[PROC. OF JSCE Structural Eng. /Earthquake Eng. Vol. 2, No. 1 April 1985]

FRONTAL-SKYLINE METHOD FOR UNSYMMETRIC MATRICES

By Yoji SHIMAZAKI

This paper presents a frontal-skyline method for reducing unsymmetric matrices which frequently arise in applying the finite element method to boundary value problems. The method makes use of the basic frontal procedure but allows the front to increase in size using compact skyline storage whenever core storage is available.

1. INTRODUCTION

If the systems of equations have sparse coefficient matrices, the skyline (profile, envelope, or variable bandwidth) storage scheme is an efficient technique in order to reduce both the computation time and the storage requirements. Various kinds of skyline methods for both symmetric and unsymmetric matrices can be found as an in-core solver[1)~3)]. The backing store (tapes or discs), however, should be utilized when the systems of equations are very large.

A frontal-skyline method was first developed for a symmetric finite element matrix[4)]. This method never requires more storage than that needed by the standard frontal method[6)] and requires fewer transfers to and from disc than either the frontal method or the blocked-skyline method[3)].

Many problems such as met in fluid mechanics have unsymmetric stiffness matrices when these are formulated by a finite element method. Hood[5)] has developed a frontal solution technique for unsymmetric matrices, which was based on the symmetric version of the scheme[6)]. Unfortunately this scheme also requires that the total number of equations in core storage be kept small, which in turn increases the number of tape segments necessary for a given problem.

Compact skyline storage can also be used for an unsymmetric matrix. This storage scheme reduces the total number of tape records for a given problem, and also increases the efficiency of the LDU decomposition routine.

There are four major subprograms associated with the method, which have many features in common with the symmetric matrix programs. The first is a prefrontal routine referred to as WAVE. The second is that part of the finite element program associated with the assembly of the stiffness matrix. The third is a

* Member of JSCE, Ph. D., Assistant professor, Department of Civil Engineering, Tokai University (1117 Kitakaname Hiratsuka, Kanagawa)

subroutine called SLIDE which rearranges the stiffness matrix between tape segments (records or blocks). And, the fourth is LDU decomposition and back substitution. Each of these will be discussed, although a listing of WAVE will not be given because of its length.

2. PROGRAM WAVE

This prefrontal program designs each tape segment for the assemblage and decomposition of the stiffness matrix. It is responsible for three major organization tasks : (1) selection of the order in which the equations will appear in each tape segment, (2) storage arrangement used in the stiffness matrix for each tape segment, and (3) how the stiffness matrix will be rearranged between tape segments. The equations are arranged such that the fully assembled, or completed equations in any tape segment appear before those that are not yet completed. Furthermore, the incomplete equations are placed in the same relative sequence that they will have as completed equations. This last condition is very important because it insures that no two equations will have to exchange locations during the rearrangement of the stiffness matrix. We will see shortly that it is also a necessary requirement if the equations are to move "forward" during the rearrangement.

To accomplish this ordering of the incomplete equation, it is first necessary to establish the order in which the equations will be completed. This is accomplished at the very start of WAVE by simply going through the specified element order and checking the connectivity of the elements. As each element is added, the nodal point variables which become completed are added to the list of completed equations. Once the final order of all equations is established, the order in which these equations appear in any given tape segment is easily determined.

Once the order in which the equations appear in a given tape segment is established, the volume needed to store the stiffness matrix by columns is easily calculated. If the maximun volume specified has not been exceeded, a new element is added. If the maximum volume has not been exceeded, the previously added element is removed and the tape segment is complete. Because the removal of the last element usually removes more than one equation, and the equations are usually of different lengths, seldom is the total specified storage completely taken. Under these conditions it is necessary to right justify the stiffness coefficients. That is, the unused portion of the stiffness matrix always appears at the beginning of the array. This requirement, together with the requirement that the incomplete equations be in the same sequence as they will be when completed, normally guarantees that during the rearrangement of the stiffness matrix, transfer of coefficients will always be forward in the array.

