固有値ソルバLOBPCG法の解説

永井佑紀

平成 26 年 10 月 10 日

1　はじめに

Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) 法と呼ばれる、行列の数値的対角化手法について解説する。最初、一つの最小固有値を求める手法について解説したあと、m 個の固有値を求める場合について述べ、最後に Fortran90 のサンプルコードを載せる。

2　固有値問題と最小化問題

固有値問題

$$Ax_i = \lambda_i x_i \tag{1}$$

を解きたい。ここで、最小固有値だけを求めたい場合を考える。このとき、レイリー商：

$$\mu(x) = \frac{x^T A x}{x^T x} \tag{2}$$

を最小化する x を求めたとき、$\mu(x)$ が最小固有値になることを示す。一般の x は固有ベクトルで展開する事ができ、

$$x = \sum_i \alpha_i x_i \tag{3}$$

と書ける。ここで、規格化 $x^T x = 1$ されているとする。これをレイリー商に代入すると、

$$\mu(x) = \frac{\left(\sum_i \alpha_i x_i^T\right)\left(\sum_j \alpha_j A x_j\right)}{\sum_i |\alpha_i|^2} = \sum_i |\alpha_i|^2 \lambda_i \tag{4}$$

となる。よって、$\mu(x)$ が最小になるのは、x が最小固有値 $\lambda_{\min}$ の固有ベクトルとなっているときで、そのときの値は最小固有値である。以上から、$\mu(x)$ が極小となる x を求めれば、それが最小固有値の固有ベクトルになっていることがわかる。$\mu(x)$ の極小値は、$\mu(x)$ をベクトル x で変分したものがゼロ：

$$\frac{\delta\mu(x)}{\delta x} = 2\frac{Ax}{x^T x} - 2\frac{(x^T A x)}{x^T x}x = 0 \tag{5}$$

となればよい。これは、整理すると、

$$Ax - \mu(x)x = 0 \tag{6}$$

という条件となる。したがって、「残差ベクトル」として

$$r(x) = Ax - \mu(x)x \tag{7}$$

を定義すると、この $r(x)$ の大きさが非常に小さくなった時、その時の x が最小固有値の固有ベクトルである。

上のレイリー商と固有値の関係については、固有値がすべて実数である時に成り立っている。したがって、A は実対称行列あるいはエルミート行列でなければならない。以下では、すべて実対称行列の場合を考える。

3　固有ベクトル探索

3.1　低次元空間での探索

次に、どのように固有ベクトルを探索するか、について述べる。まず、$n \times n$ の行列 A の固有ベクトルは n 次元空間中のベクトルであることに着目しよう。このベクトルを x^* とすると、x^* は適当な線形独立な n 本のベクトル v_i によって、

$$x^* = \sum_{i=1}^{n} a_i v_i \tag{8}$$

と書ける。このベクトル x^* は、

$$r(x^*) = 0 \tag{9}$$

を満たす。

さて、ここで、線形独立な m 本のベクトルが用意されているとする。ここで $n > m$ である。当然のことながら、一般の場合には、x^* は m 本のベクトルからは作る事ができない。次元が足りないからである。たとえば、$n = 3$ で $m = 2$ の場合、三次元空間中に存在するベクトル x^* は、適当な二本のベクトルで張って作られる二次元平面と同一平面上には存在しない。しかし、注意深く二本のベクトルを選ぶ事ができれば、x^* を同一平面にもっていくことができる。上記を言い換えると、m 本の線形独立ベクトル u_i によって張られる空間において、

$$r(x_k) = 0 \tag{10}$$

となるベクトル x_k は一般的には存在しない、ということである。しかしながら、なんとかして m 本のベクトルで張った空間にベクトル x^* を含ませたい。そのためには、選んだ m 本のベクトルを改良して、別の m 本のベクトルを用意し、これを繰り返し行ってどんどん改良する事で、最終的にベクトル x^* を含む空間を張る m 本のベクトルを見つければよい。

