解　説

仮想記憶方式の下での大形行列計算技法†

村田健郎†† 二村良彦†† 門間三尚†††

1. はじめに

ここでは表記技法をFortranプログラムに適用するとして解説するが，若干 Pascal 寄りの記法を採用したい．理由は読み易さと紙数節約にある．初めに周知の部分軸選択* ガウス消去法を示して読者の便に供しよう．但し‘交換’のために略号‘⇄’を使う．

[Prog. 1]

```
      ir=0
      do 50 k=1,n
        amax=0.0
        do 10 i=k, n
          aik=abs(a[i,k])
          if amax<aik then
   10       ipk=i; amax=aik
        ip[k]=ipk
        if amax>eps then
          if k≠ipk then
          20 do 20 j=k, n a[k,j]⇄a[ipk,j]
          do 40 i=k+1,n
            t=-a[i,k]/a[k,k]; a[i,k]=t
            do 30 j=k+1, n
          40; 30   a[i,j]=a[i,j]+t*a[k,j]
   50     else ir=ir+1
```

ここに，Aはn行n列の正方行列である．

実は Prog. 1 は Fortran にとってはよくない．実メモリ容量を超える行列を扱うと，CPU タイムに比べて，USE タイムが数千倍に及ぶことが普通となる．原因は **do** 30 のループにおいて $a[i, j]$ の右側の添字 j が動くところにある．j がひとつ進むごとに主メモリ上の番地は n 番地進む．たとえば n=500 のとき，文 $a[i, j]=a[i, j]+t*a[k, j]$ を1回行うごとに1頁**進む．したがって $n(n-k)$ 語が主メモリ内に収まるまでの間1回の $a[i, j]=a[i, j]+t*a[k, j]$ の演算ごとに主メモリとディスク間のページスワップが起るわけである．(1 μs の演算対 10 ms のページスワップの関係.)

さて，こういうひどいことを起させぬようにするには，**if** $amax>eps$ **then** の枝を次のように直せばよい：

```
      if k≠ipk then a[k,k]⇄a[ipk,k]
      do 20 i=k+1,n
   20   a[i,k]=-a[i,k]/a[k,k]
      do 40 j=k+1, n
        if k≠ipk then a[k,j]⇄a[ipk,j]
        t=a[k,j]
        do 30 i=k+1,n
   40; 30   a[i,j]=a[i,j]+a[i,k]*t
```

実は，この改造されたプログラムによっても，現在標準となっているマルチジョブの環境下では，待ち時間は我慢のならぬものとなり勝ちである．その対策は簡単ゆえ次章で軽くふれるに止め，本稿では主として問題の多い帯行列関連のプログラムについて解説したい．応用上基本的な帯ガウスと帯ハウスホルダ変換に焦点をしぼる．

2. たてブロックガウス

n^2 語が主メモリ容量 M_R 語を超えるとき，$n \cdot m < M_R$

```
for kb=1 to kbe do begin
    for k=1 to m*(kb-1) do {kb=1 のときはやらぬ}
{A部}  元来なら済んでいる可き m*(kb-1) 段までの，第 kb ブロックに対するガウス変換を今行う．
    for k=m*(kb-1)+1 to min(m*kb,n) do
{B部}  通常のガウスと同じことを第 kb ブロック内だけについて行う．
end {of kb}
```

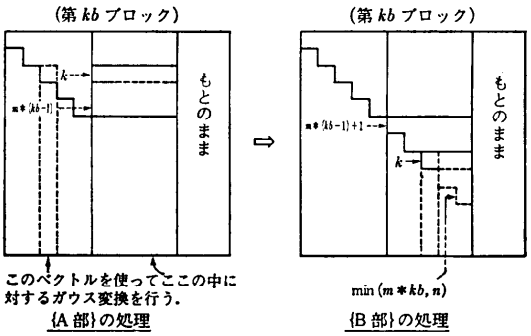


図-1

† Large Matrix Computation for Virtual Storage Systems by Kenro MURATA and Yoshihiko FUTAMURA (Central Research Laboratory, Hitachi, Ltd), Mitsutaka MONMA (Software Works, Hitach, Ltd.).

