数理解析研究所講究録
382巻 1980年 15-27

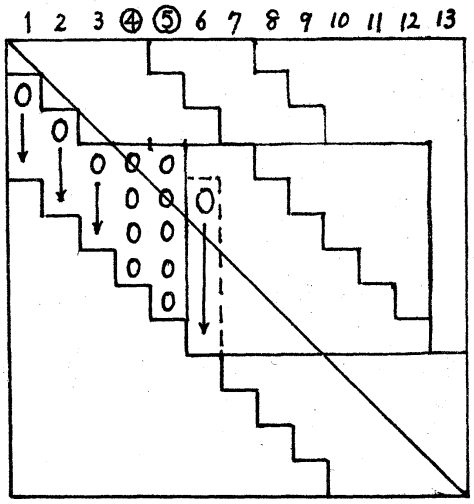
3
15

特異または特異に近い帯行列の三角化と二重対角化.

日立中研　村田　健郎

1. はじめに　ナビア・ストークス系などを有限要素法によって離散化するとき、特異な行列に見舞われることがある。そのとき、その行列の階数を知る,その他の必要を痛感したので、表記の問題をとり上げることになった。ところが、密行列に対してならば問題なく採用できるアルゴリズムが、帯に対しては特別の工夫を要することが判った。

初めにアルゴリズムの原型を次頁，次々頁に示す。次頁がガウス消去法，次々頁がハウスホルダー法に基づくものである。これらは、下図に示すように、特異のとき、帯巾がひろがる。それを、帯巾をひろげないでうまくやるよう改造しようというわけである。

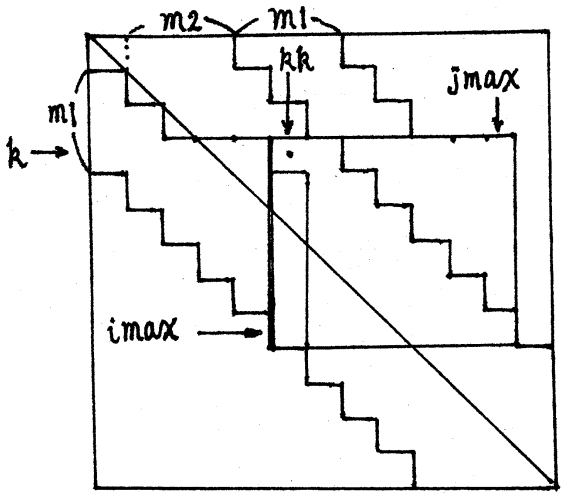


§2で帯ガウス，§3で帯ハウスホルダーに基づくものを示す。

さて、行列の特異値，すなわち A^TA の固有値を求めるために、その前段階としてハウスホルダー変換によって二重対角化を行うが、そこでも通常のアルゴリズムを帯に適用すると帯巾がひろがって実用にならぬ。それの対策法を§4に示す。その方法は、以前御紹介した対称帯行列に対する三重対角化法と類似の方法である。

プログラム GLUBR2 の原型

```
ir = 0
do   kk = 1, n
     k = kk - ir ;  imax = min(kk+m1, n)
     amax = 0.0 ;  ipk = k
     do   i = k, imax
          aik = abs(a[i,k])
          if amax < aik then
               ipk = i ;  amax = aik
     ip[k] = ipk  ;  jmax = min(kk+m1+m2, n)
     if amax > eps then
          if k ≠ ipk then
               do   j = kk, jmax   a[k,j] ⇄ a[ipk,j]
               do   i = k+1, imax
                    t = -a[i,kk] / a[k,kk] ;  a[i,kk] = t
                    do   j = kk+1, jmax
                         a[i,j] = a[i,j] + t * a[k,j]
     else ir = ir+1
```



〔プログラムの要点〕

- 消去段数 kk と、ピボット方程式番号 k とを分離し、関係：
  $k = kk - ir$
  で結ぶ。
- 方程式が特異でなければ、$k = kk$ のまゝ、終りまで推移する。
- $n - ir$ が行列の階数(rank)である。