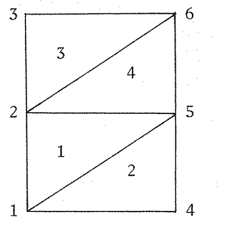
Fig. 1 illustrates the philosophy of the frontal-skyline method for unsymmetric matrix using a simple example to be used for other subroutines. In the example the number of variables of each nodes is assumed one. Fig. 1 ①shows the first step of WAVE where IORDER is the array to specify the order of selecting the elements. LIST is the array to specify the order in which the equation will be completed. IDIAG is the array designating the location in the stiffness array of diagonal terms if the tapes are not to be used. NPRC or NPR is the array designating the location in the current tape segment of equation representing the first variable associated with each nodal point. If NPR$\leqq 0$, node I does not appear in the current tape segment. Fig. 1② shows two tape segments required when the available incore storage for stiffness matrix is 14. Where IDIAG 1 is the array designating the location in the stiffness array of diagonal terms from the previous tape segment. In LDU 1, IDIAG 1 is used for designating the location of nonzero terms in the stiffness array. IDIAG 2 is the array designating the location in the stiffness array of diagonal terms for the current tape segment. MOVE is the array designating how previous stiffness matrix should be rearranged for storage arrangement of the current tape segment. ICOMP is the number of rows (or equations) which are fully assembled for the current tape segment. For simplicity, the total specified storage is completely taken by stiffness coefficients in this example.

① Determination of the order

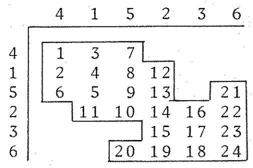
```
ELEMENT        NODAL POINT
     1          1    5    2
     2          1    4    5
     3          2    6    3
     4          2    5    6

IORDER:   2   1   4   3

  LIST:   4   1   5   2   3   6
 IDIAG:   1   4   9  14  17  24
  NPRC:   2   4   5   1   3   6
```



Finite element mesh

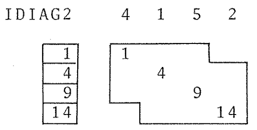


VOLUME= 24

Skyline storage arrangement

② Determination of the tape segments (MAXVOL=14)

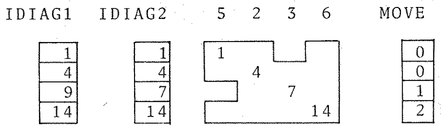
TAPE1



```
ELEMENT        NODAL POINT
     2          1    4    5
     1          1    5    2

  LIST:  4   1   5   2
 IDIAG:  1   4   9  14
   NPR:  2   4   0   1   3   0
 ICOMP:  2
```

TAPE2



```
ELEMENT        NODAL POINT
     4          2    5    6
     3          2    6    3

  LIST:  5   2   3   6
 IDIAG:  1   4   7  14
   NPR: -1   2   3  -1   1   4
 ICOMP:  4
```

Fig. 1 Example problem for subroutine WAVE.

3. ASSEMBLY OF THE STIFFNESS MATRIX

The assembly of the stiffness matrix is straightforward and is shown in Fig. 2 and 3. The $(SK)_e$ matrix is the fictitious nonsymmetric element stiffness which was made for illustrative purposes.

4. SUBROUTINE SLIDE

The success of the solution technique presented in this paper relies on the two stipulations already mentioned : (1) that all equations in a given tape segment appear in the same relative position that they will have in any other tape segment (that is, if equation "a" comes before equation "b" in a given tape segment, it will come before equation "b" in all subsequent tape segments, although they may become more separated by additional equations), and (2) that all vacancies in the stiffness matrix due to the assigned storge not being fully utilized, be at the beginning of the matrix. When both of these stipulations are met, each column will move forward during the rearrangement of the matrix between tape segments.

Routine SLIDE is a simple code which performs this task. Recall that the array MOVE specifies which new column each old column must be moved to. It likewise specifies which new row each old row is moved to. In Fig. 3 the first two entries of MOVE are zero, which indicates that these columns (and rows) were

```
(TAPE1)

    ELEMENT         NODAL POINT                       [1  4  4]
          2         1    4    5             (SK)e  =  [3  1  4]
          1         1    5    2                       [3  3  1]

                     Place in large SK matrix

                     LIST     4     1     5     2

                          | 1 | 3   | 4   |
                          | 4 | 1+1 | 4+4 | 4 |
                          | 3 | 3+3 | 1+1 | 4 |
                              |   3 |   3 | 1 |

                        Forward elimination
                           (CALL LDU1)

     ICOMP=2

(2,1row)                                     (3,2row)
2row      :    4.0   2.0   8.0   4.0         3row        :   -3.0 -10.0   4.0
1row x4.0: -   4.0  12.0  16.0               2row x 0.3: -  -3.0  -2.4   1.2
               0.0 -10.0  -8.0   4.0                           0.0  -7.6   2.8

(3,1row)                                     (4,2row)
3row      :    3.0   6.0   2.0   4.0         4row        :    3.0   3.0   1.0
1row x3.0: -   3.0   9.0  12.0               2row x-0.3: -   3.0   2.4  -1.2
               0.0  -3.0 -10.0   4.0                           0.0   0.6   2.2

                  | 1.0 |   3.0 |  4.0 |
                  | 4.0 | -10.0 | -8.0 | 4.0 |
        (SK)  =   | 3.0 |   0.3 | -7.6 | 2.8 |
                        |  -0.3 |  0.6 | 2.2 |
                                   /
                      Tape or Disk
```