3.2　低次元基底の改良

具体的にはどのように改良するかが問題である。いま、あるベクトル x が用意した m 本のベクトル v_i で張られる空間に存在する：

$$x = \sum_{i=1}^{m} b_i v_i \tag{11}$$

とする。これは

$$[x]_l = \sum_{i=1}^{m} [v_i]_l b_i \tag{12}$$

と表示する事で、$n \times m$ 行列の $V_{li} = [v_i]_l$ と m 次元ベクトル b を用いて

$$x = Vb \tag{13}$$

を書く事ができる。このベクトルを用いて $\mu(x)$ を計算すると

$$\mu(x) = b^T V^T A V b \tag{14}$$

$$= b^T A' b = \mu'(b) \tag{15}$$

となる。ここで、$A' \equiv V^T AV$ とした。さて、$\mu'(b)$ は m 次元空間中のベクトル b の関数であり、これは行列 A' のレイリー商とみなす事ができるので、最小化するベクトル b は、固有値問題：

$$A'b = \lambda' b \tag{16}$$

の最小固有値 λ' の固有ベクトルである。すなわち、考えている低次元基底の中で一番求めたい固有ベクトル x^* に近いのが、この m 次元空間での固有ベクトル b を使ったベクトル x_1 である。したがって、この固有ベクトル x_1 と、その他の $m-1$ 本の線形独立なベクトルを用いて新しく m 本の基底を用意すれば、その基底が張る新しい空間内で固有ベクトル x^* に近いベクトルを求めることができて、これを繰り返すと最終的には固有ベクトル x^* を含む低次元空間が得られる。

3.3 LOBPCG 法の場合

LOBPCG 法の場合、$m = 3$ である。一本目は、適当なベクトル x_0 である。このベクトル x_0 は固有ベクトル x^* とは当然異なっている。すなわち、このベクトル x_0 を用いた $\mu(x_0)$ は最小値ではない。二本目のベクトルは、ベクトル x_0 を修正してより小さな $\mu(x)$ を得られるようなベクトルが望ましい。LOBPCG 法では、二本目のベクトルとして $\mu(x^0)$ の勾配ベクトル (残差ベクトル):

$$\frac{\delta\mu(x)}{\delta x}\Big|_{x=x_0} = r(x_0) \tag{17}$$

を用いる。これは、$\mu(x_0)$ が最も急に変化する方向であり、このベクトルを含めた空間であれば $\mu(x_0)$ よりも小さな $\mu(x)$ を持つ x が得られる。三本目のベクトルは、x_0 および $r(x_0)$ とは線形独立な「修正ベクトル」p_0 を用意する。LOBPCG 法登場以前の共役勾配法による固有値計算では、この修正ベクトルとして共役ベクトルを用いていたため、LOBPCG は CG という言葉入っているが、実際の LOBPCG 法の計算では共役ベクトルを用いていない。LOBPCG 法では、p_0 は x_0 および $r(x_0)$ と線形独立でありさえすればよい。この三本のベクトルを用いて張った空間中で、$\mu(x)$ を最小化する。ただし、三本のベクトルは直交化していなければならないので、グラムシュミット等で直交化しておく。そのため、