プログラム HQRR2 の原型

```
ir = 0
do  kk = 1, n
    k = kk - ir  ;  imax = min(kk+m1, n)
    sum = 0.0
    do  i = k, imax
        sum = sum + a[i,kk]**2
    sk = sqrt(sum)  ;  jmax = min(kk+m1+m2, n)
    if  sk > eps then
        hk = sk*(sk + abs(a[k,kk]))  ;  ak = 1/hk
        if a[k,kk] < 0 then sk = -sk
        a[k,kk] = a[k,kk] + sk
        do  j = kk+1, jmax  p[j-kk] = 0.0
        do  i = k, imax
            awi = ak * a[i,kk]
            do  j = kk+1, jmax
                p[j-kk] = p[j-kk] + a[i,j]*awi
        wk = - a[k,kk]  ;  a[k,kk] = - sk
        do  j = kk+1, jmax
            a[k,j] = a[k,j] + wk*p[j-kk]
        do  i = k+1, imax
            wi = - a[i,kk]
            do  j = kk+1, jmax
                a[i,j] = a[i,j] + wi*p[j-kk]
    else ir = ir+1
```

〔ノート〕 通常のハウスホルダー三角化プログラムは、k と kk と同じものを使っている。上のプログラムでも、行列が特異でなければ、k と kk とは同じまゝで終りまで推移する。

2. 帯ガウス

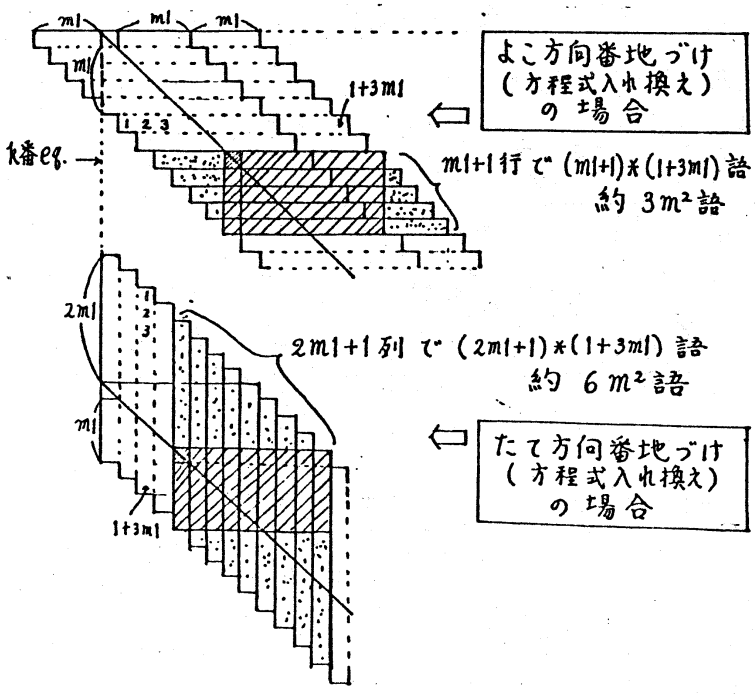
帯ガウスのうち、'方程式入れ換え方式の部分軸選択'によるものについて述べる。番地づけのし方に、よこ方向と、たて方向の二通りのし方があるが、方程式入れ換え方式にとっては、よこ方向番地づけがよい。そのとき Working Set を $2m^2$ 語におさめることができる。

ここに、mは帯半巾，Working Setとは、プログラムの主要部が実効的に要求する実メモリ容量のこと。主要部とは、最も深いループを含む二重ループの部分だと思えばよい。

〔表1〕帯ガウスとりまとめ表

番地づけ方向	Working Set (*)
よこ方向	$2m^2$ 語 ($3m^2$ 語)
たて方向	$4m^2$ 語 ($6m^2$ 語)