Fig. 2 Subroutine POISSON for TAPE 1.

```
(TAPE2)

    ELEMENT         NODAL POINT                        [1  4  4]
          4         2    5    6             (SK)e  =   [3  1  4]
          3         2    6    3                        [3  3  1]

    IDIAG1                                         IDIAG2    MOVE
    |  1 |   |    |    |      |                    |  1 |    | 0 |
    |  4 |   |    |    |      |     |              |  4 |    | 0 |
    |  9 |   |    |    | -7.6 | 2.8 |              |  7 |    | 1 |
    | 14 |        |    |  0.6 | 2.2 |              | 14 |    | 2 |

                       (CALL SLIDE)

                 LIST     5      2      3      6
                       | -7.6 | 2.8 |      |      |
                       |  0.6 | 2.2 |      |      |
                       |      |     |      |      |
                       |      |     |      |      |

                    Place in large SK matrix

         | -6.6 | 5.8 |     | 4.0 |
         |  4.6 | 4.2 | 4.0 | 8.0 |      Add element 4 and 3
                | 3.0 | 1.0 | 3.0 |
         |  3.0 | 6.0 | 4.0 | 2.0 |

                       Forward elimination
                          (CALL LDU1)

         | -6.6   | 5.8   |        |  4.0   |
         | -0.700 | 8.242 |  4.0   | 10.788 |
                  | 0.364 | -0.456 | -0.926 | ——→Tape or Disk
         | -0.455 | 1.048 |  0.419 | -7.079 |
```

Fig. 3 Subroutine POISSON for TAPE 2.

completed in tape segment 1 and will not be moved to a new location for the current tape segment. IDIAG 1 locates the diagonal term of the colum before it has been moved and IDIAG 2 locates the diagonal term of the column for its new location. Referring to Fig. 3, it is seen that column 4 will be moved to column 2. Hence, this column, whose diagonal term is located at IDIAG 1(4)=14, will be moved to its new location where its diagonal will be located at IDIAG 2(2)=4. It is also seen that the first two terms from the top of the column belong to equations which have already been eliminated, i.e., MOVE(1)= MOVE(2)=0, and therefore, need not be moved. The third term from the top will be moved to the location representing row 1 and the third term from the left will be moved to the location representing column 1, i.e., MOVE(3)=1.

5. LDU DECOMPOSITION

It is the LDU decomposition of the stiffness matrix which places the most severe test as to the efficiency of any particular solution method. Fig. 4 gives the FORTRAN listing of the LDU decomposition. The outer DO-loop (loop I) goes from the first row to the last completed row of the tape segment. For each row, the addresses of the nonzero terms on that row are placed in the IDIAG 1 array. The second DO-loop (loop J), now passes through the IDIAG 1 array, beginning with the first term past the current row of the outer loop (i.e. I+1). If, during this search it encounters a zero element, it moves on to the next location. Once a nonzero term is found, the innermost loop (loop K) is activated. This loop now proceeds to subtract row I, multiplied by the appropriate factor, from row J. The K-loop begins a second search of IDIAG 1, beginnig at J+1. Once again, when zero terms are encountered, the K-loop passes on to the next location. When a nonzero term is encounterd, it is the address of the nonzero term of the stiffness matrix (Ith row) which must be multiplied by the appropriate factor and subtracted from the corresponding term on the Jth row. When the FORTRAN variable, LU, equals 1, an LDU decomposition and forward substitution will take place in LDU 1. When this variable equals zero, only a forward substitution will be performed.

