UNIVERSITY
OF
QUEENSLAND

Department of Civil Engineering

RESEARCH REPORT SERIES

A Second Generation Frontal Solution Program

G. BEER

Research Report No. CE12
May, 1980.

CIVIL ENGINEERING RESEARCH REPORTS

This report is one of a continuing series of Research Reports published by the Department of Civil Engineering at the University of Queensland. This Department also publishes a continuing series of Bulletins. Lists of recently published titles in both of these series are provided inside the back cover of this report. Requests for copies of any of these documents should be addressed to the Departmental Secretary.

The interpretations and opinions expressed herein are solely those of the author(s). Considerable care has been taken to ensure the accuracy of the material presented. Nevertheless, responsibility for the use of this material rests with the user.

Department of Civil Engineering,
University of Queensland,
St Lucia, Q 4067, Australia,
[Tel:(07) 377-3342, Telex:UNIVQLD AA40315]

A SECOND GENERATION FRONTAL SOLUTION PROGRAM

by

G. Beer, Dip Ing *Austria*, MSc *Lehigh*, PhD, MIE Aust.

Senior Research Assistant

RESEARCH REPORT NO. CE 12

Department of Civil Engineering

University of Queensland

May, 1980

Synopsis

A computer program for the assembly and solution of symmetric positive definite equations as met in the Finite Element analysis based on the Frontal Solution algorithm by Irons is presented.

The program features improved direct access blocked I/O and the use of Front partitioning which makes the problem size which can be solved practically independent of the size of the computer memory. In addition the use of fast vector processors is considered which should improve CPU times considerably.

CONTENTS

1. INTRODUCTION

The Frontal Solution technique is based on the Gaussian Elimination method and was first published in 1970 by Irons (1).

Various authors (2,3,4,5) have since pointed out various advantages of this technique. The main advantage over band or skyline (6) solvers seems to be a simplification in the data preparation as the nubmering of nodes is not restricted to minimise a band width. Also, since the node numbers are treated as "Nicknames", design changes (i.e. adding or removing elements) may be made without having to renumber the nodes. Solution time for the Frontal solver is now sensitive to the numbering of the Elements but the sequence is a natural one. The famous example of a ring structure is often mentioned in this context (1).

The main difference between the Frontal technique and a conventional band solution lies in the manner in which the structure stiffness coefficients are stored and in the order in which the equations are eliminated.

Consider, for example, a patch of 4 node/8 D.O.F. elements in Fig. 1 in which the degrees of freedom are numbered from 1 to 18.

The stiffness coefficients of Element I are stored in the order of appearance (local node numbering) and in the manner shown in Fig. 1a. Each variable has a "Destination" which determines the position of its coefficients in the Front matrix. An asterisk, *, marks the equations which are already fully summed. These variable(s) can now be eliminated before the next element is assembled by treating the other equations with:

$$c'_{i,j} = c_{i,j} - c^*_{n,j}\,\frac{c^*_{n,i}}{c^*_{n,n}} \qquad (1)$$

Where n is the destination of the variable to be eliminated and the * denotes the coefficients which are fully summed. The reader should note that it does not matter that the coefficients $c_{i,j}$ are not in their final form since only the order in which the coefficients are added is changed.

After elimination, variables 1 and 10 cease to be "active" and the corresponding equations are transferred into buffer storage ready for output on disk. The storage locations of these equations in the Front matrix are cleared (Fig. 1b). On assembly of element II this space is re-used by Equation 3 and 6 (Fig. 1c). The Front matrix thus contains only the coefficients of "active" variables.

With the principle involved explained the reader may complete the example making the following observations:

(1) Assembly and elimination order is governed by the order in which the variable coefficients are entered as one Element is assembled after the other. A variable is eliminated as soon as the coefficients are fully summed (i.e. on its last appearance). The position of the coefficients in the Front matrix is governed by the empty spaces available.

(2) The storage requirement for storing the "active" coefficients is determined by the largest address used. Because of symmetry only one half of the Front matrix has to be stored and the storage requirement may be computed from

$$\ell = m(m + 1)/2 \tag{2}$$

Here m is the largest "Destination" of a variable (Front width).

In the original code Irons assumes that the Front matrix resides in central storage (Comment on the program listing: "if not, buy larger computer").

It can be seen that the storage requirement increases with the square of the Front width. This puts a severe limitation on the size of problems which can be solved on a special computer.

The purpose of this paper is to present a program where this limitation has been over-come making it possible to solve large 3-D problems on a mini computer. In addition, the transfer of data to and from disk is improved by using a blocked direct access I/O mode.

The inner-most DO-loop of the Gauss elimination is written suitable for fast vector processors which have appeared on the market in recent years.

The resulting program should not only be an improvement in solution capability but also in performance.

2. STORAGE AND BUFFERING DURING ELIMINATION

Similar to the original code by Irons, a working vector ELPA is used. ELPA is divided into 3 main areas which contain: (1) Element stiffness matrix of the Element to be assembled; (2) Front matrix or a partition of the front matrix; (3) Buffer for the equations which have ceased to be "active".