$$V = \begin{pmatrix} x_0 & r(x_0) & p_0 \end{pmatrix} \tag{18}$$

という行列を用意し、$A' \equiv V^T AV$ を計算して、

$$A'b = \lambda' b \tag{19}$$

という固有値問題の固有ベクトルを求め、最小固有値 $\lambda_{\min}$ の固有ベクトル $b_{\min}$ を用いて

$$x_1 = Vb_{\min} \tag{20}$$

$$= [b_{\min}]_1 x_0 + [b_{\min}]_2 r(x_0) + [b_{\min}]_3 p_0 \tag{21}$$

を計算し、次の一本目のベクトルとする。そして、二本目のベクトルは $r(x_1)$ であり、これは定義より

$$r(x_1) = Ax_1 - \mu(x_1)x_1 \tag{22}$$

$$= Ax_1 - \lambda'_{\min} x_1 \tag{23}$$

と計算できる。そして、三本目のベクトル p_1 は線形独立になるように、

$$p_1 = [b_{\min}]_2 r(x_0) + [b_{\min}]_3 p_0 \tag{24}$$

と選ぶ。

以下にアルゴリズムを示す。

1. 初期ベクトル x_0 を用意する。対応する残差ベクトル $r(x_0)$ を用意する。最初の修正ベクトルは零ベクトル $p_0 = 0$ とする。
2. 三本のベクトルは直交している必要があるので、グラムシュミット等の方法で正規直交化する。ただし $p_0 = 0$ の最初は二本のベクトルを直交化する。
3. $V = (x_0, r(x_0), p_0)$ を用意し、$A' = V^T AV$ に対する固有値問題 $A'b = \lambda' b$ を求める。
4. 上記の最小固有値に属する固有ベクトル $b_{\min}$ を用いて、$x_1 = Vb_{\min}$、$r(x_1)$、および $p_1 = (0, r(x_0), p_0)b_{\min}$ の三本のベクトルを求める。
5. 残差ベクトル $r(x_k)$ が十分に小さくなるまで、2 から 4 を繰り返す。

4 複数固有ベクトル探索

LOBPCG 法では、最小固有値の固有ベクトルだけではなく、その上の固有値も求める事ができる。m 本の固有ベクトルを求める場合には、その m 本の固有ベクトルが含まれる低次元基底を考える必要がある。よって、基底は 3 本ではなく、$3m$ 本用意する。つまり、初期ベクトルを m 本、対応する残差ベクトルを m 本、修正ベクトルを m 本である。一つのベクトル x^i に対し残差ベクトルは一つ決まる $r(x^i)$。また、修正ベクトルも元と同様に決める事ができる。この $3m$ 次元空間の基底を改良しながら、m 本の固有ベクトルを求める。また、それぞれのベクトルは

$$X = \begin{pmatrix} x^1 & \cdots & x^m \end{pmatrix} \tag{25}$$

$$R = \begin{pmatrix} r(x^1) & \cdots & r(x^m) \end{pmatrix} \tag{26}$$

$$P = \begin{pmatrix} p^1 & \cdots & p^m \end{pmatrix} \tag{27}$$

$$V = \begin{pmatrix} X & R & P \end{pmatrix} \tag{28}$$

と束ねておくと、アルゴリズムを簡便に記述できる。最初の初期ベクトルは、なるべく求めたい空間に近い空間を張ってほしいので、一回目は m 次元固有値問題

$$X_0^T A X_0 B = B\Lambda \tag{29}$$

を満たす B と Λ(固有値が格納された対角行列) を求めて、

$$X_1 = X_0 B \tag{30}$$

としておく。これは、初期ベクトルで張られた空間の中で最適なものを探していることに相当している。次に、R を

$$R_1 = AX_1 - X_1\Lambda \tag{31}$$

で計算する。そして、今度は初期ベクトルと残差ベクトルで張られた $2m$ 次元空間の中での最適なベクトルを求める。その際、グラムシュミットで直交化しておく。そして、$2m$ 次元固有値問題

$$\begin{pmatrix} X & R \end{pmatrix}^T A \begin{pmatrix} X & R \end{pmatrix} B = B\Lambda \tag{32}$$