After the stiffness matrix for each tape segment has been decomposed into its LDU components, the entire matrix is read onto disc including the rows not yet completed as well as those that are. Unfortunately, it is not practical to read only the completed rows onto disc when column storage is used. However, this procedure does not increase the total central processor storage needed for the solution, nor the number of records that must be written and read. It does increase the amount of auxiliary storage needed and the amount of central processor time required to read and write the larger records.

```
      SUBROUTINE LDU1
     1 (SK,F,LU,NEQ,ICOMP,IDIAG1,
     2 IDIAG2,IEQS,IDA,IDD,IDH)
C
      DIMENSION
     1 SK(IDH),F(IDA),IDIAG1(IDD),
     2 IDIAG2(IDD),IEQS(IDD)
C
      IEND=ICOMP
      IF(IEND.EQ.NEQ) IEND=NEQ-1
      DO 500 I=1,IEND
C
      JCHK=0
        J1=0
      DO 350 J=1,NEQ
      J2=IDIAG2(J)-(J-I)
      IDIAG1(J)=0
      IF(J2.LE.J1) GO TO 340
      JCHK=1
      IDIAG1(J)=J2
  340 CONTINUE
        JP=J+1
      IF(JP.GT.NEQ) GO TO 350
      JH=((IDIAG2(JP)-IDIAG2(J))-1)/2
      J1=IDIAG2(J)+JH
  350 CONTINUE
      IF(JCHK.EQ.0) GO TO 500
C
      SKII=SK(IDIAG2(I))
        IP=I+1
      DO 400 J=IP,NEQ
      J1=IDIAG1(J)
      IF(J1.EQ.0) GO TO 400
      J2=IDIAG2(J-1)+(IDIAG2(J)-J1)
      IF(LU.EQ.0) GO TO 390
      SK(J2)=SK(J2)/SKII
C
      DO 380 K=IP,NEQ
      K3=IDIAG1(K)
      IF(K3.EQ.0) GO TO 380
      K2=IDIAG2(K)-(K-J)
      IF(K.LT.J) K2=IDIAG2(J-1)+(J-K)
      SK(K2)=SK(K2)-SK(K3)*SK(J2)
  380 CONTINUE
C
  390 CONTINUE
      JEQ=IEQS(J)
      IEQ=IEQS(I)
      F(JEQ)=F(JEQ)-SK(J2)*F(IEQ)
C
  400 CONTINUE
C
  500 CONTINUE
      RETURN
      END
```

Fig. 4 Subroutine LDU1.

After the LDU decomposition, a subroutine for the back substitution, LDU 2, will be called. In LDU 2 routine, each segment will be read in the reversed order to obtain a result.

6. NUMERICAL EXAMPLE

In order to demonstrate the present program, the finite element mesh shown in Fig. 5 is used. In this example, it is assumed that only one variable is associated with each node and that the maximum available storage for the stiffness matrix is 800.

Fig. 6 shows the final order in which the equations will be completed. The figure also shows the total number of elements, the total number of nodal points, the number of nodal points per element, and the connectivity of the elements. In this example, the maximum available storage for the stiffness matrix should be 1713 if tape segments are not to be used. Three tape segments are required to accomplish the LDU decomposition.

Fig. 7 shows the complete output from WAVE for tape segment 2. The total number of elements in this segment is 5. This segment has 24 fully

