数理解析研究所講究録
422巻 1981年 78-103

78

対称帯行列固有値解析
——ページスワップに注目して——

日立 中研 村田健郎

1. はじめに. 最近，計算機の内部演算能力の向上にくらべて、ページスワップ時間の短縮の方には見るべき進歩がないために、待ち時間ネックが益々目立ってきた。流れ場や、電磁場の偏微分方程式を有限要素法で解くとか、固有値問題を解こうとかのとき特に著しい。ここでは、代表的な対称帯行列固有値解析アルゴリズムについての、そういう観点からの見直しを行い、併せてそのうちの二つ，すなわちスツルム・逆反復法系統に属するものと帯QR法についての改良アルゴリズムを提示したい。

初めに、周知のスツルム・逆反復法を例にとって、問題の捉え方を例示したい．対称帯行列固有値問題 $Ax=\lambda x$ について考える．帯半巾をm，元数をnとする．言葉を節約するためにAは正定値とする．与えられた数αより小さい固有値と固有ベクトル：

$$0<\lambda_1\leqq\lambda_2\leqq\cdots\leqq\lambda_p<\alpha$$

$$v_1,\ v_2,\ \cdots,\ v_p$$

を求めたいとする．大筋を示そう．

〔1〕 $A-\alpha I$ を U^TDU 分解して D の負要素の個数 $c(\alpha)$ をスツルム法によって求める。それが p である. $p=c(\alpha)$.

〔2〕バイセクション法を施行する: 分点 α_ν 毎にスツルムを使用し、すなわち $A-\alpha_\nu I=U^TDU$ を行って、D の負要素の個数 $c(\alpha_\nu)$ を求めることによって、各 $\lambda_1,\cdots,\lambda_p$ を分離する. これで各 λ_k 毎にその下界と上界: $x_{uk}\leqslant\lambda_k<x_{lk}$ が判る。

〔3〕各 k 毎に、$\alpha_k=(x_{uk}+x_{lk})/2$ をシフト点に えらび、$A-\alpha_k I$ を U^TDU 分解した結果を使用して逆反復を行う:

$$\left[\begin{array}{l}(A-\alpha I)\,b^{(l)}=w^{(l-1)};\quad eig=\alpha_k+1/w^{(l-1)T}b^{(l)}\\ w^{(l)}=b/(b^Tb)^{\frac{1}{2}}\ ;\ \text{收束判定};\quad l=l+1\end{array}\right.$$

〔4〕上で求まった eig から改めて $A-eigI$ を作り、それを UTDU 分解した結果を使用しての逆反復によって固有ベクトルを求める. 最後に固有値も再びレーリー商によって更新する. (〔4〕は、粗い精度でよいときには省略される。)

スツルム・逆反復法と呼ばれる計算法の大筋は上述の如くであるが、問題点を整理しよう.

(1) CPU タイムと USE タイムに関し:

上述の方法に関し、従来言われて来たことは、U^TDU 分解のための $\frac{1}{2}m^2n$ 回積和 (ガウスなら $2m^2n$) が一体何回必要か が主たる関心事であって、逆反復法に入って以後の、

反復1回当りの $2mn$ 回積和（ガウスなら $3mn$）は大したことない，ということであった。

CPUタイムについては、大筋はその通りである．ところがページスワップに関しては、U^TDU 分解のためには1回当り $mn/500$ 回（ガウスなら $3mn/500$），逆反復のためには1回当り $2mn/500$ 回（ガウスなら $3mn/500$ あるいは $6mn/500$ 回）*，逆反復回数の方も大いに問題になる．

〔例〕積和 $1\mu s$，ページスワップ1回当り 20 ms の機械で、$m=80$，$n=8000$ のとき、1回当りの CPU タイムとページスワップタイムは．

U^TDU　25 sec : 25 sec　　逆反復　0.3 sec : 50 sec

である．スツルムは、λ_1 のため 10~20 回程度，あと $\lambda_2,\cdots,\lambda_p$ のためにはひとつ当り平均 3~5 回程度が普通である．いっぽう、逆反復の方は、よく分離された固有値に対しては 3~10 回程度であるが、近接固有値に対しては逆反復が 50 回~100 回にも及ぶことが稀ではない．

〔2〕 計算の信頼性・安全性：

特に近接固有値が密集している場合，上述の方法そのままでは収束が甚しく遅いことがある．ときには真値からほど遠い値に‘収束’したかの如くふるまったり、あるいは時間制限にかかって途中で打ちきられることがしばしばである．

この事情のため、汎用の数値計算ライブラリには採用し難いということになっている。

＊ 本稿では、もっぱら、mn 語が、そのプログラムに割り当てられた実メモリ容量をはるかに超える場合について議論する．

2. $Ax=\lambda Mx$ に対するスツルム・逆反復法の改良.

初めに、改良の要点を示そう：

〔1〕 STURM プログラムを、ページスワップ負担が逆反復のためのガウスとくらべて無視できるほど小さくなるよう、巧妙なプログラムを書く. BISECT と INVRAY で使用.

〔2〕 BISECT プログラム：epsbi を与え、その大きさに応じて固有値を分離する.(STURMを使う. 目新しいことはない.)

〔3〕 INVRAY プログラム： 次の三つの部分から成る.

(1) (BISECTによって) 単純根と判定された固有値に対し、再度スツルムを2回行って、シフト率 alfa を決定する. 後続の二段の逆反復過程を、安全かつ効果的に行わせるためのものである.

(2) シフト率を、各分離区間毎に固定した通常の逆反復を行う. 当該区間のそとの固有値に属する固有ベクトル成分を減衰させるためのものである. 単純固有値のあるものについては、この段で固有値，固有ベクトル共に 所望の精度に達することがある.(特に目新しいことはない.)

(3) レーリーシフトつき逆反復：アルゴリズムに目新しいことはないが、使用するガウス消去法プログラムに工夫がある.

〔4〕 全体のコントロール：CPU タイムと USE タイムの何れを重要視するかに応じたコントロールが可能であるようにする.

2.1 STURM と BTSECT

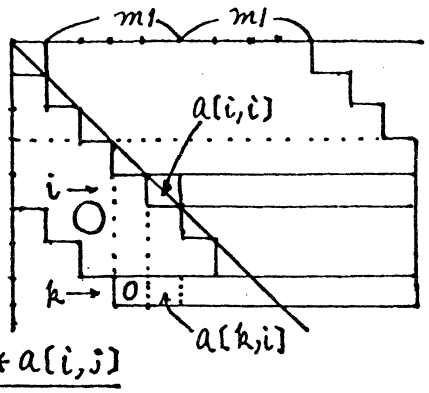
オーソドックスな Martin·Wilkinson のアルゴリズムによることにする[1]。その原理：

> α より小さい $Ax=\lambda Mx$ の固有値の個数は、主座小列式：$\det(A_k-\alpha M_k)$, $k=1,\cdots,n$ の符号変化の回数 $C(\alpha)$ に等しい。

この原理による Martin·Wilkinson のアルゴリズムは、

STURM　　$a[i,j]=a_{i,j}-\alpha * m_{i,j}$　として、

```
p[1] = sign(a[1,1]) ; ip=1 ; count = 0
do k=2,n
    imin = max(1, k-m1) ; jmax = min(k+m1,n)
    do i=imin,k-1
        if abs(a[i,i])<abs(a[k,i]) then
            ip = -ip
            do j=i,jmax
                a[i,j] ⇄ a[k,j]
        if a[i,i] ≠0 then
            t = -a[k,i]/a[i,i]
            do j=i,jmax
                a[k,j] = a[k,j]+t*a[i,j]
    p[k] = ip Π_{i=1}^{k} sign(a[i,i])
    if p[k-1]*p[k]<0 then count=count+1
```



