step

$$\mathbf{u}_1 = \mathbf{a}_1 = \begin{pmatrix} 3 \\ 4 \\ 0 \end{pmatrix} \quad |\mathbf{u}_1| = \sqrt{3^2 + 4^2 + 0} = 5 \quad \mathbf{q}_1 = \frac{\mathbf{u}_1}{|\mathbf{u}_1|} = \frac{1}{5} * \begin{pmatrix} 3 \\ 4 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.6 \\ 0.8 \\ 0 \end{pmatrix}$$

- k = 2 step

ここでは \circ はベクトルの内積でその結果はスカラー (=普通の数値) となる。また $*$ はスカラーとベクトルの単なる掛け算である

$$\begin{aligned} \mathbf{u}_2 &= \mathbf{a}_2 - (\mathbf{a}_2 \circ \mathbf{q}_1) * \mathbf{q}_1 = \begin{pmatrix} 1 \\ 5 \\ 0 \end{pmatrix} - \left((1 \ 5 \ 0) \circ \begin{pmatrix} 0.6 \\ 0.8 \\ 0 \end{pmatrix} \right) * \begin{pmatrix} 0.6 \\ 0.8 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 5 \\ 0 \end{pmatrix} - 4.6 * \begin{pmatrix} 0.6 \\ 0.8 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 5 \\ 0 \end{pmatrix} - \begin{pmatrix} 2.76 \\ 3.68 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} -1.76 \\ 1.32 \\ 0 \end{pmatrix} \end{aligned}$$

$$|\mathbf{u}_2| = \sqrt{(-1.76)^2 + 1.32^2} = 2.2 \quad \mathbf{q}_2 = \frac{\mathbf{u}_2}{|\mathbf{u}_2|} = \frac{1}{2.2} * \begin{pmatrix} -1.76 \\ 1.32 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.8 \\ 0.6 \\ 0 \end{pmatrix}$$

- k = 3 step

$$\begin{aligned} \mathbf{u}_3 &= \mathbf{a}_3 - (\mathbf{a}_3 \circ \mathbf{q}_1) * \mathbf{q}_1 - (\mathbf{a}_3 \circ \mathbf{q}_2) * \mathbf{q}_2 = \begin{pmatrix} 1 \\ 2 \\ 6 \end{pmatrix} - \left((1 \ 2 \ 6) \circ \begin{pmatrix} 0.6 \\ 0.8 \\ 0 \end{pmatrix} \right) * \begin{pmatrix} 0.6 \\ 0.8 \\ 0 \end{pmatrix} \\ &\quad - \left((1 \ 2 \ 6) \circ \begin{pmatrix} -0.8 \\ 0.6 \\ 0 \end{pmatrix} \right) * \begin{pmatrix} -0.8 \\ 0.6 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 6 \end{pmatrix} \end{aligned}$$

$$|\mathbf{u}_3| = 6 \quad \mathbf{q}_3 = \frac{\mathbf{u}_3}{|\mathbf{u}_3|} * \begin{pmatrix} 0 \\ 0 \\ 6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\mathbf{Q} = \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3 = \begin{pmatrix} 0.6 & -0.8 & 0 \\ 0.8 & 0.6 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

配列 Q はベクトル $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$ を横に繋いで得られる

2.2 配列 R を求める (合成過程)

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{pmatrix}$$

$$r_{11} = |u_1| = 5$$

$$r_{12} = a_2 \circ q_1 = (1 \ 5 \ 0) \circ \begin{pmatrix} 0.6 \\ 0.8 \\ 0 \end{pmatrix} = 4.6$$

$$r_{13} = a_3 \circ q_1 = (1 \ 2 \ 6) \circ \begin{pmatrix} 0.6 \\ 0.8 \\ 0 \end{pmatrix} = 2.2$$

$$r_{22} = |u_2| = 2.2$$

$$r_{23} = a_2 \circ q_2 = (1 \ 5 \ 0) \circ \begin{pmatrix} -0.8 \\ 0.6 \\ 0 \end{pmatrix} = -0.5$$

$$r_{33} = |u_3| = 6$$

$$R = \begin{pmatrix} 5 & 4.6 & 2.2 \\ 0 & 2.2 & -0.5 \\ 0 & 0 & 6 \end{pmatrix}$$