Allocation of the space for these areas is completely flexible and will depend on the type of problem solved. Whereas the space of first area is fixed by the size of the element stiffness matrix, the space allocation for areas 2 and 3 must be adjusted to give optimal solution times within the core limitations of the computer used. This will be discussed in detail later.

The number of equation coefficients in the Front matrix for a particular Front width is computed from Equation 2 and if it is greater than the space available, the partitioning algorithm has to be activated.

The number of Equations $\Delta\ell_N$ which fit into a particular N can be computed from the inequality

$$\Delta\ell_N\ (\Delta\ell_N + 1)\ /2 + (k_N - 1)\ {}^*\Delta\ell_N < n_p \tag{3}$$

Here k_N is the destination of the first equation in partition N and n_p is the size of the storage space for the Front matrix. Solution of Equation 3 gives

$$\Delta\ell_N = \text{INT}\left[-\frac{2k_N - 1}{2} + \sqrt{\left(\frac{2k_N - 1}{2}\right)^2 + 2n_p}\right] \tag{4}$$

Where INT means the truncation of the result. This allows us to determine the limits of each partition i.e. partition N can accommodate Equation k_N to k_{N+1}.

For a coefficient k,i (k < i) residing in partition N, where

$$k_N < k < k_{N+1} \tag{5}$$

the address, ℓ, is computed from (see Fig. 2),

$$\ell = \ell_o + (k - k_N)\ (k_N - 1) + (k - k_N)\ (k - k_N + 1)/2 + i \tag{6}$$

Here ℓ_o specifies the start address of the space reserved for the Front matrix.

3. PROGRAMMING STRATEGY

As long as the Front matrix is small and fits into core the program strategy is simple and follows three basic steps for each element:

(1) Read element stiffness matrix
(2) Assemble its coefficients into Front matrix
(3) Eliminate variables which are ready

The elimination essentially consists of two steps. First, the equation coefficients of the variable which is to be eliminated (n) are moved to the equation buffer. When the buffer is full its contents are written on disk and the pointer reset to the beginning of the buffer before the coefficients are moved (It should be noted at this stage that all the coefficients to the right of the minus sign in Equation 1 are now in the buffer). Then all the equations in the Front matrix are modified by Equation 1.

When the current Front matrix becomes large and no longer fits in the allocated space, partitioning is invoked automatically. The program strategy becomes more complex and follows the following basic steps:

(1) Read Element stiffness matrix.
(2) Swap Partition 1 into core and assemble all coefficients which are resident in this partition.
(3) Eliminate variables which are ready in this partition (if any).
(4) Swap Partition 2 into core and assemble further coefficients.
(5) Modify all coefficients for the variables eliminated previsouly (if any) using the coefficients $c^*_{n\ell}$ in the Equation buffer. This is referred to as elimination of 'old variables' in the listing.
(6) Eliminate variables which are ready in this partition ('new variables').

Steps 4, 5 and 6 are repeated for all subsequent partitions to the last one. Note that at step 3 not all the coefficients $c^*_{n\ell}$ of the equation n are available and the equation in the buffer is still incomplete. Thus, additional coefficients have to be transferred in step 5.

Our work is not completed yet since some lower partitions have not been modified due to elimination of variables in higher partitions. So we have to retrack and modify the Equations which have not yet been modified using the coefficients $c^*_{n\ell}$ in the equation buffer.

4. I/O OPERATIONS DURING ELIMINATION

When the Front matrix is small and fits into the allocated space, the I/O operations are simple:

(1) Read the assembly information for each element (The 'destination vector' is coded to indicate when each variable is ready for elimination).
(2) Read Element stiffness matrix.
(3) Whenever the equation buffer is full, i.e. no further equation fits, write its contents onto disk.

In the present program, the standard Fortran READ is used for operations 1 and 2. For large amounts of data the speed of the I/O operation depends greatly on the transfer mode. It has been found by the author that on a Data General Eclipse minicomputer, the data transfer is 10 times faster when the machine dependent routines RDBLK nad WRBLK are used. The prerequisite for using these routines is that the number of coefficients to be transferred is divisible by the physical block size on disk (128 real numbers in this case). Since operation 3 may involve a large number of coefficients, a blocked I/O mode is used. The equation buffer is divided into a number of blocks and the space allocated for it should be a multiple of the block size. When the buffer is full it may, however, not always fill the last block completely. To avoid empty spaces on disk the last block is not written in this case, but the coefficients are rather transferred to core to the beginning of the buffer with new coefficients moved into the subsequent spaces.

In this context it should be noted that in core transfers are typically a factor of 10^3 faster than out-of-core transfers (i.e. transfers to and from disk).

When the Front matrix becomes too large and no longer fits into core, I/O operations become more complex and frequent. The Front matrix has to be swapped in and out of core as required. Blocked I/O transfer now becomes essential and the programming critical since a program slowed down by too many I/O operations may no longer be competitive.

The aim is to reduce the I/O operations to a bare minimum even if this means that more in core operations are necessary to do this (see above statement about I/O transfer speeds).