を解き、下から m 個の固有値と固有ベクトルからなる行列:

$$B^{\mathrm{m}} = \begin{pmatrix} b_1 & \cdots & b_m \end{pmatrix} \tag{33}$$

$$\Lambda^{\mathrm{m}} = \mathrm{diag}\,(\lambda_1 \cdots \lambda_m) \tag{34}$$

を作成し、

$$X_2 = \begin{pmatrix} X & R \end{pmatrix} B^{\mathrm{m}} \tag{35}$$

$$R_2 = AX_2 - X_2\Lambda^{\mathrm{m}} \tag{36}$$

を計算する。そして、修正ベクトルとして、

$$P_2 = \begin{pmatrix} 0 & R \end{pmatrix} B^{\mathrm{m}} \tag{37}$$

を用いる。こうして作られた X_2、R_2、P_2 を用いて、直交化したあとに、V_2 を作り、$3m$ 次元の固有値問題

$$V_2^T AVB = B\Lambda \tag{38}$$

を解き、下から m 個の固有値固有ベクトルを用いて

$$B^{\mathrm{m}} = \begin{pmatrix} b_1 & \cdots & b_m \end{pmatrix} \tag{39}$$

$$\Lambda^{\mathrm{m}} = \mathrm{diag}\,(\lambda_1 \cdots \lambda_m) \tag{40}$$

を作成する。そして、

$$X_3 = VB^{\mathrm{m}} \tag{41}$$

$$R_3 = AX_3 - X_3\Lambda^{\mathrm{m}} \tag{42}$$

$$P_3 = \begin{pmatrix} 0 & R_2 & P_2 \end{pmatrix} B^{\mathrm{m}} \tag{43}$$

という新しい $3m$ 本のベクトルを作る。以下を繰り返し、残差ベクトルがすべて要求する精度の範囲内で零となれば、そのときの X が固有ベクトルとなっている。

5　サンプルプログラム

サンプルプログラムを示す。このプログラムでは、まずLOBPCG法で固有値を求めた後、比較にためにLAPACKで全対角化を行っている。行列ベクトル積は、サブルーチンtestsubAxで定義されており、ここを変更することで異なる行列に変える事ができる。サンプルの行列は、一次元強束縛模型のハミルトニアンであり、最小値は-2である。また、直交化には修正グラムシュミット法を用いた。コンパイルは、ifortの場合は

```
ifort test_LOBPCG.f90 -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5 -lpthread
```