2.3 配列 A の QR 分解

$$A = \begin{pmatrix} 3 & 1 & 1 \\ 4 & 5 & 2 \\ 0 & 0 & 6 \end{pmatrix} = \begin{pmatrix} 0.6 & -0.8 & 0 \\ 0.8 & 0.6 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 5 & 4.6 & 2.2 \\ 0 & 2.2 & -0.5 \\ 0 & 0 & 6 \end{pmatrix}$$

3 Gram-Schmidt の J プログラムとその実行

DT31

3 1 1

4 5 2

0 0 6

途中経過を示した J プログラム gramsq は最後にあげてある。

gramsq DT31

k=1 =====

3

4

0

```

Uab =5
0.6
0.8
0
k=2 =====
1
5
0
0.6 0.8 0
coef=4.6
_1.76
1.32
0
Uab =2.2
Q2: ----
_0.8
0.6
0
k=3 =====
1
2
6
0.6 0.8 0
coef=2.2
_0.32
0.24
6
j-repeat:1
1
2
6
_0.8 0.6 0
coef=0.4
_4.996e_16
2.498e_16
6
Uab =6

```

```

Q3: ----
_8.32667e_17
 4.16334e_17
          1
0.6 _0.8 0
0.8  0.6 0
  0   0 1
== Q Matrix: ==
0.6 _0.8 0
0.8  0.6 0
  0   0 1
R Matrix =====
--0 1
1 5 0
0.6 0.8 0
4.6
--0 2
1 2 6
0.6 0.8 0
2.2
--1 2
1 2 6
_0.8 0.6 0
0.4
== R Matrix: ==
5 4.6 2.2
0 2.2 0.4
0  0  6
*** end ***

```

Gram-Schmidt 実行の動詞 `gramsm` と QR 法による固有値計算 `eigen_value` など J プログラムは最後にあげた。なお、J のプリミティブ `128!:1` でも同じ結果を与える。

```

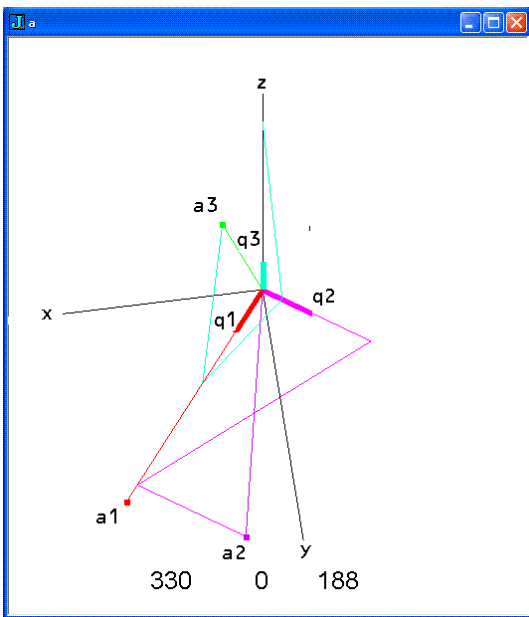
gramsm DT31
+-----+-----+
|0.6 _0.8 0|5 4.6 2.2|
|0.8  0.6 0|0 2.2 0.4|
|  0   0 1|0  0  6|

```

+-----+-----+

eigen_value DT31
6.23607 1.76393 6

4 J-3D グラフィックスによる図解



列ベクトル a_1, a_2, a_3 から、 Q 行列を成す列ベクトル q_1, q_2, q_3 への変換過程を示す。
次々に投影をとり、ベクトルが方向を変えて行き、 q_1, q_2, q_3 は直交することに注意。
3次元内のベクトルの移動過程が示される。