上のスツルムを使って、αl より小さい固有値の個数 p = count を求めたのち、$0<\lambda_1\leqslant\lambda_2\leqslant\cdots\leqslant\lambda_p<\alpha l$ なる λ_k を epsbi の粗さで分離するためのプログラム BISECT は次のようになる：

BISECT

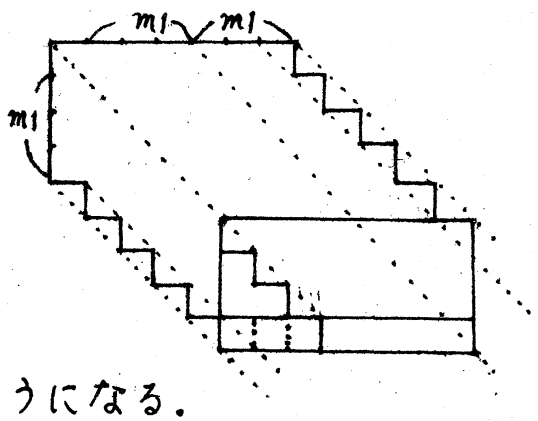
$xu[k]$, $xl[k]$, $nc[k]$, $k=1,\cdots,p$ を用意する.

```
do  k=1,p
    xl[k]=αl ; nc[k]=p ; xu[k]=0.0
k=1
repeat
    repeat
        α=(xu[k]+xl[k])/2 ; Call STURM
        do  i=k,count
            xl[i]=α ; nc[i]=count
        do  i=count+1,p
            xu[i]=max(α,xu[i])
        until k≥nc[k] or xl[k]-xu[k]≤epsbi
    if  k=nc[k] then
        k=k+1
    else
        do  i=k,count
            xu[i]=xu[k]
        k=count+1
    until  k>p
```

STURM のプログラムについて.

スツルムにおいては、欲しいのは count値だけであって、三角分解結果は不要である. そこで、ページスワップを減らすために、約 $2m^2$語のワークエリアを用意して、それをサイクリックに使うことによって目的を達するようにプログラムを書く. また、$A-\alpha M$ を先に作ってしまうのでなく、各k段のガウス過程の直前に k+m1 行の $a_{k+m1,j}+\alpha\cdot m_{k+m1,j}$ を作る.

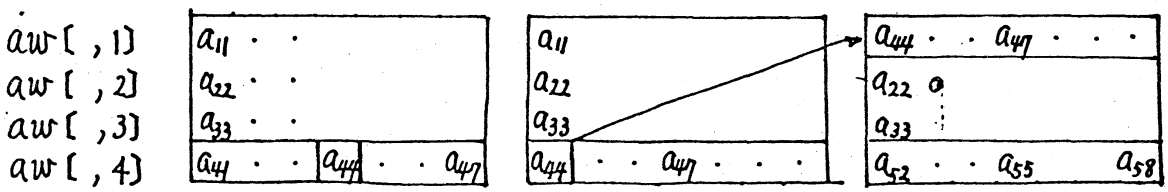
プログラムの解読を助けるために、m1=3 の場合について例示しよう。



消去の度に左シフトという技巧を使う。k=1,2,3までは特別だが簡単で、下図左のようになる。

k=4の段：4番の方程式の係数 $a_{4,j}-\alpha*m_{4,j}$ を aw[,4] に作り、これに対する消去作業を行う。消去の度に左シフトを行う。完了したらそれを aw[,1] に收める。

k=5の段： aw[,2], aw[,3], aw[,1] の順にピボット方程式が使われる。・・・・



(k=4スタート)　(消去完)　(k=5スタート)

k=m1 までのところは簡単だから省略して、k=m1+1 以降のプログラムを示そう。(次頁)

〔ノート〕 差分法や有限要素法によって作られた行列AやMは、帯のふちと、真中三本がノンゼロで、あと帯の中もゼロというようになっているから、これらは、インデックスつきのパックされたアレイに收めておき、消去作業の直前に aw[,m] に $aw[j-k+m,m]=a_{k,j}-\alpha*m_{k,j}$ とやって作る。({1}の部分.)

$m = m1+1$; $mm = 2m1+1$; $eps = 1.0 * 10^{-16}$

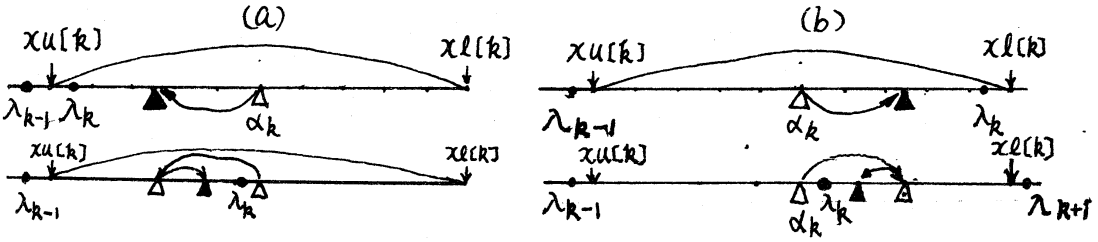
```
do k=m1+1, n
    jmax = min(n, k+m1)
    do j = k-m1, jmax                                   } {1}
        aw[j-k+m, m] = a_{k,j} - alfa * m_{k,j}
    do j = jmax+1, k+m1  aw[j-k+m, m] = 0.0

    kq = (k-1) ÷ m1 ; ist = k - kq*m1
    do iw = ist, m1
        if abs(aw[1, iw]) < abs(aw[1, m]) then
            ip = -ip
            do jw = 1, mm   aw[jw, iw] ⇄ aw[jw, m]
        if aw[1, iw] ≠ 0.0 then
            piv[(kq-1)*m1+iw] = aw[1, iw] ; t = -aw[1,m]/aw[1,iw]
            do jw = 2, mm
                aw[jw-1, m] = aw[jw, m] + t * aw[jw, iw]
            aw[mm, m] = 0.0
        else
            piv[(kq-1)*m1+iw] = eps ; ir = ir+1
            do jw = 2, mm
                aw[jw-1, m] = aw[jw, m]
            aw[mm, m] = 0.0
    do iw = 1, ist-1
        if abs(aw[1,iw]) < abs(aw[1,m] then
            ip = -ip
            do jw = 1, mm   aw[jw, iw] ⇄ aw[jw, m]
        if aw[1, iw] ≠ 0.0 then
            piv[kq*m1+iw] = aw[1, iw] ; t = -aw[1,m]/aw[1, iw]
            do jw = 2, mm
                aw[jw-1, m] = aw[jw, m] + t * aw[jw, iw]
            aw[mm, m] = 0.0
        else
            piv[kq*m1+iw] = eps ; ir = ir+1
            do jw = 2, mm
                aw[jw-1, m] = aw[jw, m]
            aw[mm, m] = 0.0
    piv[k] = aw[1, m]
    do jw = 1, mm
        aw[jw, ist] = aw[jw, m]
    piv[k] = aw[1, k] ; P[k] = Π_{i=1}^{k} sign(piv[i])
    if P[k-1] * P[k] < 0 then count = count+1
```