†† (株)日立製作所中央研究所

††† (株)日立製作所ソフトウェア工場

* partial pivoting

** 4キロバイト

なる整数 m をえらび，m 個の列ごとにまとめて1ブロックとし，第1ブロックから最終の第 kbe ブロックまで kbe 個*のブロックに行列Aを区切って，**図-1** に示すアルゴリズムで行うと，待ち時間を大幅に縮めることができる．また，このアルゴリズムによると，普通のガウスによるのと全く同じ答が得られる．

3. 帯ガウス

帯ガウスにはいろいろのやり方があるが，ここでは方程式入れ換え方式の部分軸選択によるものだけについて述べる．それでも番地づけのし方に二通りある．よこ方向番地づけとたて方向番地づけである．**表-1** に見るように，よこ方向番地づけの方が方程式入れ換え方式の部分軸選択にはマッチしている．ワーキングセットを $2m^2$ 語に収めることができる．ここに m は帯半幅でありワーキングセットとは，そのプログラムの主要部が実効的に要求するところの実メモリ容量のことだとしておく．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のワーキングセットの欄の（ ）内のものは，通常のプログラム技法による場合である．

帯ガウスには，密ガウスと様子のちがう点がいくつかある．

1) ガウス三角分解だけのときのワーキングセット $2m^2$ 語と，三角分解結果を反復利用するときのワーキングセット $3m^2$ 語との開きが大である．密行列の普通のプログラムでは，三角分解のワーキングセットは $n(n-k)$ 語（k は消去段数），反復利用時のワーキングセットは n^2 語である．

2) 内積ガウス（クラウト法）は，密行列に於いては n^2 語が主メモリに入ればその長所を享受できた．帯に対しては，部分軸選択とプログラムの局所参照性とが両立しない．うまく行くのは，対角優位行列のような，部分軸選択をしなくてよい場合に限られる．

3) 特異な行列に遭遇したとき，その階数を調べるのに，密行列の場合には容易に手当てができるが，帯

表-1 帯ガウスとりまとめ表（帯半幅を $m(=m1+1)$ とする）

番地づけ方向	ワーキングセット	データ格納規則
よこ方向	$2m^2$ 語 ($3m^2$ 語)	$ar[m+j-i,i]=a_{ij}$
たて方向	$4m^2$ 語 ($6m^2$ 語)	$ac[m+i-j,j]=a_{ij}$

* ガウス三角分解：LA=U の下三角部を L，上三角部を U とする．

$m1=3, m2=3, n=14$　$kk=3$ で $ir=1$，$kk=4$ で $ir=2$，$kk=7$ で $ir=3$ となる．

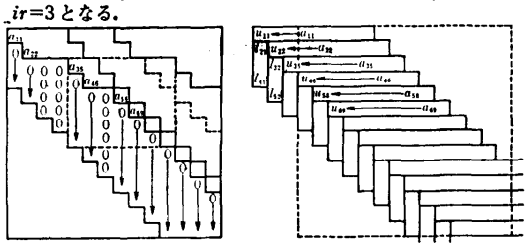
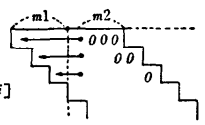


図-2

の場合には特別の配慮を要する．

次の Prog. 2 は，ワーキングセット $2m^2$ 語で，特異でない行列に対してはL部*の a_{ik} を $ac[i-k,k]$ に，U部*の a_{kj} を $ar[j,i]$ に作り上げる．特異な行列に対しては，その階数を $n-ir$ によって定める．**図-2** は $n=14$, $ir=3$ の例である．普通にプログラムしたのでは図-2左で示すように帯幅がひろがるのを，Prog. 2 では巧妙に帯の中で始末をつけているわけである（図-2右）．$n-ir$ が行列の階数である．

```
[Prog. 2]
    ir=0; m=m1+1
    do 20 i=1, m1
      do 10 j=1, i+m2
        ar[j, i]=ar[m-i+j, i]
10      ar[m-i+j, i]=0
20  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
          do 40 j=1, jmax ar[j, k]⇄ar[j, ipk]
40        do 60 i=k+1, imax
            t=-ar[1, i]/ar[1, k]
            do 50 j=2, jmax
              ar[j-1, i]=ar[j, i]+t*ar[j, k]
50          ar[jmax, i]=0.0
            if ir=0 then ac[i-k, k]=t
60      else
          ir=ir+1
          do 80 i=k, imax
            do 70 j=2, jmax
              ar[j-1, i]=ar[j, i]
70          ar[jmax, i]=0.0
80  continue
90  end
```