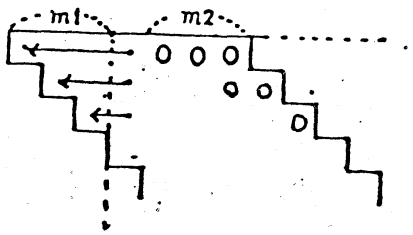
(*)内の数字は、普通のプログラム技法による場合である。(下図) これによるときは、特異のとき、帯巾がひろがってしまう。



次頁に示すアルゴリズムは、Working Set は $2m^2$ 語であり、特異のときには n-ir がその行列の階数となるよう仕組んである。(特異でないときのものは文献(1)にある。)

GLUBR2　（以下 左 off diag. を m1, 右 off diag を m2 として）

```
      ir = 0 ; m = m1+1
      do 20 i = 1, m1
          do 10 j = 1, i+m2
              ar[j,i] = ar[m-i+j,i]
20    10    ar[m-i+j,i] = 0
      do 90 kk = 1, n
          k = kk-ir ; imax = min(kk+m1, n)
          amax = 0.0 ; ipk = k
          do 30 i = k, imax
              aik = abs(ar[1,i])
              if aik > amax then
30                amax = aik ; ipk = i
          ip[k] = ipk ; jmax = min(m+m2, n-k+1)
          if amax > eps then
              if k ≠ ipk then
40                do 40 j = 1, jmax  ar[j,k] ⇄ ar[j,ipk]
              do 60 i = k+1, imax
                  t = -ar[1,i] / ar[1,k]
                  do 50 j = 2, jmax
50                    ar[j-1,i] = ar[j,i] + t * a[j,k]
                  ar[jmax,i] = 0.0
60                if ir = 0 then ac[i-k,k] = t
          else
              ir = ir+1
              do 80 i = k, imax
                  do 70 j = 2, jmax
                      ar[j-1,i] = ar[j,i]
90    80    70    ar[jmax,i] = 0.0
```



〔アルゴリズムの要点〕

do 20 のループ：　i = 1 から m1 までの行を、初めに m−i こま左に ずらす。

do 50 のループ：（特異でないとき）ガウス変換をやったのち 1こま左にずらす.

do 80 のループ：（特異のとき　）ガウス変換はやらず、1こま左にずらす.

3. 帯ハウスホルダ三角分解

帯に対して高精度計算をしたいときには、帯ハウスホルダ三角分解をすすめる。特に、特異性に関する精密診断のためには、この方が確かである。前節に示したガウスによるものでは、特異値そのものは変ってしまうのであった。

行列Aの特異値とは、行列A^TAの固有値の平方根のことであった。ハウスホルダ変換 QA は、特異値を不変に保つ。

(∵ $Q^TQ=I$ のとき、$(QA)^T(QA) = A^TQ^TQA = A^TA$

ハウスホルダ変換行列： $Q = I - a w w^T$, ここに、aはスカラー, wはベクトルで、

$$s = sqrt(a_{kj}^2 + a_{k+1\,j}^2 + \cdots + a_{k+l\,j}^2)$$

$$w^T = (0, \cdots, 0, a_{kj} \pm s, a_{k+1\,j}, \cdots, a_{k+l\,j})$$

$$a = 1/(s^2 + |a_{kj}|s)$$

を考える。通常のハウスホルダ三角化に於いては、第k段でのQ_kを、上の式のjをkにとったものを使うが、我々はこの節と次節で、kとjが大いに異なるものでも、必要に応じて自由に使う。さて、3頁の'原型'プログラムによるときは、$sk < eps$ となって ir が増すにつれて帯中がひろがる。(その様子は1頁の図の如くである。)