The number of swapping operations on the partitions can be determined from the basic steps delineated in the last chapter and depends on the location of variables which are to be eliminated. In the worst case, we need (2N - 1) swaps where N is the number of partitions currently used. This number is critical for the performance of the program as partitions usually involve a large number of coefficients. Ways to optimise the number of I/O operations are discussed in the next chapter.

A further complication arises which could endanger the economy of the program. For the basic step 5, (elimination of 'old variables') the coefficients $c^*_{n\ell}$ which are thought to reside in the equation buffer are needed. But in the meantime, I/O operation 3 (transfer of equation buffer on to disk when full) may have been carried out and the required coefficients may no longer be in core. Thus we must keep track of which coefficients are in core and which are on disk. If the coefficients are no longer in core they must be swapped into core and this requires additional I/O operations, the number of which depends greatly on the size of the equation buffer. If the buffer is very large then swapping of the equation buffer may occur only rarely.

5. OPTIMISATION OF I/O OPERATIONS AND CHOICE OF FRONT PARTITION AND EQUATION BUFFER LENGTH

When selecting the size of the buffer for the Front matrix, we must aim to avoid partitioning since it is expensive. If no partitioning is involved the equation buffer may be made small to accommodate a big Front matrix, the only restriction being that, the buffer has to be at least 2 blocks long and accommodate the largest equation. On the other hand, when partitioning of the Front is unavoidable because of core restrictions or size of problem, there is a case for decreasing the partition size in favour of a large equation buffer for the reasons explained in the last chapter.

To reduce the number of I/O operations further, a number of situations where swapping is not required is examined. Swapping of partitions is not required when:

(A) There are no coefficients to be assembled into the partition, and no variables have been eliminated yet in the current element loop.

(B) There are no coefficients to be assembled and the coefficients $c^*_{n\ell}$ $(k_N < \ell > k_{N+1})$ are zero.

(C) Swapping of the equation buffer is not required when the coefficients $c^*_{n\ell}$ $(k_N < \ell > k_{N+1})$ in the Front partition are zero.

A check on conditions C and B is made by Subroutine SAVES in the program.

6. SHORT DESCRIPTION OF COMPUTER PROGRAM

The program consists of two main subroutines PREFR and SFRONT. The subroutine PREFR works out the coded destinations of the variables and writes them onto disk. This program is essentially the same as published by Irons and is included for completeness. Subroutine SFRONT performs the assembly and reduction of the structure or substructure stiffness matrix as detailed in the last chapters.

It uses the following subroutines:

MOVE ..... to move equation coefficients to buffer.
GAUSS .... to modify the coefficients in the Front matrix with Eq. (1).
ASSEMB ... to assemble stiffness coefficients into the Front matrix.
EMPDI .... to empty equation buffer on disk when full.
RESBUF ... to reset buffer pointer.
SWAPF .... to swap Front partitions in and out of core.
SAVES .... to save swaping (see last chapter).
UNCOD .... to uncode coded destinations (uses CODEST)
PALI ..... to work out partition limits (k_N, K_{N+1}).

In addition the following functions are used:

LADDR(M) .... is the local address in the current Front partition of a coefficient M, M.
LADST(I,J) .. is the local address of a coefficient i,j in the Element stiffness matrix.

Subroutines are also used to clear integer and real arrays and write error messages.

The blocked I/O operations are performed by subroutine BLKIO which has a machine dependent coding. Files are opened and channel numbers assigned by FILO which is also machine dependent.

The fast vector operations are performed by subroutine SVECT. When a computer with vector processor is used, the appropriate coding as given in the Users Manual of the machine should be inserted here. For use on machines without this capability the Standard Fortran coding may be used as shown.

The computer program is listed in Appendix B. A list of some important arrays and variables is given in Appendix A. In addition, a program is included to test and demonstrate the substructure capability of the sub-routine SFRONT in Appendix C.

7. SUBSTRUCTURING WITH THE FRONTAL SOLUTION

For very large structures, it is often desirable to divide the mesh into several smaller meshes or substructures. These are treated as large elements and the boundary stiffness matrix obtained by elimination of the 'internal' degrees offreedom.

The substructuring has the following main advantages:

(1) The process of solving the structure is a continuous one and errors may be detected at substructure level. Remedial actions need only to be taken in the particular substructure involved.

(2) Sometimes a structure consists of many subareas having a similar geometry. Thus the stiffness matrix of a particular type of substructure may be computed only once and the main structure assembled with as frequent re-use of the substructure stiffness as possible.

(3) For excavation type of problems in rock or soil mechanics the substructuring technique offers additional advantages. By defining the rock or soil mass in the full excavation as one large substructure and the material to be excavated at each stage as smaller substructures the analysis of each excavation stage just requires the assembly of substructure stiffnesses and the solution for the substructure boundary degrees of freedom.

Substructuring with the Front Solution is relatively simple. All that has to be done is to suspend the elimination of selected variables at the boundary of the substructure. The coefficients which remain in the Front matrix after the elimination of all other variables then constitute the stiffness coefficients for the super element. After suitable reordering, the stiffness matrices of all super elements can be obtained and assembled in the usual manner to solve for the complete structure.

Thus, substructuring involves the basic steps.