で行うことができる。

このプログラムでの行列ベクトル積の回数は、参考文献にあるように、もっと減らせるはずである。

ソースコード 1: test_LOBPCG.f90

```
1  !,!,-------------------------------------------------------
2  !,!, ,This, ,is, ,the, ,sample, ,code, ,for, ,LOBPCG
3  !,!, ,Yuki, ,Nagai,,, ,Ph,., ,D
4  !,!, ,10/10/2014, ,(,dd,/,mm,/,yyyy,)
5  !,!, ,This, ,program, ,needs, ,LAPACK
6  !,!,-------------------------------------------------------
7
8  module LOBPCG
9  contains
10
11   subroutine calcLOBPCG(n,sub_Ax,md,x,lam,eps)
12     implicit none
13     interface
14       subroutine sub_Ax(n,x,Ax)
15         integer,intent(in)::n
16         real(8),intent(in)::x(1:n)
```

```
17            real(8),intent(out)::Ax(1:n)
18          end subroutine sub_Ax
19        end interface
20        integer,intent(in)::n,md
21        real(8),intent(out)::lam(1:md)
22        real(8),intent(out)::x(1:n,1:md)
23        real(8),intent(in)::eps
24        real(8)::xtemp(1:n,1:md),r(1:n,1:md),p(1:n,1:md)
25        real(8)::xhx(1:md,1:md),v(1:3*md,1:3*md)
26        real(8)::z(1:n,1:3*md),ztemp(1:n,1:3*md),zhz(1:3*md,1:3*md),zlam(1:3*md),zv(1:3*md,1:3*md)
27        real(8)::d
28        integer::i,j,ii,jj,ite,itemax,nz
29        real(8)::reps(1:md),norm
30
31        itemax = 100000*2
32  ,!, ,eps, ,=, ,1,d,-6
33
34        call random_number(x)
35        call gram_real(n,md,x,xtemp)
36
37        do ii = 1,md
38          call sub_Ax(n,xtemp(1:n,ii),x(1:n,ii)) ,!,x, ,=, ,A, ,xtemp
39        end do
40
41        xhx = matmul(transpose(xtemp),x)
42        call eigensystem_real(xhx,lam,v(1:md,1:md),md)
43
44        x = matmul(xtemp,v(1:md,1:md))
45        r = 0d0
46        do ii = 1,md
47          r(1:n,ii) = x(1:n,ii)*lam(ii)
48        end do
49  ,!, ,write,(*,*), ,sum,(,abs,(,r,))
50        do ii = 1,md
51          call sub_Ax(n,x(1:n,ii),xtemp(1:n,ii)) ,!,xtemp, ,=, ,A, ,x
52        end do
53        r = xtemp - r
54        p = 0d0
55
56        loopite: do ite = 1,itemax
57          z(1:n,1:md) = x(1:n,1:md)
58          z(1:n,1+md:2*md) = r(1:n,1:md)
59          z(1:n,1+2*md:3*md) = p(1:n,1:md)
60          ztemp = z
61  ,!, ,if,(,ite, ,==, ,1, ,.,or,., ,mod,(,ite,,10), ,==0), ,then
62  ,!, ,p, ,=, ,0,d0
63          if(ite == 1 ) then
64            nz = md*2
65          else
66            nz = md*3
67          end if
68          call gram_real(n,nz,z(1:n,1:nz),ztemp(1:n,1:nz))
69
70          do ii = 1,nz
71            call sub_Ax(n,ztemp(1:n,ii),z(1:n,ii)) ,!,z, ,=, ,A, ,ztemp
72          end do
73          zhz(1:nz,1:nz) = matmul(transpose(ztemp(1:n,1:nz)),z(1:n,1:nz))
74          call eigensystem_real(zhz(1:nz,1:nz),zlam(1:nz),zv(1:nz,1:nz),nz)
75
76          v = zv(1:nz,1:md)
77          lam(1:md) = zlam(1:md)
78          x(1:n,1:md) = matmul(ztemp(1:n,1:nz),zv(1:nz,1:md))
79          r = 0d0
80          do ii = 1,md
81            r(1:n,ii) = x(1:n,ii)*lam(ii)
82          end do
83          do ii = 1,md
84            call sub_Ax(n,x(1:n,ii),xtemp(1:n,ii)) ,!,xtemp, ,=, ,A, ,x
85          end do
86          r = xtemp -r
87          do ii = 1,md
88            reps(ii) = sqrt(sum(abs(r(1:n,ii))**2))
89          end do
90          if(eps > maxval(reps)) then
```

```
91              write(*,*) ite,maxval(reps)
92              do ii = 1,md
93                  norm = sqrt(sum(abs(x(1:n,ii))**2))
94                  x(1:n,ii) = x(1:n,ii)/norm
95              end do
96              exit loopite
97          end if
98          if(mod(ite,1000)==0) write(*,*) ite,maxval(reps)
99          if(ite == 1) then
100             p(1:n,1:md) =matmul(r(1:n,1:md),v(1+md:2*md,1:md))