2.2 INVRAY

前節の BISECT によって xu[k] と xl[k] の間にはさまれた λ_k を、逆反復法によって求めるわけであるが、原理的には、

$$\alpha_k = (xu[k] + xl[k])/2$$

をシフト点にえらんでの逆反復法を行なえば、必ず收束する筈である。ところが、たとえば下図のような状況のとき、收束ははなはだしく遅い。(特にページスワップの負擔が大きい.) そこで、



(1) 逆反復に入る前に、あと二回余計に二分法を行うことによってシフト点を定める。(上例の場合▲に移す。)

ところがこうしても近接根のときは收束がおそいことは相変らずであるから、

(2) 逆反復をある程度やって、まだ收束しないときは、レーリー商シフトつきの逆反復にひきつぐ。

じつは(1)において、一回でなく二回スツルムを行なわねばならぬのは、この(2)のことを安全に行なわせたいがためである。もし、シフト点の両側に殆んど等しい距離のところに λ_k と λ_{k+1} がある、ということだと、レーリ商シフトをやってもうまく λ_k に近づいて呉れないことがある。

INVRAY { 固有ベクトルも同時に求めるもの } [Ax=λx 型用]

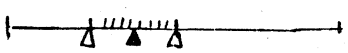
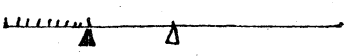
```
lmin , lmax , llmax , eps1 , epse を与える.
v_k (k=1 to p) に初期ベクトルを与える.
k = 1
while k≤p do
    if xu[k]≠xu[k+1] or k=p then                                   {1}
        alfa = (xu[k]+xl[k])/2 ; Call Sturm
        if count = k then                                          {2}
            xl[k] = alfa ; alfa=(xu[k]+alfa)/2 ; Call Sturm
            if count ≠k then                                       {3}
                xu[k] = alfa ; alfa = (alfa+xl[k])/2
            else xl[k] = alfa                                      {3'}
        else                                                       {2'}
            xu[k] = alfa ; alfa = (alfa+xl[k])/2 ; Call Sturm
            if count = k then                                      {4}
                xl[k] = alfa ; alfa = (xu[k]+alfa)/2
            else xu[k] = alfa                                      {4'}
    else
        alfa = xu[k] • ; lmin = 1
100 continue
    epsir = 10^-18 ; Call GLUBR3 (ar, ac, ip, n, m1, alfa, epsir, ir)
    w = v_k
    do j = max(k-8, 1), k-1                                        {5}
        w = w - (v_j^T v_k) v_j
    w = w/(w^T w)^1/2 ; alfav0 = 0.0 ; iconv = 0
    do l = 1 , lmax                                                {6}
        b = w ; Call GSBR (ar, ac, b, ip, n, m1, --)
        alfav = 1.0 / w^T b ; eig = alfa + alfav
        if |alfav - alfav0| < eps1 then iconv = 1
        alfav0 = alfav ; w = b
        do j = max(k-8, 1), k-1
            w = w - (v_j^T b) v_j                                  {7}
        w = w / (w^T w)^1/2
        if l > lmin and iconv = 1 then go to 200
200 continue
```

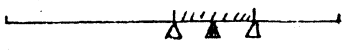
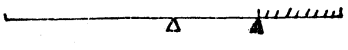
① ②

①　②

```
difc = ‖Aw − eig * w‖2
if dif < epse then ll = 0 ; go to 400

irregu = 0
do ll = 1, llmax
    b = w
    Call GUSBR3(ar, b, n, m1, eig, epsir, ir)
    beta = wᵀb ; alfav = 1.0/beta ; eig = eig + alfav
    w = b ; jmin = max(k-8, 1)                          {8}
    do j = jmin, k-1  w = w − (vjᵀb) vj                 {9}
    w = w/(wᵀw)^(1/2)
    if |alfav| < epse then go to 300                    {10}
300 continue
    difc = ‖Aw − eig * w‖2                              {11}
400 continue
    v_k = w
    if eig < xu[k] or eig ≥ xl[k] then irregu = 1
    write k, l, ll, difc, eig, irreg {その他 f, dif1など}
    if eig ≥ xl[k] * (1 + 10^-15) then                  {13}
        alfa = (alfa + xl[k])/2 ; k = k+1 ; go to 100
    else k = k+1
```

〔解説〕

{3}：  　{3'} 

{4}：  　{4'} 

{5}：最近求まった8個の固有ベクトルは、Wからぬきとられる。

{7}と{9}：両直交化である。直前から8個前までの固有ベクトルの成分をぬきとった。

次頁にGUSBR3を示す。これをLU分解部と右辺に対する前進・後退部とに分けたものがGLUBR3とGSBである。

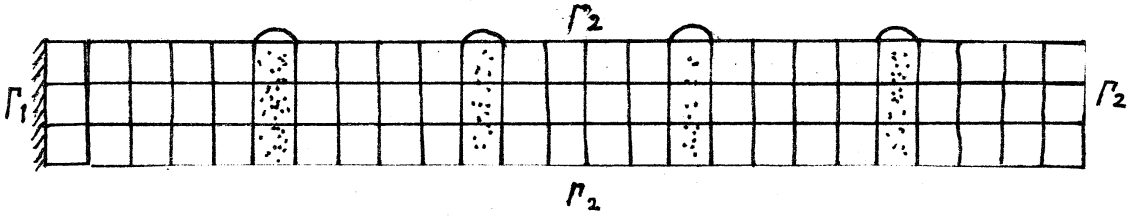
GUSBR3 (ar, b, n, m1, eig, epsir, ir)

```
m = m1+1 ; ir = 0
do i = 1, m1
    do j = 1, i+m1
        ar[j,i] = a_{i,j} - eig * δ_{i,j}
    do j = i+m1+1, m1+m1+1
        ar[j,i] = 0.0

do k = 1, n
    imax = min(k+m1, n) ; jmax = min(k+m1+m1, n)
    if k+m1 ≤ n then
        do j = k ··· , jmax
            ar[1+j-k, k+m1] = a_{k+m1,j} - eig * δ_{k+m1,j}
    amax = abs(ar[1,k]) ; ipk = k
    do i = k+1, imax
        aik = abs(ar[1,i])
        if aik > amax then
            amax = aik ; ipk = i
    jkmax = min(1+m1+m1, n-k+1)
    if k ≠ ipk then
        b[k] ⇄ b[ipk]
        do jk = 1, jkmax
            ar[jk,k] ⇄ ar[jk,ipk]
    if amax < epsir then
        ar[1,k] = epsir * sign(ar[1,k]) ; ir = ir+1
    do i = k+1, imax
        t = - ar[1,i] / ar[1,k]
        b[i] = b[i] + t * b[k]
        do jk = 2, jkmax
            ar[jk-1, i] = ar[jk,i] + t * ar[jk,k]
        ar[jkmax, i] = 0.0

do k = n downto 1
    jmax = min(k+m1+m1, n)
    sum = 0.0
    do j = k+1, jmax
        sum = sum + ar[1+j-k,k] * b[j]
    b[k] = (b[k] - sum) / ar[1,k]
```