[注意]　ガウスの消去変換は，行列の特異値すなわち A^TA の固有値を不変に保たない．したがって，たちのひどくわるい問題を解くとか，行列の階数を精密に決めようとかのときには，次章で示すハウスホルダ変換による三角分解に席をゆずらねばならぬことがあ

る．

4. 帯ハウスホルダ三角分解

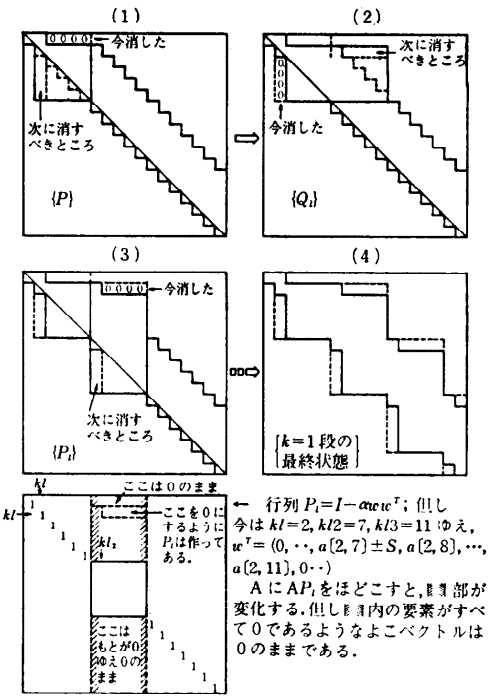
帯ハウスホルダ三角分解も，帯ガウスと同様に，よこ方向番地づけのとき $2m^2$ 語，たて方向番地づけのとき $4m^2$ 語のワーキングセットとすることができる．次に示す Prog. 3 は，特異な行列に対しては $n-ir$ によって階数を教えるようにできている．特異でない行列に対しては，$ac[\ ,k]$ に，ハウスホルダ変換行列 $I-\alpha_k w_k w_k^T$ 作成のためのベクトル w_k を収容する．これはたとえば反復計算 $Ax^{(\nu)}=Bx^{(\nu-1)}+b$ への応用を考慮してのことである．紙面節約のため，Prog. 2 の **do** 20 の終端までのところが省略してある．また，**do** 文に番号をまだ入れていないものを示した．

[Prog. 3] {**do** 20 のループまでは Prog. 2 と同じとして}

```
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(m+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]
    ark=-ar[1,k]; ar[1,k]=-sk
    do jk=2, jkmax
      ar[jk,k]=ar[jk,k]+ark*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
```

5. 特異値分解 (SVD) 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 $k=1,2,\cdots$ について行うことによって，もとの行列 A の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 (Singular Value Decomposition, 略して SVD) の前処理として使用されている．ところが，通常行われている方法を帯行列



← 行列 $P_l=I-\alpha ww^T$; 但し今は $kl=2, kl2=7, kl3=11$ ゆえ，$w^T=(0,\cdots,a[2,7]\pm S, a[2,8],\cdots, a[2,11],0\cdots)$

A に AP_l をほどこすと，■■部が変化する．但し■■内の要素がすべて0であるようなよこベクトルは0のままである．

図-3

[Prog. 4] {$l2=m1+m2$; $l1=l2-1$; $l=l2+1$ として}

```
for k=1 to n-2 do
  k2=min(k+l2,n);
{P} a[k,k+2],...,a[k,k2] を消すための，P=(I-αww^T)による A=AP. すなわち k 段を二重対角比.
  kl=k+1; {kl は klocal の略. このあと二重対角化の後始末}
  while kl<n do
    kl1=min(kl+l1,n);
{Q_l} a[kl+1,kl],...,a[kl1,kl] を消す変換 A=Q_lA
    kl2=kl+l2; kl3=min(kl2+l1,n)
    if kl2<n then
{P_l}   a[kl,kl2+1],...,a[kl,kl3] を消す A=AP_l
    kl=kl+l2
```