この J-3D グラフィックスでは、図の回転、移動、拡大、縮小が自由にできる。そして、あらゆる視点からの観察、理解が可能である。

5 終りに

線形代数は、単なる行列の計算法ではない。配列なる数値の集合をベクトルとその間の行列計算を通して、空間内に表示し、目に見える形で理解する道具である。

つまり、線形代数とは現代のデジタル解析幾何学である。

ここで、われわれの目の前にある J システムは、ベクトルをそのままの形で演算することが出来る。

さらに J の 3D グラフィックスと合わせて線形代数にとって、またとない強力なツールであるということが出来よう。

6 Jのプログラムリスティング

```
NB. 戸田、小野「入門数値計算」p.196 =====
DT3 =: 3 3$3 1 5 4 5 10 0 0 30
DT31 =: 3 3$3 1 1 4 5 2 0 0 6
```

```
NB. QR calc on 戸田、小野「入門数値計算」 p.196
NB. programed by T. Nishikawa 2016/2/2
NB. Usage: gramsm DT3 => Q-matrix and R-matrix
```

```
gramsm =: 3 : 0
A =. y.
N =. {. $A
NB. Q Matrix Calc. =====
i =. 0
while. i < N
  do.
    if. i = 0 do.
      U =. (N, 1)$ i {"(1) A
      Uab =. (%:@(+/@: *:)), U
      Uabs =. Uab
      Q =. U % Uab
      goto_next.
    end.
    if. i > 0
      do.
        U =. (N, 1)$ i {"(1) A
        j =. 0
        while. j < i
          do.
            Aj =. (N, 1)$ i {"(1) A
            Qj =. (j) {"(1) Q
            COEF =. ({.@,)(, Aj) ipn Qj
            U =. U - COEF * Qj
            j =. j + 1
          end.
        Uab =. (%:@(+/@: *:)), U
```



```

        Uabs =. Uabs, Uab
        Qn =. U % Uab
        Qn =. cleanz Qn
        Q =. Q ,"(1) Qn
    end.
label_next.
    i =. i + 1
end.
NB. R Matrix Calc. =====
R =. (N, N)$0
i =. 0
while. i < N
do.
    j =. 0
    while. j < N
    do.
        if. j = i do.
            R =. (j{Uabs) (<i, j) } R
        end.
        if. i < j do.
            AR =. j {"(1) A
            QR =. i {"(1) Q
            R =. (AR ipn QR) (<i, j) } R
        end.
        j =. j + 1
    end.
    i =. i + 1
end.
NB. Q-matrix and R-matrix Calculated =====
Q;R
)

diag =: i.@# {"_1 ]

NB. QR method =====
qr =: 3 : 0
NB. A1 =. (128!:0) y.      using J primitive

```

```

A1 =. grmsm y.          NB. using J program by T. Nishikawa
((>@{:) (+/ . *) (>@{.)) A1
)

```

```

eigen_value =: 3 : 0
40 eigen_value y.
:
diag qr^:x. y.
)

```

```

NB. 3D graphics ? Gram-Schmidt Process =====
load 'trig numeric'
require 'gl3'

```

```

NB. import from jzopengl
load 'jzopengl'

```

```

NB. e.g. coinsert 'jzopengl' =====
coinsert=: 3 : 0
n=. ;: :: ] y.
p=. ; (, 18!:2) @ < each n
p=. ~. (18!:2 coname''), p
(p /: p = <,'z') 18!:2 coname''
)

```

```

coinsert 'jzopengl'

```

```

A=: noun define
pc a closeok;
xywh 0 0 340 300;cc g isigraph ws_clipchildren ws_clipsiblings rightmove bottommove
pas 0 0;
rem form end;
)

```

```

run=: a_run
a_run=: verb define
JOB =: y.
wd A

```

```

glaRC''
NB. R =: 120 0 0
R =: 330 0 188
NB. R =: 90 0 0
NB. T =: 0 0 0
T =: _1.5 _0.5 _6 NB. rl/ud/nf
NB. SC =: 0.375
SC =: 1
glafont 'arial 30'
glusefontbitmaps 0 32 26 32
setfocus g
wd 'pshow;ptop'
)

```

```

a_g_size=:verb define
wh=.glqwh''
glviewport 0 0,wh
glmatrixmode GL_PROJECTION
glloadidentity''
glortho _6, 6, _6, 6, _2, 10
NB. gluPerspective 45, (%/wh),1 8
)