2.3 結果に関する中間報告

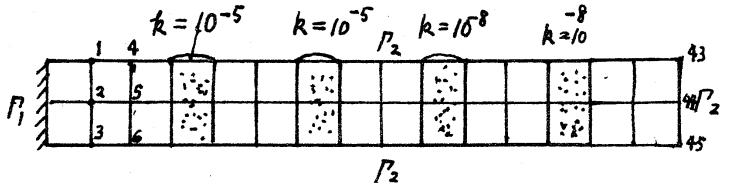
固有値計算プログラムのデバッグと、評価を行うに際して決定的に重要なのは、テスト問題の選択である。いろいろのレベルの近接度をもつた固有値系を、システマティックに作るために、有限要素法を利用した。すなわち、下図のような場を作り、▨部のk値を任意に変えれば、思いのまゝの近接度をもった行列が生成される。(□部の $k=1.0$ にとる。)

支配方程式 : $\nabla\cdot(-k\nabla u)=\lambda u \rightarrow Au=\lambda Mu$ (但し$M=I$とした。)



Γ_1 : $u=0$ (固定), Γ_2 : $\nabla u=0$ (自由)

〔簡単な問題例〕



節点番号を上のようにつけたから、これは、$m=4$ ($m1=3$)の帯行列となる、$n=45$。熱伝導の問題だと思えば、▨のところが、熱伝導率の非常に小さい物質となつている。

次頁上段に行列 A を示す。左端の列にならんでいるのが対角項である。対称だから上半分だけ示している。次頁下段は、$epsbi=10^{-4}$ によって BISECT を行なつて、$epse=10^{-10}$, $epse=10^{-8}$ で LL反復を行なつたときのもの。$eps1=10^{-4}$とした。

2.0000000000000000	-1.0000000000000000	0.0	-0.5000000000000000
4.0000000000000000	-1.0000000000000000	0.0	-1.0000000000000000
2.0000000000000000	0.0	0.0	-0.5000000000000000
2.0000000000000000	-1.0000000000000000	0.0	-0.5000000000000000
4.0000000000000000	-1.0000000000000000	0.0	-1.0000000000000000
2.0000000000000000	0.0	0.0	-0.5000000000000000
1.0000100000000000	-0.5000050000000000	0.0	-0.0000050000000000
2.0000200000000000	-0.5000050000000000	0.0	-0.0000100000000000
1.0000100000000000	0.0	0.0	-0.0000050000000000
1.0000100000000000	-0.5000050000000000	0.0	-0.5000000000000000
2.0000200000000000	-0.5000050000000000	0.0	-1.0000000000000000
1.0000100000000000	0.0	0.0	-0.5000000000000000
2.0000000000000000	-1.0000000000000000	0.0	-0.5000000000000000
4.0000000000000000	-1.0000000000000000	0.0	-1.0000000000000000
2.0000000000000000	0.0	0.0	-0.5000000000000000
1.0000100000000000	-0.5000050000000000	0.0	-0.0000050000000000
2.0000200000000000	-0.5000050000000000	0.0	-0.0000100000000000
1.0000100000000000	0.0	0.0	-0.0000050000000000
1.0000100000000000	-0.5000050000000000	0.0	-0.5000000000000000
2.0000200000000000	-0.5000050000000000	0.0	-1.0000000000000000
1.0000100000000000	0.0	0.0	-0.5000000000000000
2.0000000000000000	-1.0000000000000000	0.0	-0.5000000000000000
4.0000000000000000	-1.0000000000000000	0.0	-1.0000000000000000
2.0000000000000000	0.0	0.0	-0.5000000000000000
1.0000000010000000	-0.5000000005000000	0.0	-0.0000000005000000
2.0000000020000000	-0.5000000005000000	0.0	-0.0000000010000000
1.0000000010000000	0.0	0.0	-0.0000000005000000
1.0000000010000000	-0.5000000005000000	0.0	-0.5000000000000000
2.0000000020000000	-0.5000000005000000	0.0	-1.0000000000000000
1.0000000010000000	0.0	0.0	-0.5000000000000000
2.0000000000000000	-1.0000000000000000	0.0	-0.5000000000000000
4.0000000000000000	-1.0000000000000000	0.0	-1.0000000000000000
2.0000000000000000	0.0	0.0	-0.5000000000000000
1.0000000010000000	-0.5000000005000000	0.0	-0.0000000005000000
2.0000000020000000	-0.5000000005000000	0.0	-0.0000000010000000
1.0000000010000000	0.0	0.0	-0.0000000005000000
1.0000000010000000	-0.5000000005000000	0.0	-0.5000000000000000
2.0000000020000000	-0.5000000005000000	0.0	-1.0000000000000000
1.0000000010000000	0.0	0.0	-0.5000000000000000
2.0000000000000000	-1.0000000000000000	0.0	-0.5000000000000000
4.0000000000000000	-1.0000000000000000	0.0	-1.0000000000000000
2.0000000000000000	0.0	0.0	-0.5000000000000000
1.0000000000000000	-0.5000000000000000	0.0	
2.0000000000000000	-0.5000000000000000		
1.0000000000000000			

STRUM KAISU= 60

K	L	LL	F **	DIFC	EIG	LL	F	DIFC	EIG
1	1	1	2	0.235D-12	0.000000000847585	0	2	0.108D-08	0.000000000854363
2	1	2	2	0.235D-12	0.000000005814623	1	2	0.179D-09	0.000000005814563
3	1	2	2	0.674D-14	0.000000850397529	2	2	0.845D-09	0.000000850397528
4	1	2	2	0.412D-15	0.000005818384915	2	2	0.200D-09	0.000005818384915
5	3	1		0.608D-15	0.130988706559026	1		0.588D-09	0.130988706559026
* 6	1	4	2	0.537D-15	0.633974604384458	2	2	0.105D-08	0.633974604251678
7	1	4	2	0.418D-12	0.633977726206820	6	2	0.660D-10	0.633977726206820
8	1	4	2	0.403D-12	0.633975935359265	5	2	0.856D-10	0.633982769889012
9	1	6	2	0.225D-11	0.633974597550951	4	2	0.504D-09	0.633975935359265
10	1	4	2	0.133D-13	0.633982769889012	2	2	0.788D-09	0.633974597626185
11	1	3	2	0.511D-11	0.633974599342076	2	2	0.642D-09	0.633974599341773
* 12	1	3	2	0.459D-11	0.633974604918049	3	2	0.446D-11	0.633983300138170
13	1	2	2	0.638D-15	0.633983300138170	2	2	0.438D-10	0.633974604918049
14	4	2		0.233D-15	0.759410727188718	2		0.233D-15	0.759410727188718
15	3	2		0.191D-15	0.981256155790828	2		0.191D-15	0.981256155790828
16	1	4	2	0.752D-13	1.000000003964257	1	2	0.324D-08	1.000000006083833
17	1	1	2	0.762D-13	1.000000011034313	1	2	0.492D-08	1.000000005209902
18	1	5	2	0.138D-13	1.000003968777679	4	2	0.665D-10	1.000003968777680
19	1	2	2	0.163D-13	1.000011036336149	2	2	0.664D-10	1.000011036336149
20	3	2		0.314D-15	1.655555485733648	2		0.320D-15	1.655555485733648
21	1	6	2	0.265D-14	1.697224367542230	4	2	0.133D-08	1.697224367798118
22	1	3	2	0.227D-14	1.697229641804310	3	2	0.414D-09	1.697229641804278
23	1	4	2	0.345D-14	1.697224374668973	3	2	0.141D-08	1.697224374668711
24	2	1		0.636D-15	1.697236764061253	0		0.314D-09	1.697236764061253
25	3	2		0.288D-15	1.933646468546035	2		0.410D-10	1.933646468546035
	37	76				58			