次頁のプログラムだとWorking Set $2m^2$語でうまく行って呉れる。特異のとき、得られた n-ir 値が行列の階数である。

HQRR2

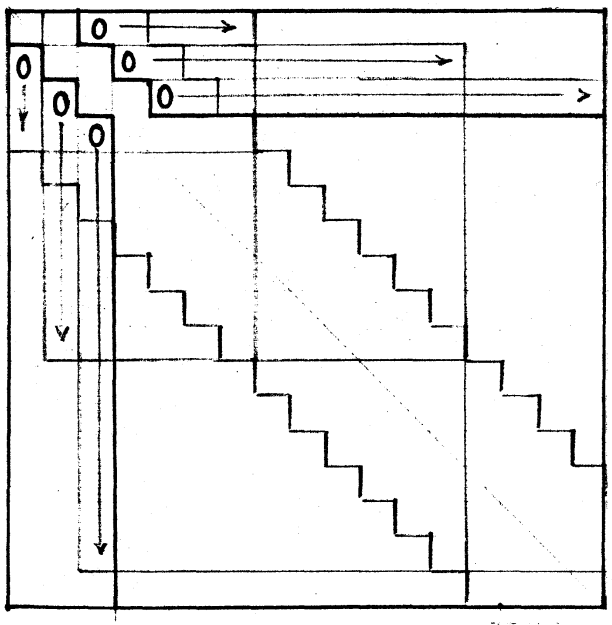
```
ir = 0 ; m = m1+1
do  i = 1, m1
    do  j = 1, i+m2
        ar[j,i] = ar[m+j-i,i] ; ar[m+j-i,i] = 0
do  kk = 1, n
    k = kk-ir ; imax = min(kk+m1, n) ; sum = 0.0
    do  i = k, imax
        sum = sum + ar[1,i]**2
    sk = sqrt(sum) ; jkmax = min(1+m1+m2, n-k+1)
    if  sk > eps then
        hk = sk*(sk+abs(ar[1,k])) ; ak = 1./hk
        if ar[1,k] < 0.0 then  sk = -sk
        ar[1,k] = ar[1,k] + sk
        if ir = 0 then
            do  i = k, imax  ac[i-k+1, k] = ar[1,i]
        do  jk = 1, jkmax  p[jk] = 0.0
        do  i = k, imax
            awi = ak*ar[1,i]
            do  jk = 2, jkmax
                p[jk] = p[jk] + awi*ar[jk,i]
        wk = -ar[1,k] ; ar[1,k] = -sk
        do  jk = 2, jkmax
            ar[jk, k] = ar[jk, k] + wk*p[jk]
        do  i = k+1, imax
            wi = -ar[1,i]
            do  jk = 2, jkmax
                ar[jk-1,i] = ar[jk,i] + wi*p[jk]
            ar[jkmax,i] = 0.0
    else
        ir = ir+1
        do  i = k, imax
            do  jk = 2, jkmax
                ar[jk-1,i] = ar[jk,i]
            ar[jkmax,i] = 0.0
```

4. 特異値分解のための二重対角化.

左から掛けるハウスホルダ変換 Q_k と 右から掛けるハウスホルダ変換 P_k を交互に $k=1,2,\cdots$, について行うことにより、もとの行列Aを、特異値を変えないで二重対角化する方法があり、特異値分解（SVD）の前処理として使用されている。ところが、この方法を帯行列に対して行うと、帯巾が途中段階でひろがって非実用的である。