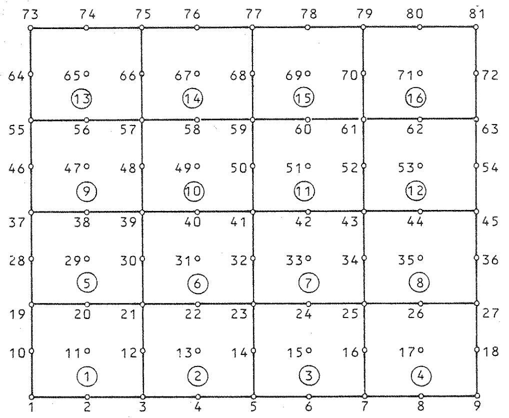


Fig. 5 Finite element mesh.

```
      NUMEL       NUMNP       NNPE
         16          81          9
    ELEMENT            NP ARRAY
       1       1       3       21       19        2       12       20       10       11
       2       3       5       23       21        4       14       22       12       13
       3       5       7       25       23        6       16       24       14       15
       4       7       9       27       25        8       18       26       16       17
       5      19      21       39       37       20       30       38       28       29
       6      21      23       41       39       22       32       40       30       31
       7      23      25       43       41       24       34       42       32       33
       8      25      27       45       43       26       36       44       34       35
       9      37      39       57       55       38       48       56       46       47
      10      39      41       59       57       40       50       58       48       49
      11      41      43       61       59       42       52       60       50       51
      12      43      45       63       61       44       54       62       52       53
      13      55      57       75       73       56       66       74       64       65
      14      57      59       77       75       58       68       76       66       67
      15      59      61       79       77       60       70       78       68       69
      16      61      63       81       79       62       72       80       70       71
     MAXVOL      IRDER      NONSYM
        800          0           1
   IORDER ARRAY
    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
   THE FOLLOWING RESULTS APPLY IF TAPES ARE NOT TO BE USED
   LISTC-ARRAY, THE ORDER IN WHICH THE NODAL POINTS WILL APPEAR IN THE K-MATRIX
     1     2    10    11     3     4    12    13     5     6    14    15     7     9     8
    18    16    17    19    20    28    29    21    22    30    31    23    24    32    33
    25    27    26    36    34    35    37    38    46    47    39    40    48    49    41
    42    50    51    43    45    44    54    52    53    55    73    56    74    64    65
    57    75    58    76    66    67    59    77    60    78    68    69    61    63    81
    79    62    72    80    70    71
   IDIAG FOR SKYLINE STORAGE IS
     1     4     9    16    25    28    41    48    57    60    73    80    89    92    97
   104   121   132   169   208   213   220   265   304   317   324   369   408   421   428
   473   512   553   560   577   588   625   664   669   676   721   760   773   780   825
   864   877   884   929   968  1009  1016  1033  1044  1081  1084  1125  1132  1141  1152
  1201  1216  1261  1268  1289  1300  1353  1368  1417  1424  1445  1456  1513  1564  1569
  1588  1645  1656  1669  1696  1713
   NPRC-ARRAY
     1     2     5     6     9    10    13    15    14     3     4     7     8    11    12
    17    18    16    19    20    23    24    27    28    31    33    32    21    22    25
    26    29    30    35    36    34    37    38    41    42    45    46    49    51    50
    39    40    43    44    47    48    53    54    52    55    57    61    63    67    69
    73    77    74    59    60    65    66    71    72    80    81    78    56    58    62
    64    68    70    76    79    75
   TAPE STORAGE IS NECESSARY
   TO AVOID USE OF TAPES MAXVOL MUST EQUAL   1713
```
Fig. 6 WAVE output.

```
    ISEG      IELEX        NEQ      ICOMP      MOVEX     ISLIDE      LCOMP
       2          5         33         24         41          0         24
 IELE-ARRAY
  8    9    10    11    12
 NP-ARRAY
 ELEM      NP1       NP2       NP3       NP4       NP5       NP6       NPP
    8       25        27        45        43        26        36
   44       34        35         9        37        39        57
   55       38        48        56        46        47        10
   39       41        59        57        40        50        58
   48       49        11        41        43        61        59
   42       52        60        50        51        12        43
   45       63        61        44        54        62        52
   53
 IEQS-ARRAY
 25    27    26    36    34    35    37    38    46    47    39    40    48    49    41
 42    50    51    43    45    44    54    52    53    55    56    57    58    59    60
 61    63    62
 MOVE-ARRAY
  0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
  0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
  1     2     3     5     7     8    11    12    15    16    19
 IDIAG1-ARRAY
  8    11    16    23    32    35    48    55    64    67    80    87    96    99   104
111   128   139   176   215   220   227   272   311   324   331   376   415   428   435
480   519   560   575   608   643   680   711   744   771   800
 IDIAG2-ARRAY
120   123   128   135   144   155   168   183   188   195   216   239   252   259   288
319   332   339   376   415   456   463   480   491   528   567   608   643   680   711
744   771   800
 NPR-ARRAY
 -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1
 -1    -1    -1    -1    -1    -1    -1    -1    -1     1     3     2    -1    -1    -1
 -1    -1    -1     5     6     4     7     8    11    12    15    16    19    21    20
  9    10    13    14    17    18    23    24    22    25    26    27    28    29    30
 31    33    32     0     0     0     0     0     0     0     0     0     0     0     0
  0     0     0     0     0     0
LIST-COMP-ARRAY
 25    27    26    36    34    35    37    38    46    47    39    40    48    49    41
 42    50    51    43    45    44    54    52    53
```

Fig. 7 WAVE output for tape 2.

assembled equations out of 33 equations. If we have more than one variable per node, ICOMP, which is the array designating fully assembled equations, will be larger than LCOMP, which is the array designating fully assembled nodal points. IEQS-ARRAY is the array of all global equation numbers for the equations on the current tape segment. MOVE-ARRAY is the array designating how previous stiffness matrix should be rearranged for storage arrangement of the current segment. Referring to the MOVE-ARRAY, previously incompleted terms will be rearranged from the position specified by IDIAG 1 to the position specified by IDIAG 2. Minus 1 and 0 appear in NPR-ARRAY. Minus implies that the node I has fully assembled and has been eliminated in the previous segments. Zero implies that the node I has not yet assembled on the current segment and will appear in the subsequent segments.