epsbi=10^{-4} , epse=10^{-10} , epse=10^{-8} ,

* K=6 とK=12 のものが 10^{-10} のオーダーの近接度となっていて、いちばん近い。

** F=2 とあるのは BISECTにて'重複'扱いされたことを示している。

STRUM KAISU= 88

K	L	LL	F	DIFC	EIG *
1	1	1	2	0.235D-12	0.000000000847585
2	1	2	2	0.235D-12	0.000000005814623
3	1	2	2	0.67-D-14	0.000000850397529
4	2	0		0.525D-11	0.000005818384915
5	3	2		0.517D-11	0.130988706559026
6	1	3	2	0.655D-12	0.633974604384458
7	1	3	2	0.433D-12	0.633974599342072
8	1	5	2	0.337D-12	0.633974597550948
9	1	4	2	0.116D-12	0.633975935359265
10	1	2	2	0.468D-12	0.633974604918049
11	2	1		0.255D-12	0.633977726206820
12	2	0		0.307D-10	0.633982769889014
13	2	0		0.307D-10	0.633983300138170
14	4	2		0.581D-15	0.759410727188718
15	3	2		0.427D-14	0.981256155790828
16	1	4	2	0.752D-13	1.000000003964257
17	1	1	2	0.792D-13	1.000000011034313
18	2	2		0.148D-13	1.000003968777679
19	2	0		0.525D-11	1.000011036336149
20	3	2		0.442D-12	1.655555485733648
21	1	4	2	0.655D-12	1.697224367542230
22	1	3	2	0.133D-14	1.697224374668973
23	2	1		0.125D-11	1.697229641804310
24	2	1		0.155D-11	1.697236764061253
25	3	2		0.117D-11	1.933646468546035
	44	49			

LL	F	DIFC	EIG
0	2	0.108D-08	0.000000000854363
1	2	0.179D-09	0.000000005814563
2	2	0.845D-09	0.000000850397528
0		0.200D-09	0.000005818384915
2		0.588D-09	0.130988706559026
2	2	0.224D-09	0.633974604378081
2	2	0.526D-09	0.633974599231349
7	2	0.356D-08	0.633974602328289
4	2	0.642D-09	0.633975935359265
2	2	0.354D-08	0.633974599670639
1		0.142D-10	0.633977726206820
0		0.314D-10	0.633982769889014
0		0.307D-10	0.633983300138170
2		0.580D-15	0.759410727188718
2		0.427D-14	0.981256155790828
1	2	0.324D-08	1.000000006083633
1	2	0.492D-08	1.000000005209902
1		0.494D-11	1.000003968777679
0		0.681D-11	1.000011036336149
2		0.442D-12	1.655555485733648
2	2	0.123D-08	1.697224367760609
2	2	0.125D-08	1.697224374668917
0		0.374D-08	1.697229641806272
0		0.440D-08	1.697236764060845
2		0.312D-09	1.933646468546035
38			

epsbi = 10^{-6} , epse = 10^{-10} , epse = 10^{-8}

以上は、epse = 10^{-10} と 10^{-8} のときのものであった。次に、epse = 10^{-12} としたものを示そう。これでみると、DIFC, すなわち $\|Aw - eig*w\|_2$ がすべて 10^{-15} 以下に收まっている。

K	L	LL	F	DIFC	EIG *	DIF1
1	1	2	2	0.310D-15	0.000000000847585	0.235D-12
2	1	2	2	0.445D-15	0.000000005814623	0.202D-10
3	1	2	2	0.335D-15	0.000000850397528	0.205D-09
4	2	1		0.342D-15	0.000005818384915	0.525D-11
5	3	2		0.233D-15	0.130988706559026	0.162D-10
6	1	4	2	0.204D-15	0.633974604384458	0.495D-12
7	1	4	2	0.170D-15	0.633974599342072	0.916D-12
8	1	6	2	0.235D-15	0.633974597550948	0.129D-12
9	1	4	2	0.204D-15	0.633975935359265	0.170D-10
10	1	2	2	0.325D-15	0.633974604918049	0.170D-08
11	2	2		0.350D-15	0.633977726206820	0.193D-12
12	2	1		0.248D-15	0.633982769889012	0.307D-10
13	2	0		0.728D-14	0.633983300138170	0.728D-14
14	4	2		0.578D-15	0.759410727188718	0.385D-09
15	3	2		0.425D-14	0.981256155790828	0.217D-08
16	1	5	2	0.316D-15	1.000000003964257	0.752D-13
17	1	2	2	0.614D-15	1.000000011034313	0.248D-13
18	2	2		0.333D-15	1.000003968777679	0.123D-11
19	2	1		0.694D-15	1.000011036336149	0.525D-11
20	3	2		0.226D-15	1.655555485733648	0.103D-08
21	1	4	2	0.162D-14	1.697224367542230	0.491D-10
22	1	3	2	0.133D-14	1.697224374668973	0.503D-11
23	2	2		0.265D-15	1.697229641804310	0.106D-14
24	2	1		0.428D-15	1.697236764061253	0.232D-08
25	3	2		0.293D-15	1.933646468546035	0.119D-08
	44	57				

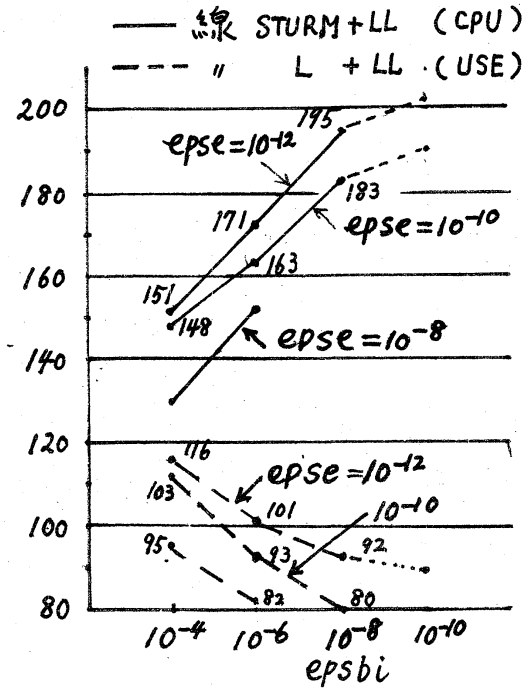
【参考】
右端のDIF1は、LLの最後のひとつ前のときにおけるDIFCの値を示している。
DIF1とDIFCを比べて見ると、最後の反復でどれぐらい改善されたかが判る。

epsbi = 10^{-6} epse = 10^{-12}

* EIGは、epse = 10^{-10} でのものですでに多くがすべて 10^{-15} の位まで求まっている。

epsbi と epse を動かすとき、CPU タイムと USE タイムが傾向としてどのように変化するかをしらべてみる。CPU タイムにとつては、STURM 回数と LU 分解数が利く。USE タイムにとつては、LU 分解数のほかに L 反復の回数が利く。下表にそのとりまとめた結果を示す。(L 反復の前の LU 分解はいつでも、必ずやるから表にはあげない。)

epsbi	epse	(BISECT) STURM	(INVRAY) STURM	L	LL	STURM+LL	L+LL
10^{-4}	10^{-12} 10^{-10} 10^{-8}	60	12	37	79 76	151 148 130	116 113 95
10^{-6}	10^{-12} 10^{-10} 10^{-8}	88	26	44	57 49 38	171 163 152	101 93 82
10^{-8}	10^{-12} 10^{-10} 10^{-8}	115	38	50	42 30	195 183	92 80