{P_l 部の詳細}

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

に対して行うと，帯幅が急激にひろがって途中密行列化する．帯幅の増大を一定値以内に制限してうまく二重対角化することができることを最近発見したので，その方法のひとつを紹介したい．もとの行列の元数を n，帯の全幅を $l=1+m1+m2$ とする．幅 l のワクの中で，前章 Prog. 3 によって三角化してのち，上三角部の非対角要素を残した二重対角化を行う場合について示そう．もとの帯行列から直接に二重対角化を行う方法もあるが，その場合の方が若干手がこんでいるので，紙面の都合上，簡単な方法の方だけ示すわけである．図-3 がその説明図である．ここでは $k=1$ 段目だけ示した．Prog. 4 にアルゴリズムの大筋を示す．ハウスホルダ変換のプログラムを作成したことのある方ならば，これから本当のプログラムを作ることは容易であろうが，変換 $\{P_i\}$ 部だけは細部を示しておいた．

6. 帯行列に対する固有値解析

$Ax=\lambda x$ 形の対称帯固有値解析を行うため，標準的なハウスホルダ三重対角化法をほどこすと，途中段階で密行列化する．それをさけるために，前節類似の技法を用いることによって，帯のワク内で三重対角化する方法が文献 3) に示してある．

文献 4) のランチョス法（の変種）は，筆者の追試の限りでは大変うまく行く．しかもこれは $Ax=\lambda Bx$ 形にも応用が利くので注目に価するが，ランチョス法は汎用性に関していまひとつ評価が定まらない．

通常のハウスホルダ法によるのと，文献 3) の方法によるのとでどれぐらいちがうかの実測例を示そう．帯半幅 40，元数 1,000 の帯のとき，実メモリを 650 KB 与え，積和ステートメントを 1.2 μs，ページスワップを 12 ms で行うシステム (HITAC 8800) で，

通常ハウスホルダ：

CPU タイム 30 分，USE タイム 200 分

文献 3) の法：

CPU タイム 5 分，USE タイム 5 分

であった．元数 2,000 となると通常ハウスホルダ法では不可能であるが，文献 3) の法では CPU タイム 20 分，USE タイム 120 分で答を出すことができた．

帯の幅がせまく，所要固有値の個数が比較的すくないときは，帯に対する直接 QR 法〔文献 2）の II/7〕もよいであろう．文献 3) ほどには汎用性に欠けるが，同時反復法（サブスペース法），スツルム逆反復法も，帯のワク内で処理可能である．

$Ax=\lambda Bx$ 形の対称帯となると汎用性と仮想メモリ向きの両特性を備えたうまい方法がない．汎用性に関してそれぞれに問題はあるが，

スツルム逆反復法，サブスペース法，ランチョス法のそれぞれは，帯のワク内で仮想メモリ向きのプログラムを作ることができる．

非対称帯固有値解析となると，$Ax=\lambda x$ 形でも A の要素の僅かの変動に対して固有値，固有ベクトルが大きく変動する問題がごく普通にあらわれるから，汎用的なプログラムはますます難しい．比較的汎用性の高いのは，ハウスホルダ変換によってヘッセンベルグ行列化してのち QR 法に乗り換えるという系統のものである．しかし，通常の方法でやるとヘッセンベルグ行列化の途中で密行列化する．前節類似の技法を駆使することによってワーキングセットだけは抑えることができるが，右上半部の密行列化はさけられない．

7. おわりに

紙数の関係で，詳しく述べることはできなかったが一応最も基本的な，ガウス消去変換と，ハウスホルダ変換関連の事項については述べたつもりである．特に関係の深い文献だけをあげて終りとしたい．

参考文献

1) Wilkinson: The Algebraic Eigen value Problem, Oxford (1965).
2) Wilkinson, Reinsch: Linear Algebra, Springer (1971).
3) 村田，堀越：対称帯行列三重対角化，情報処理，Vol. 16, No. 2 (1975).
4) Paige: Computational Variants of the Lanczos, Meth. J. IMA (1972).
5) Sparse Matrix Computations, Academic P. (1976).
6) 山本哲朗：数値解析入門，サイエンス社.
7) Forsythe, Moler: 渋谷，田辺訳：計算機のための線形計算の基礎，培風館.

（昭和 54 年 12 月 3 日受付）