途中段階に於ける帯巾の増大を一定値以内に收めてうまく二重対角化する方法を発見したのでその概要を紹介したい。

初めに、普通の方法で、帯巾が、際限なくひろがる様子を示しておこう。下図の例では $k=3$ 段目で残りの部分が完全に密行列化している。

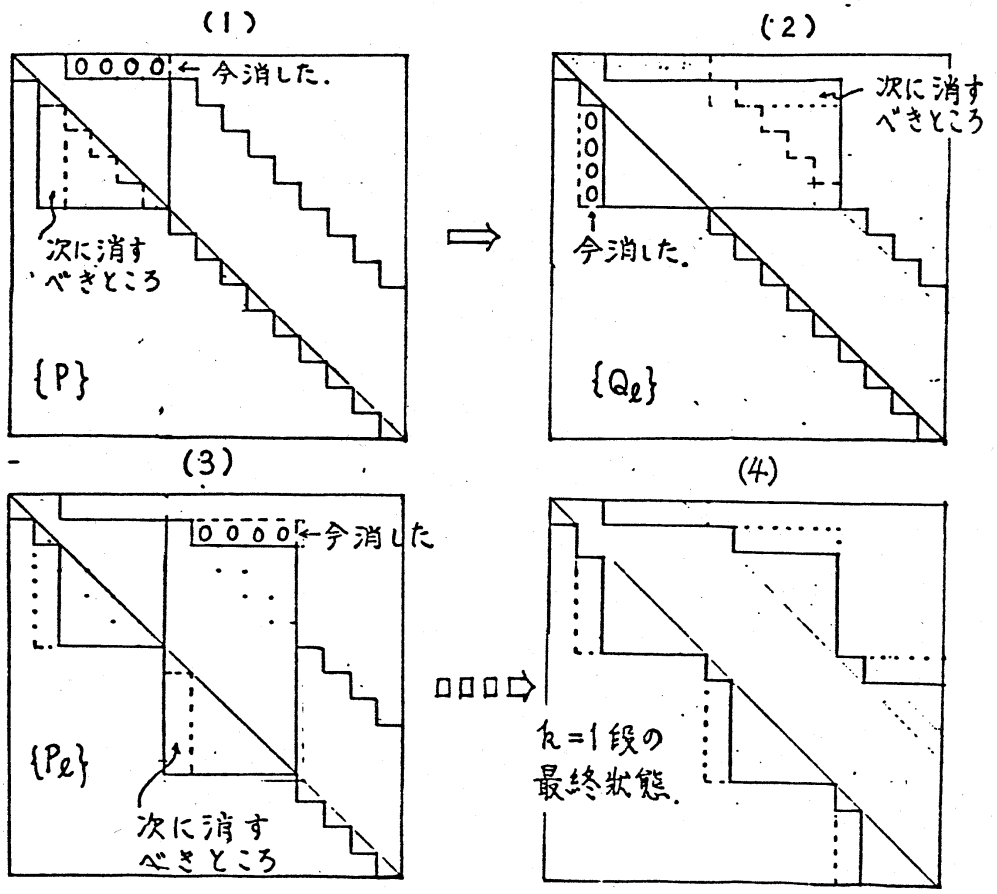


さて、初めに、前節の方法によって三角化してのち、それをうまく帯巾をあまりひろげないで二重対角化する方法を示そう。（次頁）

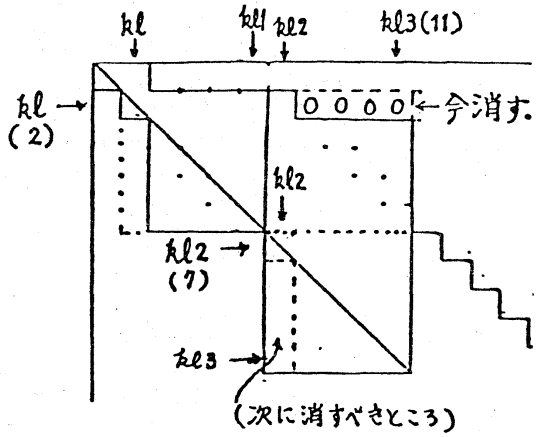
この方法は、後で示す諸方法とくらべて、所要演算ステップ数，所要メモリ容量共に大きいが、（$6mn^2$ 積和，$6mn$ 語）判り易い。