(1) The PREFRONT subroutine read the substructure "Nicknames" into the vector NIX. This will modify the coding of the destinations of the substructure variables in such a way as to prevent their elimination.

(2) Perform the usual assembly and elimination for all elements which make up the substructure.

(3) Remove zero rows and columns from the Front matrix and reorder to obtain the substructure stiffness matrix in condensed form.

After this has been done for all substructures, perform the assembly and elimination in the usual way but this time involving all substructures which make up the structure to be analysed.

The substructuring capability is demonstrated with a test program in Appendix II where the substructure consists of a regular assembly of 4 node/8 degrees of freedom Elements.

8. RE-SOLUTION

Once the global stiffness matrix has been reduced and stored a re-solution for as many load cases as desired can be made.

It is convenient to separate the resolution and back substitution part completely from the reduction of the left hand side in order to have as much space available as possible. Because the size of the vector needed for each load case is only MAXPA no partitioning of the Front should be required even for large problems and the basic procedures are as follows:-

(1) Read the Element right hand side (RHS) into the first part of ELPA.

(2) Assemble into the space reserved for the Front-RHS.

(3) Reduce RHS using the coefficients c^*_n on disk i.e. modify the Front RHS with

$$F^1_i = F_i - \frac{c^*_{in}}{c^*_{n,m}} F^*_n \tag{7}$$

The procedure is exactly the same as a non-partitioned reduction except that vectors are involved instead of matrices.

The results are obtained in Element form by back substitution i.e.

$$x_n = \frac{1}{c^*_{nn}} \left(\Sigma_{i=n} \; c_{n,i} - F^*_n \right) \tag{8}$$

in the same manner as by Irons.

9. FURTHER FACILITIES OF THE COMPUTER PROGRAM AND DISCARDED FACILITIES

This section deals with features which are included in the present program and facilities which have not been considered but can be implemented easily.

9.1 Treatment of Constraints

In the present program a restrained degree of freedom is treated by setting the corresponding destination to zero and thereby preventing the assembly of the corresponding equation.

This is the simplest and most economical way. Various other types of constraints, as shown by Irons (1) can be easily implemented.

9.2 Computation of the Determinant of the Structure Stiffness Matrix

This is often required for vibration and stability analysis and is incorporated by additional coding in Subroutine GAUSS. After elimination the value of $\log_{10}/K/$ is stored in the variable DET. In addition the frequency of the occurrence of a negative diagonal element is determined and stored in NEG. If NEG is odd the sign of the determinant is positive otherwise negative. The variables DET and NEG are in Common block/ EIGEN/.

9.3 Check on Singularity and Indefiniteness

A check on singularity and indefiniteness is made during elimination. If the diagonal coefficient is less than or equal to zero an Error message is produced. Because of machine accuracy, the diagonal coefficient will not be exactly zero even for a singluar matrix. More appropriate checks have been suggested (6), that is,

(1) Singularity

$$d_j < t_j \tag{9}$$

(2) Indefiniteness

$$d_j < - t_j \tag{10}$$

where d_j is the j-th diagonal element at the j-th elimination stage and

$$t_j = 8\varepsilon\ r_j$$

where ε is the smallest positive floating point number for which $(1 + \varepsilon) > 1$ on the computer used and r_j is the norm of the j-th row of K. This can be easily implemented in Subroutine GAUSS if the machine accuracy ε is known.

9.4 Check on Accuracy of Solution

In the original code by Irons a simple roundoff criterion was included. The author has found this criterion not entirely satisfactory because it is not sensitive to right hand sides and was found to register only if the difference in stiffness is too great between elements.

A better *a priori* estimate of the matrix condition is the Eucledian condition number (6). But this also involves additional unproductive computation and may be expenside.

The author favours the *a posterori* estimate by one step of iterative refinement of the solution because it is a more productive method giving not only an estimate of the accuracy but also an improved solution. It only involves a re-solution and matrix multiplication. The iterative refinement may be made only for one load case and not repeated for the other load cases if the condition number is satisfactory.

First, the load case is solved with the re-solution facility to give x_i^o, the unrefined result. Then the residual forces are worked out:

$$R_i^o = F_i - K_{ij}\, x_j^o \tag{11}$$

A second resolution with R_i^o as new right hand side will give the error on x_j^o, Δx_j^o.

The expression

$$||\Delta x_j^o|| / ||x_j^o|| \tag{12}$$

provides an estimate on the accuracy of the solution x_j^o.

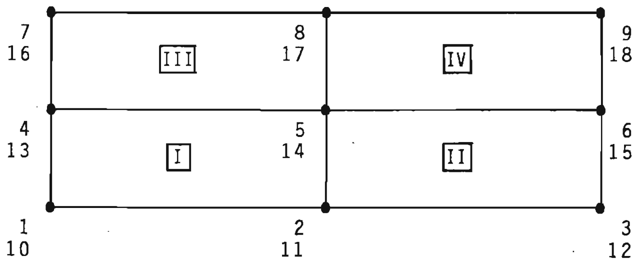
10. CONCLUSIONS

A computer program for the assembly and solution of a symmetric positive definite set of equations has been presented. The program is based on the Frontal Solution technique by Irons but uses frontal partitioning to make the problem size which can be solved practically independent of the memory size of the computer used.