上の表では印象が不鮮明ゆえ、ぐらふにしてみたのが右のものである。大体，思惑通りの結果ではあるが、epse を甘くしても、そう大して計算時間がみぢかくならぬのは意外に近い印象を受けた。

改善の余地はある。

94

〔ひとつの改善案〕

初め粗い epsbi (たとえば 10^{-4}) で BISECT を行なうと、その段階で各固有値が密集したあるグループに属しているか、それとも孤立しているかが判る。(たとえば先程の例で epsbi = 10^{-4} を使用すれば、$\lambda_1 \sim \lambda_4$ が第1密集グループ, λ_5 が孤立根, $\lambda_6 \sim \lambda_{13}$ が第2密集グループ … という具合である。)

そこで、孤立根に対してはほぼ先程の通りの処理を行なうが、密集グループに属するものに対しては、各グループ毎にシフト点を えらんで、シュミット直交化つきの古典的な同時逆反復法をほどこす。

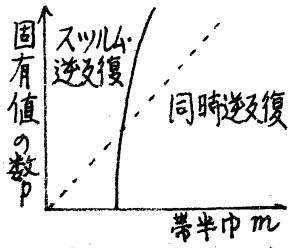
この同時逆反復法で、各固有値への収束は比較的速い。固有ベクトルの方は、各固有値に対応させて計算機の限度に近いほどの精度で求めようとするとやはり収束に手間どる。従って、適当なところで各固有値毎にシフト点を移しての逆反復にひきつぐがよい。

しかし、固有ベクトルについては、それらが張るところの部分空間が、問題の密集固有値に対応する固有空間に収束すればそれで十分だという考えならば、その必要はない。

同時逆反復法は、ベクトルの収容法をうまく考え、プログラムに若干の工夫をこらすことによって、ページスワップを大巾に減らすことができる。

3 同時逆反復法

CPUタイムについては、スツルム・逆反復法と、同時逆反復法の適用範囲の傾向は、およそ右図のようになる。ページスワップについてはどうであろうか、ここではBatheのSubspace法をとり上げる。常識的なアルゴリズムはおよそ下記の如くなろう:




```
初期 X = {x_1, ..., x_q} を用意, q = min(p+8, 2p)
A を L^T D U 分解 ; k = 0
while k ≤ kmax and ||A x_p - λ̃_p M x_p|| / ||λ̃_p M x_p|| > eps do
    do i = 1, q                    { Y = MX を作る }
        x_i = x_i / sqrt(x_i^T x_i) ; y_i = M x_i
    do i = 1, q                    { AX̄ = Y を解く }                    {*}
        z = L^{-1} y_i ; x̄_i = (D L^T)^{-1} z
    do i = 1, q                    { q×q 行列 Ã = X̄^T A X̄, M̃ = X̄^T M X̄ を作る }
        z = M x̄_i ; m̃_i = X̄^T z ; ã_i = X̄^T y_i
    solve eigenprob: Ã Q̃ = M̃ Q̃ Λ̃
    do i = 1, q                    { もとの空間のベクトルにひきのばす }
        x_i = X̄ q̃_i
    k = k+1
```

研究集会当時にはこれでもって評価し、スツルム逆反復法とくらべた。そして、'ページスワップに関しては $q > m$ となると全くサブスペースは不利である。' と結んだ。実は、上の通りのアルゴリズムだとそうなるが、サブスペースにおいては、右辺のベクトル Y をブロック化して收め、ブロック毎に前進・後退代入をやればページスワップを大巾に減らす

ことができることが判った。SAP Ⅳ の Subspace プログラムを調べてみると、これは仮想メモリ方式向けでないが、AとYと共によこ方向にブロックわけして処理することによって、ディスクと主メモリの転送回数を減らす工夫がすでにしてあった。我々の場合，仮想メモリ方式の利点を利用できるから、Aの方は普通の通りとして、（すなわちDLは普通の通りとして、Y（したがってXも）だけをよこ方向にブロック化して收容してのアルゴリズムを採用すればよい。

—以上，前言を訂正し、おわびをしたい。—

さて、問題のサブスペースに関し、極く最近原著者 Bathe 自身による注目すべき改良が発表された[3]。

4. 対称帯 $Ax=\lambda x$ に対する直接QR法

問題をこの型に限るならば、Aを特殊なハウスホルダ変換の列によって帯巾をもとの二倍におさえて三重対角化する方法が、筆者らによって考案，開発された。[4] この方法は汎用的ではあるが、帯巾が大きくなると、やはりページスワップネックとなる．今までやった問題の上限は、帯半巾 80，元数 8000 である。それにしても、所望の固有値の数pが大きくなるとこの方法によらざるを得ないが、pが比較的小さい問題には、帯QR法の方が有望となる。そういう観点から、帯QR法をとり上げることにした。

〔原理の復習〕 $A = A_1$ とする．単位直交変換 Q_ℓ^T により、A_ℓ を三角化： $Q_\ell^T A_\ell = R_\ell$ 即ち $A_\ell = Q_\ell R_\ell$, $Q_\ell^T Q_\ell = I$ する．ここに $Q_\ell^T = (I - \alpha_\ell^{(n-1)} w_\ell^{(n-1)} w_\ell^{(n-1)T}) \cdots (I - \alpha_\ell^{(1)} w_\ell^{(1)} w_\ell^{(1)T})$.

Q_ℓ R_ℓ を逆順に掛けて $A_{\ell+1} := R_\ell Q_\ell (= Q_\ell^T A_\ell Q_\ell)$ とすると $\ell \to \infty$ のとき $A_{\ell+1}$ は $\lambda_n \gg \cdots \gg \lambda_1 > 0$ を対角要素とする対角行列に收束する．内容的には、$P_\ell := Q_1 \cdots Q_\ell$ の各列ベクトルは、單位行列 I の各列ベクトルを初期ベクトルとしたシュミット直交化つき同時逆反復法によって得られたベクトル $\bar{x}_\ell$ に等しい。すなわち同時逆反復法の一種と見なされる。(2)

我々のアルゴリズムは、帶QR法がシュミット直交化つきの同時逆反復法だという見方に強く影きようされているので、以下（本質的には(2)に從って）その証明の大筋を見ておく．

$$A_{\ell+1} := P_\ell^T A P_\ell \ , \text{但し} \ P_\ell := Q_1 \cdots Q_\ell \ , \ S_\ell := R_\ell \cdots R_1 \qquad \cdots(1)$$

とおくと、 $P_\ell S_\ell = Q_1 \cdots Q_{\ell-1} Q_\ell R_\ell R_{\ell-1} \cdots R_1 = P_{\ell-1} A_\ell S_{\ell-1}$ $\cdots(2)$

(1) より、 $A_\ell = P_{\ell-1}^T A P_{\ell-1}$, 從って $P_{\ell-1} A_\ell = A P_{\ell-1}$, これを(2)に入れて、

$$P_\ell S_\ell = A P_{\ell-1} S_{\ell-1} = A^2 P_{\ell-2} S_{\ell-2} = \cdots = A^\ell \qquad \cdots(3)$$

同時逆反復法の方を次n列ベクトルの方から順に直交化して

$$A X_\ell = \bar{X}_{\ell-1} \ , \ \bar{X}_\ell = L_\ell X_\ell \quad \text{即ち} \quad A \bar{X}_\ell L_\ell^{-1} = \bar{X}_{\ell-1} \qquad \cdots(4)$$