{ $l := m1+m2+1$, $l1 := m1+m2-1$, $l_2 = m1+m2$ として }

```
for k := 1 to n-2 do
    k2 := min(k+m1+m2, n) ;
{P} a[k,k+2], …, a[k,k2] を消すための変換 A := AP
                すなわち k段の二重対角化の主操作
    kl := k+1 ; { kl は k local の略. このあと二重対角化の後始末 }
    while kl < n do
        kl1 := min(kl+l1, n)
{Ql}    a[kl+1, kl], …, a[kl1, kl] を消す変換 A := Ql A
        kl2 := kl+l2 ; kl3 := min(kl2+l1, n) ;
        if kl2 < n then
{Pl}        a[kl, kl2+1], …, a[kl, kl3] を消す A := A Pl
        kl := kl+l2
```



結局，三種類のハウスホルダ変換 P，Q_ℓ，P_ℓ の組み合せによったわけである。P，Q_ℓは周知であるが、P_ℓは目新しいかもしれない。これだけをプログラムを示そう。

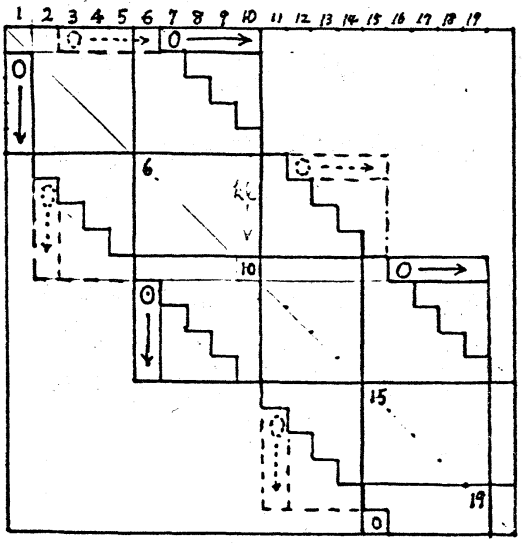


{ P_ℓ 部の詳細 }

```
sum := 0.0 ;
for j := kl2 to kl3 do
    sum := sum + a[kl, j]**2
sk := sqrt ( sum) ;
if sk > eps then
    hk := sk*(sk + abs(a[kl, kl2])) ; ak := 1.0/hk
    if a[kl, kl2]<0.0 then
        sk := -sk
    w[1] := a[kl, kl2]+sk ; a[kl, kl2] =- sk ;
    for jk := 2 to kl3-kl1 do
        w[jk] := a[kl, jk+kl1]
    for i := kl+1 to kl3 do
        sum := 0.0 ;
        for jk = 1 to kl3-kl1 do
            sum := sum + a[i, jk+kl1]*w[jk]
        p[i-kl] := ak*sum
    for i := kl+1 to kl3 do
        pi := -p[i-kl] ;
        for jk := 1 to kl3-kl1 do
            a[i, jk+kl1] := a[i, jk+kl1] +pi* w[jk]
for j := kl2+1 to kl3 do
    a[kl, j] := 0.0
```

次に直接二重対角する方法を示す。こちらは複雑な代りに $4mn^2$ 回の積和，所要メモリ $4mn$ 語である。

```
for k:=1 to n-1 do
    km1 := min(k+m1, n)
    a[k+1,k] 以下を消すため、w=(0,…,a[k,k]±s,…,a[km1,k],0…)^T による Q=(1-αwwT) による変換 A:=QA を行う。   S1
    km2 := k+m2 ;
    if km2 < n then
        a[k,km2+1]…(より右)を消す変換 A:=AP を行う。   S2
    kl := min(k+m2+m1, n) ;
    while kl<n do;
        kl1 := min(kl+m1, n) ;
        a[kl+1, kl-m1] 以下を消す変換 A:=QA を行う。   S3
        kl2 := kl+m2 ;
        if kl2 < n then
            a[kl, kl2+1], …(より右) を消す変換 A:=AP を行う。   S4
        kl := kl+m2+m1
    a[k,k+2] …(より右) を消す変換 A:=AP を行う。   S5
    kl := min(k+m1+1, n) ;
    while kl<n do
        a[kl+1, kl-m1] 以下を消す変換 A:=QA を行う。   S6
        if kl+m2 < n then
            a[kl, kl+m2+1] …(より右) を消す変換 A:=AP を行う。   S7
        kl := kl+m2+m1
```