In addition, a great deal of effort has been made to optimise the I/O operations during partitioned elimination. Fast vector or pipeline processing has also been considered in the coding.

The resulting program is an improvement, not only in capability but also in performance. The program should be useful not only in mini-computer applications but also for large computers, because a reduction or optimisation of the band width is not required in the Frontal solution.

The solution time and storage requirement is influenced only by the numbering of the Elements which is a natural one.



+ .... non-zero entry
P .... diagonal
0 .... zero coeff.

EQUATION	DESTINATION	1	2	3	4	5	6	7	8
1	1*	P							
2	2	+	P						
5	3	+	+	P					
4	4	+	+	+	P				
10	5*	+	+	+	+	P			
11	6	+	+	+	+	+	P		
14	7	+	+	+	+	+	+	P	
13	8	+	+	+	+	+	+	+	P

(a)

EQUATION	DESTINATION	1	2	3	4	5	6	7	8
□	1	0							
2	2	0	P						
5	3	0	+	P					
4	4	0	+	+	P				
□	5	0	0	0	0	0			
11	6	0	+	+	+	0	P		
14	7	0	+	+	+	0	+	P	
13	8	0	+	+	+	0	+	+	P

(b)

EQUATION	DESTINATION	1	2	3	4	5	6	7	8	9	10
$\boxed{3}$	1	P									
2	2	+	P								
5	3	+	+	P							
4	4	0	+	+	P						
$\boxed{6}$	5	+	+	+	0	P					
11	6	+	+	+	+	+	P				
14	7	+	+	+	+	+	+	P			
13	8	0	+	+	+	0	+	+	P		
11	9	+	+	+	0	+	+	+	0	P	
12	10	+	+	+	0	+	+	+	0	+	P

(c)

FIGURE 1 : Storage of stiffness coefficients during Frontal solution

```
STORAGE IN FRONT:

                 I
K(N)       [+++P           ]
           [+++++P         ]
K          [++*++P         ]
K(N+1)     [+++++++P       ]

STORAGE IN ELPA:

 +++P+++++P++*++P|++++++++P
LO              L
```

FIGURE 2 : Storage of coefficients in long vector ELPA

APPENDIX A - LIST OF IMPORTANT ARRAYS AND VARIABLES

ELPA	main working space
HBWR	an indicator if a partition has been written at least once
INDABL	indicator on the space availability in the Front matrix also indicates in which partition space is available (coding: "+" occupied "-" ve free)
LCDEST	list of coded element destinations
LUDEST	list of uncoded element destinations (not entirely true since the destinations are still coded with a "-" ve sign for variables ready for elimination).
LPAL	list of partition limits
LISTEQ	list of start addresses of equations in the buffer or on disk. Lists address of pivot, block number and relative address in block for each element loop
KURPA	current equation length
LBLK	length of physical block on disk (real words)
NBLKA, NBLKE	first and last block currently in the equation buffer
NPA, NPAC	number of partition to be swapped into core and number of partition currently in core
NELZ, LFRBUF, LEQBUF	length of buffers for element stiffness matrix, Front matrix and equation buffer (real words) MUST BE DIVISIBLE BY THE BLOCK LENGTH.