101         else
102             p(1:n,1:md) =matmul(r(1:n,1:md),v(1+md:2*md,1:md))+matmul(p(1:n,1:md),v(1+2*md:3*md,1:
                    md))
103         end if
104     end do loopite
105     return
106   end subroutine calcLOBPCG
107
108   subroutine eigensystem_real(mat_h,lam,p,nb)
109     implicit none
110     integer,intent(in)::nb
111     real(8),intent(in)::mat_h(1:nb,1:nb)
112     real(8),intent(out)::lam(1:nb)
113     real(8),intent(out)::p(1:nb,1:nb)
114     integer::lwork
115     integer info
116     real(8):: work(nb*4)
117
118     lwork = nb*4
119     p = mat_h
120     call dsyev('V','U',nb,p,nb,lam,work,lwork,info)
121
122     return
123   end subroutine eigensystem_real
124
125   subroutine gram_real(m,n,mat_v,mat_v_out)
126     implicit none
127     integer,intent(in)::m,n
128     real(8),intent(in)::mat_v(1:m,1:n)
129     real(8),intent(out)::mat_v_out(1:m,1:n)
130     integer::i,j
131     real(8)::v(1:m),nai,vi(1:m),viold(1:m)
132     real(8)::norm
133     mat_v_out = mat_v
134     do i = 1,n
135         viold = mat_v_out(:,i)
136         do j = 1,i-1
137             nai = dot_product(mat_v_out(1:m,j),viold(1:m))
138             vi = viold - nai*mat_v_out(:,j)
139             viold = vi
140         end do
141         norm = sqrt(dble(dot_product(viold,viold)))
142         mat_v_out(:,j) = viold/norm
143     end do
144     return
145   end subroutine gram_real
146
147 end module LOBPCG
148
149 program main
150   use LOBPCG,only:calcLOBPCG,eigensystem_real
151   implicit none
152   integer,parameter::n = 10000
153   integer,parameter::md=1
154   real(8)::x(1:n,1:md),lam(1:md)
155   real(8),allocatable::A(:,:),lamA(:),p(:,:)
156   integer::i,j
157   real(8)::eps
158   integer t1, t2, t_rate, t_max, diff
159   external testsubAx
160   eps = 1d-6
161
162   call system_clock(t1)
163   call calcLOBPCG(n,testsubAx,md,x,lam,eps)
```

```
164    call system_clock(t2, t_rate, t_max)
165    if ( t2 < t1 ) then
166      diff = t_max - t1 + t2
167    else
168      diff = t2 - t1
169    endif
170    write(*,*) "LOBPCG",diff/dble(t_rate),"[s]"
171    do i = 1,md
172       write(*,*) i,lam(i)
173    end do
174
175    allocate(A(1:n,1:n))
176    allocate(lamA(1:n),p(1:n,1:n))
177    A = 0d0
178    do i = 1,n
179       j = i
180       A(i,j) =0d0*2d0
181       j = i -1
182       if(j >= 1) A(i,j) = 1d0
183       j = i +1
184       if(j <= n) A(i,j) = 1d0
185    end do
186
187    call system_clock(t1)
188    call eigensystem_real(A,lamA,p,n)
189    call system_clock(t2, t_rate, t_max)
190    if ( t2 < t1 ) then
191      diff = t_max - t1 + t2
192    else
193      diff = t2 - t1
194    endif
195    write(*,*) "LAPACK",diff/dble(t_rate),"[s]"
196
197    do i = 1,md
198       write(*,*) lam(i),lamA(i)
199    end do
200
201  end program main
202
203  subroutine testsubAx(n,x,Ax)
204    implicit none
205    integer,intent(in)::n
206    real(8),intent(in)::x(1:n)
207    real(8),intent(out)::Ax(1:n)
208    integer::i,j
209    Ax = 0d0
210    do i = 1,n
211       j = i
212       Ax(i) = Ax(i) + 0d0*2d0*x(j)
213       j = i-1
214       if(j >= 1) Ax(i) = Ax(i) + 1d0*x(j)
215       j = i+1
216       if(j <= n) Ax(i) = Ax(i) + 1d0*x(j)
217    end do
218
219    return
220  end subroutine testsubAx
```

参考文献

Knyazev, Andrew V. (2001), "Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method", SIAM Journal on Scientific Computing 23 (2): 517-541, doi:10.1137/S1064827500366124

S. Yamada, T. Imamura, and M. Machida, "量子大規模固有値問題における共役勾配法の収束性:適応的シフト前処理の収束性の評価", 日本計算工学会論文集 Vol. 2006 (2006) P 20060027