Fig. 8 shows the profiles of three tape segments.

The program POISSON, which is a finite element program for the solution of Poisson's equation, is used to verify the previously produced frontal-skyline algorithm. In order to obtain a nonsymmetric type of stiffness matrix,

$$\partial/\partial x(\Phi\cdot\partial\Phi/\partial x)+\partial/\partial y(\Phi\cdot\partial\Phi/\partial y)=0$$

with the boundary conditions of $\Phi\,(0,y)=0$, $\Phi(1,y)=1$ and $\partial\Phi(x,0)/\partial y=\partial\Phi(x,1)/\partial y=0$ is solved using standard Galerkin's finite element method. The exact solution of this problem is $\Phi=\sqrt{x}$. Seven times of iterations were required to obtain the results when Newton-Raphson method was incorporated.

The CPU time required to solve this example problem with the presented method is 8.307 seconds. The time may be shortened if a direct access file is used. On the other hand, when in-core band solver is used for the same example problem (band width is 41), the CPU time is 8.636 seconds. The subroutine WAVE, which determines the algorithm of frontal-skyline method for this problem, requires the CPU time of 1.084 seconds. The computer machine used here is UNIVAC 1100/80 B.

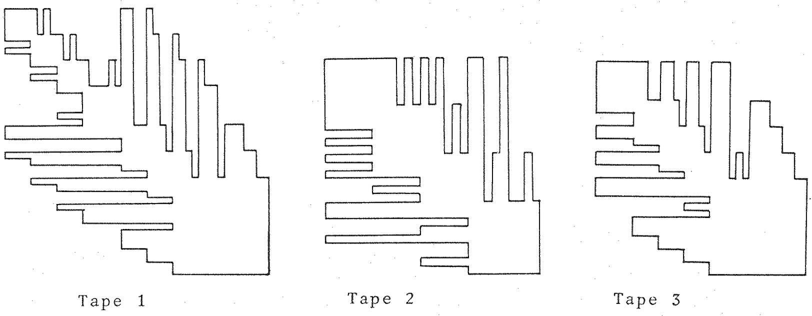


Fig. 8 Skyline storage for all three segments.

7. CONCLUSIONS

The compact skyline storage scheme for unsymmetric stiffness matrix has been incorporated into the frontal method. The solution process is extremely efficient under a variety of large finite element equations which can not be solved within an in-core storage area. In the problems involving hybrid elements, e. g. incompressible viscous flows, skyline storage is particulary efficient. The method is also valuable for the problems where the front might vary greatly in its width from one position to the next. Finally, because the subroutine WAVE is a separate program from a main routine, a variety of programs can easily be incorporated with this method.

8. ACKNOWLEDGEMENT

The author is greatly indebted to professor E. G. Thompson, Colorado State University, for his valuable suggestions during the course of this study.

References

1) Felippa, C. A. : Solutions of Linear Equations with Skyline-Stored Symmetric Matrix, Comput. Structures, Vol. 5, pp. 13~29, 1975.
2) Mendelssohn, E. and Baruch, M. : Solution of Linear Equations with a Symmetrically Skyline-Stored Nonsymmetric Matrix, Comput. Structures, Vol. 18, No. 2, pp. 215~246, 1984.
3) Jennings, A. : Matrix Computation for Engineers and Scientists, Wiley, London, 1977.
4) Thompson, E. G. and Shimazaki, Y. : A Frontal Procedure Using Skyling Storage, Int. J. Num. Meth. Engng., Vol. 15, pp. 889~910, 1980.
5) Hood, P. : Frontal Solution Program for Unsymmetric Matrices, Int. J. Num. Meth. Engng., Vol. 10, pp. 379~399, 1976.
6) Irons, B. M. : A Frontal Solution Program for Finite Element Analysis, Int. J. Num. Meth. Engng.,, Vol. 2, pp. 25-32, 1970.

(Received August 22 1984)