APPENDIX B - LISTING OF THE COMPUTER PROGRAM

```
      SUBROUTINE PREFR
C----------------------------------------------------------
C   PREFRONT ROUTINE
C----------------------------------------------------------
      COMMON NIX(10000),MAXPA,NELEMZ,LDEST(62)
C------------------------------------------------------
C   VARIABLES IN BLANK COMMON:
C       NIX    ...   WORKING SPACE
C       MAXPA    ...   MAX. FRONT WIDTH
C       NELEMZ   ...   NUMBER OF ELEMENTS
C       LDEST    ...   ELEMENT DESTINATIONS
C-----------------------------------------------------
      COMMON /FILES/ NF6,NDIM6,NF7,NDIM7,NF8,NDIM8,NF9,NDIM9
      DIMENSION LVABL(60),MVABL(500),LCDEST(60)
      EQUIVALENCE (LPREQ,LDEST(1))
      EQUIVALENCE (KUREL,LDEST(2))
      EQUIVALENCE (LCDEST(1),LDEST(3))
      NIXEND= 2000
      CALL ICLAR(MVABL,500)
      MAXPA=1
      NIZZ= 0
C--------------------------------------------------------
C   PUT ALL ELEMENT NICKNAMES IN LONG VECTOR NIX
C------------------------------------------------------
      DO 10 NELEM= 1,NELEMZ
      CALL GETELN(NELEM,KUREL,LVABL)
      DO 8 I=1,KUREL
      NIC= LVABL(I)
      NIZZ= NIZZ +1
      NIX(NIZZ)= -NIC
    8 CONTINUE
      NIX(NIXEND+1-NELEM)= NIZZ
   10 CONTINUE
C--------------------------------------------------------
C   PUT SUBSTRUCTURE NICKNAMES(IF ANY) AT THE END OF NIX
C---------------------------------------------------------
      CALL GETSUN(KUREL,LVABL)
      IF(KUREL .EQ. 0) GOTO 11
      NIZS= NIZZ
      DO 9 I=1,KUREL
      NIC= LVABL(I)
      NIZZ= NIZZ+1
      NIX(NIZZ)= -NIC
    9 CONTINUE
   11 CONTINUE
      KURELS= KUREL
      LCUREQ= 0
      NVABZ= 0
C-----------------------------------------------------
C   FIND DESTINATIONS
C-----------------------------------------------------
      N1= 1
      DO 26 NELEM=1,NELEMZ
      LPREQ= LCUREQ
      LCUREQ= NVABZ
      NIXE= NIXEND+1-NELEM
      NZ= NIX(NIXE)
      KUREL= NZ - N1 + 1
      DO 22 NEW= N1,NZ
      NEWA= NEW
```

```
      NIC= NIX(NEW)
      LDES= NIC
      IF(NIC .GT. 0) GOTO 20
      LDES= 1
   14 CONTINUE
      IF(MVABL(LDES) .EQ. 0) GOTO 16
      LDES= LDES + 1
      IF(LDES .LE. MAXPA) GOTO 14
      MAXPA= LDES
   16 CONTINUE
      MVABL(LDES)= 1
C-----------------------------------------------
C   RECORD FIRST,LAST AND INTERM. APPEARANCES
C-----------------------------------------------
      KOUNT= 1000
      DO 18 LAS= NEW,NIZZ
      IF(NIX(LAS) .NE. NIC) GOTO 18
      NIX(LAS)= LDES
      KOUNT= KOUNT + 1000
      LAST= LAS
   18 CONTINUE
      NIX(LAST)= LDES + 1000
      LDES= LDES + KOUNT
      NIX(NEW)= LDES
   20 CONTINUE
      NZ1= NEW-N1+1
      LCDEST(NZ1)= LDES
   22 CONTINUE
      N1= NEWA+ 1
C-------------------------------------------------
C   UPDATE MVABL,COUNT ELIMINATED VARIABLES
C   AND WRITE DESTINATIONS ON DISK
C-------------------------------------------------
      DO 24 KL=1,KUREL
      CALL CODEST(KL,NSTRES,LDES,LCDEST,KUREL)
      IF(NSTRES .NE. 0 .AND. NSTRES .NE. 1) GOTO 24
      MVABL(LDES)= 0
      NVABZ= NVABZ+1
   24 CONTINUE
      WRITE(NF9'NELEM) (LDEST(I),I=1,NDIM9)
   26 CONTINUE
C-------------------------------------------------------
C   WRITE SUPERELEMENT DESTINATIONS ON DISK
C-------------------------------------------------------
      NIZZ= NIZS
      KUREL=KURELS
      IF(KUREL .EQ. 0) GOTO 23
      DO 25 KL=1,KUREL
      NIZZ= NIZZ + 1
      LCDEST(KL)= NIX(NIZZ) - 1000
   25 CONTINUE
   23 CONTINUE
      NELEM= NELEMZ+1
      WRITE(NF9'NELEM) (LDEST(I),I=1,NDIM9)
      RETURN
      END
```

```
      SUBROUTINE SFRONT
C-----------------------------------------------
C
C   S U P E R F R O N T
C
C   A SECOND GENERATION FRONTAL SOLUTION PROGRAM
C
C   G.BEER   UNIVERSITY OF QUEENSLAND   1979
C
C---------------------------------------------
      COMMON ELPA(5000),MAXPA,NELEMZ,LDEST(62)
C-----------------------------------------------------
C     VARIABLES IN BLANK COMMON :