左図は、$m1=4$, $m2=5$ のときのもの。

- S1, S2 を行ったのち、$kl=10$ となり、S3, S4 を行い $kl=19$ となる。S3 を行い S4 は行なわず $kl=28$ となる。。次いで S5 を行い $kl=6$ となり，S6，S7 を行い $kl=15$ となり …… 。

- 実線が、a[k+1,k] 以下を消すこと、およびそれに伴う あと始末を示している。
  点線が、a[k,k+2],…(より右)を消すこと、およびそれに伴う あと始末を示している。

この、直接二重対角化の方法は、前の三角化経由の方法とくらべて、演算時間，所要総メモリ容量，Working Set共に小さいのだが、ページスワップの統ページ数はむしろ大きい。各k段毎に、帯の右下端まで二回ページスワップをひきおこすがゆえである。(帯中は$\frac{2}{3}$だが、それが2倍されて$\frac{4}{3}$倍)

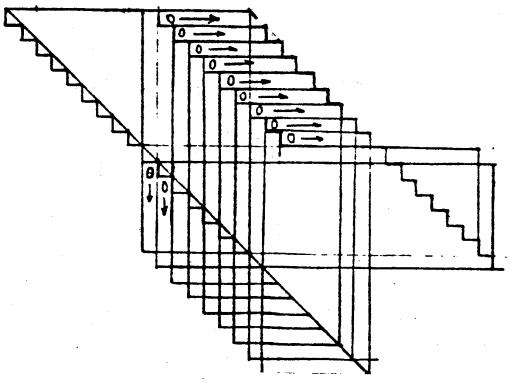
第三の方法は、三角化→三角の帯を半分に縮小→二重対角化という手順をふむものである。これは，$5mn^2$積和，$4mn$語であるが、ページスワップに関しては最も有利である．

$4mn$語が、所与の実メモリ容量を大巾に超えるときには、この方法が最も実用的である。次頁に、三角の帯を半分に縮小するアルゴリズムの大筋を示す。

〔附記〕 実は、三角化されたものを二重対角化するアルゴリズムで、各k段毎にQ_ℓ変換とP_ℓ変換を右下端までやらないで、必要最小限のところまで進めたら次の段のP,Q_ℓ,P_ℓ変換を並行的に行うことにすれば、ページスワップは半減する。(例えば、9頁下段の図の(3)の次のQ_ℓを行ったら、$k=2$に対するP変換をやる。以下、$k=1$段のQ_ℓ,P_ℓ変換と、$k=2$段のQ_ℓ,P_ℓ変換を並行列に行う。)

他のアルゴリズムに対しても同様の技巧が適用可能である。しかし、こういう技巧は、ターンアラウンドタイムが我慢ならぬ程に長くなったとき、止むを得ずやることであろう。

```
l1 := m1+m2 ; l := l1+1 ; mh1 := l1 div 2 ; mh2 := l1-mh1 ;
k := 1 ;
while k+mh2<n do
    k21 := max(k+mh2+mh1, n-1)
    for kk := k+mh2 to k21 do
        a[kk-mh2, kk+1], …(の右)を消す A := AP 変換.
        P = I - αwwᵀ に使う wᵀ = (0, …, 0, a[k-mh2±s, kk], …)
    kl := k+mh2 ;
    while kl<n do
        for ii := kl to min(kl+mh1, n-1) do
            a[ii+1, ii], …(の下)を消す A := QA 変換
        kll := kl+l1 ;
        if kll<n then
            for jj := kll to min(kll+mh1, n-1) do
                a[jj-l1, jj+1], …(の右)を消す A := AP 変換
        kl := kl+l1
    k := k+mh2
```



〔文献〕

1) J.H. Wilkinson・C. Reinsch Linear Algebra (1971) Springer
Contribution I/6, I/8, I/10