```

NB. Key-In Command =====

```

a_g_char =: verb define
R =: 360 | R + 2 * 'xyz' = 0 { sysdata
R =: 360 | R - 2 * 'XYZ' = 0 { sysdata
T =: T + 0.5 * 'run' = 0 { sysdata NB. r(right) = +x, u(up) = +y, n(near) = +z
T =: T - 0.5 * 'ldf' = 0 { sysdata NB. l(left) = -x, d(down) = -y, f(far) = -z
SC =: SC * 1 + 0.25 * 'b' = 0 { sysdata NB. bigger
SC =: SC * 1 - 0.25 * 's' = 0 { sysdata NB. smaller
glpaintx''
)

```

NB. Paint on the Display =====

```

a_g_paint =: verb define
glClearColor 1 1 1 1

```

```

glClear GL_COLOR_BUFFER_BIT + GL_DEPTH_BUFFER_BIT
glEnable GL_DEPTH_TEST
glMatrixMode GL_MODELVIEW
glLoadIdentity''
glTranslate T
glRotate R ,. 3 3 $ 1 0 0 0
glScale 3#SC
gramsm1 '' NB. Gram Schmidt Process Display
drawtext ''
glSwapBuffers ''
)

```

```

NB. Write Data on the display =====
NB. indicate rotated angle x, y, z in degree
drawtext =: verb define
glMatrixMode GL_MODELVIEW
glLoadIdentity ''
glColor 0 0 0 1 NB. color is determined by glMaterial Command
glRasterPos _3 _4 _2 NB. be carefull, not change position
glCallLists 10 ": R
glpaintx''
)

```

```

gramsm1 =: 3 : 0
NB. X, Y, Z axes =====
glLineWidth 1
glBegin GL_LINES
  glColor 0 0 0 1 NB. X-axis
  glVertex 0 0 0
  glVertex 6 0 0
  glColor 0 0 0 1 NB. Y-axis
  glVertex 0 0 0
  glVertex 0 6 0
  glColor 0 0 0 1 NB. Z-axis
  glVertex 0 0 0
  glVertex 0 0 7

```

```

glEnd ''

NB. Data Display =====
NB. Using DT31 Data
NB. k=1 step
glColor 1 0 0 1
glPointSize 6
glBegin GL_POINTS
  glVertex 3 4 0
glEnd ''
glLineWidth 1
glBegin GL_LINES
  glVertex (L:0) 3 4 0;0 0 0
glEnd ''
glLineWidth 6
glBegin GL_LINES
  glVertex (L:0) 0.6 0.8 0;0 0 0 NB. Q1
glEnd ''

NB. k=2 step
glColor 0.8 0 1 1 NB. A2
glPointSize 6
glBegin GL_POINTS
  glVertex 1 5 0
glEnd ''
glLineWidth 1
glBegin GL_LINES
  glVertex (L:0) 1 5 0;0 0 0
  glColor 1 0 1 1
  glVertex (L:0) 1 5 0;2.76 3.68 0
  glVertex (L:0) 2.76 3.68 0;_1.76 1.32 0
  glVertex (L:0)_1.76 1.32 0;0 0 0
glEnd ''
glLineWidth 6
glBegin GL_LINES
  glVertex (L:0) _0.8 0.6 0;0 0 0 NB. Q2
glEnd ''

```

```

NB. k=3 step
glColor 0 1 0 1 NB. A3
glPointSize 6
glBegin GL_POINTS
  glVertex 1 2 6
glEnd ''
glLineWidth 1
glBegin GL_LINES
  glVertex (L:0) 1 2 6;0 0 0
  glColor 0 1 0.8 1
  glVertex (L:0) 1 2 6;0 0 0
  glVertex (L:0) 1 2 6;1.32 1.76 0 NB. <a3, q1>*q1
  glVertex (L:0) 1.32 1.76 0; _0.32 0.24 0 NB. <a3, q2>*q2
  glVertex (L:0) _0.32 0.24 0;0 0 6
glEnd ''
glLineWidth 6
glBegin GL_LINES
  glVertex (L:0) 0 0 1;0 0 0 NB. Q3
glEnd ''
)

```