C     ELPA   ...  WORKING SPACE
C     MAXPA   ...  MAXIMUM FRONT WIDTH (FROM PREFRONT)
C     NELEMZ  ...  NUMBER OF ACTIVE ELEMENTS
C     LDEST   ...  ELEMENT DESTINATIONS
C-----------------------------------------------------
      EQUIVALENCE (KUREL,LDEST(2))
      EQUIVALENCE (LCDEST(1),LDEST(3))
      COMMON /EIGEN/ DET,NEG
      COMMON /PARTL/ NST,NEND
      COMMON /PARA/ L0,L1
      COMMON /BLOKL/ LBLK
      COMMON /FILES/ NF6,NDIM6,NF7,NDIM7,NF8,NDIM8,NF9,NDIM9
      COMMON /IOCONV/ IREAD,IWRIT
      COMMON /EQL/ KURPA
      COMMON /ENDQN/ LASTBL
      COMMON /BUFSZE/ NELZ,LFRBUF,LEQBUF,LFRBBL,LEQBBL
      COMMON /INCORE/ NBLKA,NBLKE
C---------------------------------------------------
C   CURRENT COMPILATION IS FOR:
C
C   ELEMENT SIZE= 60 D.O.F.
C   MAXIMUM FRONT WIDTH= 500
C   MAXIMUM NUMBER OF PARTITIONS= 50
C
C   DIMENSION FOR LISTEQ= 50 + 60*3 = 230
C-----------------------------------------------
      DIMENSION LUDEST(60)
      DIMENSION LCDEST(60)
      DIMENSION INDABL(500)
      DIMENSION HBWR(50)
      DIMENSION LPAL(51)
      DIMENSION LISTEQ(230)
C   MAX NUMBER OF PARTITIONS THIS COMPILATION:
      MAXPAR= 50
C   SIZE OF ELPA:
      LSIZE= 5000
      IF(NELZ+LFRBUF+LEQBUF .GT. LSIZE) CALL ERROR(0.,LSIZE,3)
      LFRBBL= LFRBUF/LBLK
      LEQBBL= LEQBUF/LBLK
C----------------------------------------------
C     ELPA ADRESSES
C----------------------------------------------
C   START OF FRONT MATRIX
      L0= NELZ
C   START OF EQUATION BUFFER
      L1= NELZ + LFRBUF
C   CLEAR ARRAYS AND WORK OUT PARTITION LIMITS
```

```
        CALL CLEAR(ELPA,1,LSIZE)
        CALL PALI(MAXPAR,MAXPA,INDABL,LPAL,NOPAR)
        PRINT 3002,MAXPA,NOPAR
 3002   FORMAT(//' MAXIMUM FRONT WIDTH=',I5/
     1          ' MAX. NO. OF PARTITIONS=',I5//)
        CALL ICLAR(HBWR,NDPAR)
        DET= 0.
        NEG= 0
        NBLK= 0
        NBLKA= 1
        NBLKE= LEQBBL
        IEQ= L1
        NPAC= 1
        DO 1 NELEM= 1,NELEMZ
        TYPE  3003,NELEM
 3003   FORMAT(I5)
C READ CODED ELEMENT DESTINATIONS
        READ(NF9'NELEM) (LDEST(I),I=1,NDIM9)
C UNCODE AND UPDATE SPACE INDICATOR
        CALL UNCOD(LCDEST,KUREL,LUDEST,INDABL,MAXPA,NDPAR)
C READ ELEMENT STIFFNESS
        READ(NF6'1) (ELPA(I),I=1,NDIM6)
        NVAR= 0
        III= 1
        DO 2 NPA=1,NDPAR
C       FIRST EQUATION IN PARTITION NPA
        NST= LPAL(NPA) + 1
C       LAST EQUATION IN PARTITION
        NEND= LPAL(NPA + 1)
        IF(NEND .GT. KURPA) NEND= KURPA
        NVA= 0
        LISTEQ(III)= 0
        I= III
C-----------------------------------------------
C   ASSEMBLY
C-----------------------------------------------
        DO 4 NV=1,KUREL
        IRDY= 0
        LDES= LUDEST(NV)
        IF(LDES) 21,4,20
   21   CONTINUE
        LDES= -LDES
        IRDY= 1
   20   CONTINUE
        LPA= INDABL(LDES)
        IF(LPA .NE. NPA) GOTO 4
        CALL SWAPF(NPA,NPAC,HBWR)
        CALL ASSEMB(LDES,LUDEST,NV,KUREL)
        IF(IRDY .EQ. 0) GOTO 4
        NVA= NVA + 1
        NVAR= NVAR + 1
        LISTEQ(III)= NVA
        I= I + 1
        LISTEQ(I)= LDES
        I= I+2
    4   CONTINUE
C-------------------------------------------------------------------
C   ELIMINATION OF OLD VARIABLES (THOSE ELIMINATED IN PREVIOUS PARTI
C--------------------------------------------------------------TIONS)
        IF(NPA .EQ. 1) GOTO 5
        IF(NVAR .EQ. 0) GOTO 5
```

```
      ICYCL= 1
      NEW= 0
      I= 0
      NPAM= NPA-1
      DO 6 NP=1,NPAM
      I= I + 1
      NVA= LISTEQ(I)
      IF(NVA .EQ. 0) GOTO 6
      CALL SWAPF(NPA,NPAC,HBWR)
      DO 7 N=1,NVA
      I= I+1
      LDES= LISTEQ(I)
      I= I+1
      NNBLK= LISTEQ(I)
      I= I+1
      LIEQ= LISTEQ(I)
      IF(LASDES .EQ. LDES) GOTO 77
      CALL SAVES(LDES,IEQ,ICYCL,JES)
      IF(JES .EQ. 1) GOTO 7
   77 CONTINUE
      CALL RESBUF(IEQ,NBLK,NNBLK,LIEQ,ICYCL)
      CALL MOVE(NEW,LDES,IEQ)
      CALL GAUSS(NEW,LDES,IEQ)
    7 CONTINUE
    6 CONTINUE
    5 CONTINUE