と書くと、 $\bar{X}_\ell = A^{-1} \bar{X}_{\ell-1} L_\ell = A^{-2} \bar{X}_{\ell-2} L_{\ell-1} L_\ell = \cdots = A^{-\ell} \bar{X}_0 L_1 \cdots L_\ell$

より、 $\bar{X}_\ell = A^{-\ell} \tilde{L}_\ell$, $A^\ell = \tilde{L}_\ell \bar{X}_\ell^T$. これを転置して、 $\tilde{L}_\ell$ とおく.

$$A^\ell = \bar{X}_\ell \tilde{L}_\ell^T \qquad \cdots(5)$$

(3),(5) より $A^{2\ell} = S_\ell^T S_\ell$, $A^{2\ell} = \tilde{L}_\ell \tilde{L}_\ell^T$ を得るゆえコレスキー分解の一意性によって、$S_\ell = \tilde{L}_\ell^T$ が從がう．一方，(3),(5)から $P_\ell = A^\ell S_\ell^{-1}$, $\bar{X}_\ell = A^\ell (\tilde{L}^T)^{-1}$ ゆえ、$P_\ell = \bar{X}_\ell$ が從がう．（終）

シュミット直交化つき同時逆反復法だということから、重複あるいは近接固有値があってもうまく行く筈である。また原理通りやったのでは近接固有値があるとき収束ははなはだ遅い。そこで、原点シフトを必ずやることに昔からなっている。

〔プログラム作成上の要点をおさえるための考察〕

$$\lambda_1 \leqslant \lambda_2 \leqslant \lambda_3 \leqslant \cdots \leqslant \lambda_n$$

$$\bar{x}_1^{(0)} = c_1 u_1 + c_2 u_2 + c_3 u_3 + \cdots + c_n u_n \quad , c_1 \neq 0 \text{（初期ベクトル）}$$

と書く。（QR法ではこれらはimplicitである。同時逆反復とみての話）

$\lambda_1 < \lambda_2$ のときは、逆反復法の原理からして、

$$\bar{x}_1^{(l)} = \left(\frac{1}{\lambda_1}\right)^l \left\{ c_1 u_1 + \left(\frac{\lambda_1}{\lambda_2}\right)^l c_2 u_2 + \left(\frac{\lambda_1}{\lambda_3}\right)^l c_3 u_3 + \cdots \right\} / k \quad \cdots\cdots$$

$$= \alpha_1 u_1 + \varepsilon_{12} u_2 + \cdots \quad \text{と書く.}$$

$\lambda_1 = \lambda_2 < \lambda_3$ なら、

$$\bar{x}_1^{(l)} = \left(\frac{1}{\lambda_1}\right)^l \left\{ c_1 u_1 + c_2 u_2 + \left(\frac{\lambda_1}{\lambda_3}\right)^l c_3 u_3 + \cdots \right\} / k$$

{ k は正規化のための因子 }

$A_{l+1} = P_l^T A P_l = \bar{X}_l^T A \bar{X}_l$, $\bar{X}_l = [\bar{x}_n^{(l)}, \cdots, \bar{x}_1^{(l)}]$, $\bar{x}_3^{(0)} =$ （例えば） $\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

となっているから、

A_{l+1} の右下隅に近いところの要素：

(3,3) ・ ・
・ (2,2) (2,1)
・ (1,2) (1,1)

をしらべてみる。それらは $\bar{x}_i^T A \bar{x}_j$ で、

$$\bar{x}_i^T A \bar{x}_i = (\alpha_i)^2 \lambda_i + \sum_{k \neq i} (\varepsilon_{ik})^2 \lambda_k \qquad \text{（対角項）}$$

$$\bar{x}_i^T A \bar{x}_j = \alpha_i \varepsilon_{ji} \lambda_i + \varepsilon_{ij} \alpha_j \lambda_j + \sum_{k \neq i,j} \varepsilon_{ik} \varepsilon_{jk} \lambda_k \qquad \text{（非対角項）}$$

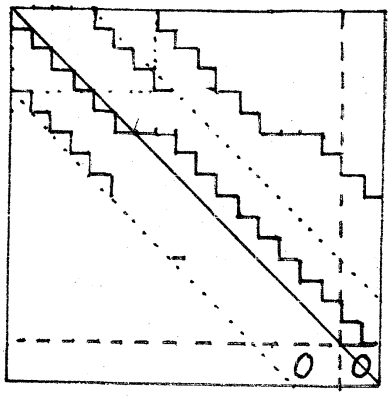
いま、$\lambda_1 < \lambda_2$ だが $\lambda_1 \fallingdotseq \lambda_2$（近接）だとしてみると、右下隅の (1,1) 要素 $\bar{x}_1^T A \bar{x}_1 = (\alpha_1)^2 \lambda_1 + (\varepsilon_{12})^2 \lambda_2 + \cdots$ の収束は順調であるが、(1,2)=(2,1) 要素 $= \bar{x}_1^T A \bar{x}_2 = \alpha_1 \varepsilon_{21} \lambda_1 + \varepsilon_{12} \alpha_2 \lambda_2 + \cdots$ は、ε_{12}（$\bar{x}_1^{(l)}$ の u_2 成分）のため収束がおそい。

プログラム作成上の要点としては、重複，あるいは極度に近接した固有値が密集しているとき、如何にうまく処理するか、ということがその第一である。

〔アイデアの解説〕

$0<\lambda_1\leqslant\lambda_2\leqslant\cdots$ とし、$\bar{A}=A-\lambda_1 I$ とおく．行列 $\bar{A}$ の階数不足値 ir を tol の粗さ（精しさ）に応じて決定できる特別のハウスホルダ三角分解アルゴリズム[5]を応用した新しい Q^TAQ 変換のプログラム： QTAQBR を使用する。

λ_1 が S 重の固有値なら、$\bar{A}$ は S 重のゼロ固有値を持ち、$\bar{A}$ の階数不足値 ir がちょうど S となる。（S=2 の場合）→



例えば S=2 の場合，三角分解を

$$Q^T\bar{A}=R,\quad \bar{A}_2=Q^T\bar{A}Q$$

を作ると、$\bar{A}_2$ は元の帯のワク内に収まり、かつ下二行はゼロとなる。（今は対称ゆえ右二列もゼロ）これは2重のゼロ固有値をもっていることは明らかである。

このときは $\bar{A}_2=Q^T\bar{A}Q$ の下二行，右二列を一挙に落して先に進めばよい。従って、重複固有値があれば、それだけ仕事がはかどる，という好ましい結果になる。λ_1，λ_2 を重複とみるかどうかは、tol をどう与えるかによるゆえ、これを特に粗くとれば、仕事を早く済ますことができる。

EIGQR　{ tol (精), tolc (粗) を与える. (例 $tol=10^{-10}$, $tolc=10^{-4}$) }
alfa : 所望の固有値 $\lambda_1 \cdots \lambda_p$ の上界とする。(あるいは anorm = ‖A‖.)

```
k = 1 ; nv = n ; eig[0] = 0.0 ; eigv = 0.0
eps = 10^-16 * alfa
行列 A(n,n) に対し QTAQBR を行う
if iy > 0 then      {ゼロ固有値を始末する}
    do i = 1, iy
        eig[i] = eig[0]
    nv = n - iy ; k = k + iy
while k ≤ p do
    repeat          {シフト実 固定の QTAQ 粗反復}
        diag = a[nv,nv] ; eps = 10^-16 * alfa
        行列 A(nv,nv) に対し QTAQBR を行う.        {*}
        until abs(a[nv,nv] - diag) < tolc * alfa
    eigv = a[nv,nv] ; eig[k] = eig[k-1] + eigv
    eps = max(10^-16 * alfa, tol * abs(eigv))
    repeat          {毎回シフトを行ないながらの QTAQ 精反復}
        do i = 1, nv                {シフト}
            a[i,i] = a[i,i] - eigv
        行列 A(nv,nv) に対し QTAQBR を行う
        eigv = a[nv,nv] ; eig[k] = eig[k] + eigv
        until iy > 0    {単根なら iy = 1}
    do i = 1, iy-1      {iy = 1 ならやらぬ}
        eig[k+i] = eig[k]
    nv = nv - iy ; k = k + iy
```

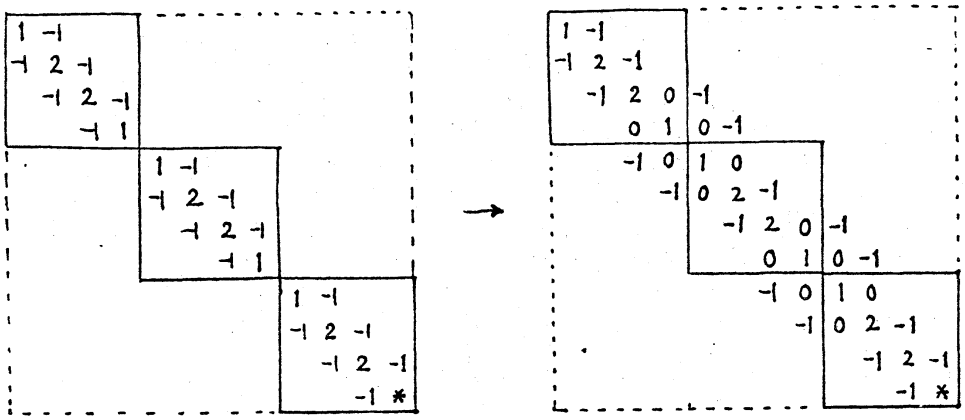
{*} の QTAQBR は、これから求めようとしている λ_k (およびそれと近接して λ_{k+1} がもしあれば λ_{k+1} も) に対応する固有ベクトル成分 を、他とくらべて卓越させるためのものである。

```
QTAQBR ( a[n,nv], eps, ir, irov, l, m1

ir = 0 ; m = m1+1 ; irmax = l-m ; irov = 0
do  kk = 1, nv
    k = kk-ir ; imax = min0(kk+m1,nv)
    sum = Σ_{i=k}^{imax} a[i,kk]**2
    sk = sqrt(sum) ; jmax = min0(kk+2*m1,nv)
    if sk > eps then
        hk = sk*(sk+abs(a[k,kk])) ; alf[k] = 1.0/hk
        if a[k,kk] < 0.0 then sk = -sk
        wk[1,k] = a[k,kk]+sk ; a[k,kk] = -sk
        do  i = k+1 imax
            wk[i-k+1,k] = a[i,kk] ; a[i,kk] = 0.0
        pT = α wTA  を行う
        A = A - w pT を行う
    else
        ir = ir+1
        do  i = k, imax  a[i,kk] = 0.0
        if (ir > irmax)∧(k+l-1 ≤ n) then
            irov = 1 ; return
    irk[k] = ir
⊙   if kr+m1+irk[kr] < kk then
        R(I-α w wT) を行う { p = α R w , A = R-p wT }
        kr = kr+1
○ while kr ≤ nv-ir do
        R(I-α w wT) を行う { p = α R w , A = R-p wT }
        kr = kr+1
  return
```

RQ 変換を⊙と○とに分けることによってページスワップ

を大中に減らすことができる。なほこのプログラムは'原理'を確認するために作ったもので、実際には、帯のワク内でうまくやるための可なり手のこんだプログラムを作る.(未プログラム)

[テスト例]



$* = 1+e$, $e = 10^{-12}, 10^{-10}, 10^{-8}, 10^{-6}$

$tol = 10^{-8}$

k	$e = 10^{-12}$	10^{-10}	10^{-8}	10^{-6}
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.000000000000	0.000000000025	0.000000002500	0.000000249999
4	0.585786437627	0.585786437669	0.585786441894	0.585786864403
5	0.585786437627	0.585786437669	0.585786441894	2.000000250000
6	0.585786437627	0.585786437669	0.585786441894	3.414213635594
7	2.000000000000	2.000000000025	2.000000002500	0.585786437626
8	2.000000000000	2.000000000025	2.000000002500	0.585786437626
9	2.000000000000	2.000000000025	2.000000002500	1.999999999999
10	3.414213562373	3.414213562380	3.414213563105	1.999999999999
11	3.414213562373	3.414213562380	3.414213563105	3.414213562373
12	3.414213562373	3.414213562380	3.414213563105	3.414213562373

反復回数→ 粗7, 精4　　7, 4　　7, 4　　10, 11

あと (+1) (+1) で12ケタまで正しく出る　　〃 (+1) (+1)

$tol = 10^{-12}$

k	$e = 10^{-12}$	10^{-10}	10^{-8}	10^{-6}
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.000000000000	0.000000000025	0.000000002500	0.000000249999
4	0.585786437627	0.585786437669	0.585786441894	0.585786864403
5	0.585786437627	0.585786437626	0.585786437626	2.000000250000
6	0.585786437627	0.585786437626	0.585786437626	3.414213635596
7	2.000000000000	2.000000000024	2.000000002499	0.585786437626
8	2.000000000000	2.000000000000	3.414213563105	0.585786437626
9	2.000000000000	2.000000000000	2.000000000000	3.414213562373
10	3.414213562373	3.414213562380	2.000000000000	3.414213562373
11	3.414213562373	3.414213562373	3.414213562373	2.000000000000
12	3.414213562373	3.414213562373	3.414213562373	2.000000000000

粗7, 精4　　18, 12　　14, 11　　10, 13

13頁に示した例題を従来の帯QR法でやると、固有値の求まる順が、はなはだしく乱れる。(左端の場からの單根 0.1309…, 0.7594…, のたぐいがづい分あとになってやっと求まる。) 新しいEIGQRで相当改善はされるが、不充分である。 こういう問題に対しては、

(1) 粗いTol (例えば 10^{-4}) を使ってEIGQRを走らせる。その結果Tol値以下に近接したものは'重複'と判定される。重複度を含め所望の箇数を可なり上まわって求める。

(2) スツルムチェックにより所望の固有値は上で求めたものに全部含まれているかどうか調べる。

(3) '重複'扱いを受けた固有値グループ毎にレフト矢を えらび、今度は精しいTol (例えば 10^{-12}) によってEIGQRを走らせる。

〔文献〕

(1) Martin · Wilkinson : 'Solution of Symmetric and Unsymmetric Band Equations and the Calculations of Eigenvectors' Numer. Math. 9, 279-301 (1967)

(2) Bathe / Wilson, 菊地訳 : 有限要素法の数値計算 530頁~531頁

(3) Bathe / Ramaswamy : 'An accelerated subspace iteration method' Comp. Meths. Appl. Mech. Eng. 23 (1980) 313-331

(4) 村田 / 堀越 : '対称帯行列を三重対角化するための新アルゴリズム' 情報処理 Vol 16, No2, (1975年2月)

(5) 村田 / 二村 / 門間 '仮想記憶方式の下での大形行列計算技法' 情報処理 Vol 21, No2, (1980年4月)