C-------------------------------------------
C   ELIMINATION OF NEW VARIABLES (THOSE TO BE ELIMINATED IN CURRENT
C-------------------------------------------                PARTITION)
      NEW= 1
      NVA= LISTEQ(III)
      I= III
      IF(NVA .EQ. 0) GOTO 44
      DO 45 N=1,NVA
      I= I + 1
      LDES= LISTEQ(I)
      LASDES= LDES
      LEQ= IEQ-L1
      IBLK= LEQ/LBLK + 1
      NNBLK= NBLK + IBLK
      LIEQ= LEQ - (IBLK-1)*LBLK
      I= I + 1
      LISTEQ(I)= NNBLK
      I= I + 1
      LISTEQ(I)= LIEQ
      CALL EMPDI(IEQ,NBLK)
      CALL MOVE(NEW,LDES,IEQ)
      CALL GAUSS(NEW,LDES,IEQ)
      INDABL(LDES)= -INDABL(LDES)
   45 CONTINUE
C   ADDRESS OF LAST COEFFICIENT IN EQUATION BUFFER
      LEQ= IEQ- L1
      IBLK= (LEQ-1)/LBLK + 1
      LASTBL= NBLK + IBLK
      NDADD= LEQ - (IBLK-1)*LBLK
  44  CONTINUE
      III= I+1
    2 CONTINUE
      LASTP= NDPAR-1
C-----------------------------------------------------------------
C       NOW RETRACK AND MODIFY EQUATIONS IN LOWER
```

```
C       PARTITIONS NOT YET MODIFIED
C-----------------------------------------------------------------
        IF(LASTP .EQ. 0) GOTO 8
        ICYCL= 2
        NEW= 0
        DO 9 NPA=1,LASTP
        NST= LPAL(NPA) + 1
        NEND= LPAL(NPA + 1)
        IF(NEND .GT. KURPA) NEND=KURPA
        I= LISTEQ(1)*3 + 1
        DO 10 NP=2,NDPAR
        I= I + 1
        NVA= LISTEQ(I)
        IF(NP .GT. NPA) GOTO 99
        I= I + NVA*3
        GOTO 10
   99   CONTINUE
        IF(NVA .EQ. 0) GOTO 10
        DO 11 N=1,NVA
        I= I + 1
        LDES= LISTEQ(I)
        I= I + 1
        NNBLK= LISTEQ(I)
        I= I + 1
        LIEQ= LISTEQ(I)
        CALL RESBUF(IEQ,NBLK,NNBLK,LIEQ,ICYCL)
        CALL SAVES(LDES,IEQ,ICYCL,JES)
        IF(JES .EQ. 1) GOTO 11
        CALL SWAPF(NPA,NPAC,HBWR)
        CALL GAUSS(NEW,LDES,IEQ)
   11   CONTINUE
   10   CONTINUE
    9   CONTINUE
    8   CONTINUE
    1   CONTINUE
C-----------------------------------------------------------
C   WRITE CONTENTS OF EQUATION BUFFER IF NECESSARY
C-----------------------------------------------------------
        IF(NBLK .GE. LASTBL) RETURN
        NBLOKS= LASTBL-NBLK
        CALL BLKIO(IWRIT,NF7,NBLK+1,NBLOKS,ELPA,L1+1)
C-----------------------------------------------------------
C   CONDENSE AND REORDER SUBSTRUCTURE STIFFNESS MATRIX IF REQUIRED
C-----------------------------------------------------------
        NELEM= NELEMZ + 1
        READ(NF9'NELEM) (LDEST(I),I=1,NDIM9)
        IF(KUREL .EQ. 0) RETURN
        DO 200 NPA= 1,NOPAR
        CALL SWAPF(NPA,NPAC,HBWR)
        NPAC= 0
        NST= LPAL(NPA) + 1
        NEND= LPAL(NPA+1)
        IF(NEND .GT.KURPA) NEND= KURPA
        L=0
        DO 201 I=1,KUREL
        IDES= LCDEST(I)
        DO 202 K=1,I
        L= L+1
        KDES= LCDEST(K)
        LDES= MAXO(IDES,KDES)
        MDES= MINO(IDES,KDES)
```

```
      LPA= INDABL(LDES)
      IF(LPA .NE. NPA) GOTO 202
      LL= LADDR(LDES-1) + MDES
      ELPA(L)= ELPA(LL)
  202 CONTINUE
  201 CONTINUE
  200 CONTINUE
C   PRINT SUBSTRUCTURE STIFFNESS
      PRINT 3001,(K,K=1,KUREL)
 3001 FORMAT(//' CONDENSED SUBSTRUCTURE STIFFNESS MATRIX:'//4X,20I6)
      LA=1
      LE=1
      DO 203 K=1,KUREL
      PRINT 3000,K,(ELPA(L),L=LA,LE)
      LA= LE+1
      LE= LA+K
  203 CONTINUE
 3000 FORMAT(1X,I3,20F6.2)
      PRINT 3004,DET,NEG
 3004 FORMAT(//' LOG10 OF DETERMINANT=',F15.4/
     1          ' NO OF NEGATIVE PIVOTS=',I5//)
      RETURN
      END
      SUBROUTINE MOVE(NEW,LDES,IEQ)
C--------------------------------------------------
C
C   TO MOVE EQUATION COEFFICIENTS FROM FRONT PARTITION TO EQUATION
C                                                              BUFFER
C   NEW=1   ...   NEW EQUATION
C   NEW=0   ...   OLD EQUATION
C
C   LDES   ...   DESTINATION OF VARIABLE TO BE ELIMINATED
C
C   IEQ   ...   CURRENT ADDRESS OF BUFFER PONTER
C
C----------------------------------------------------
      COMMON ELPA(1)
      COMMON /PARTL/ NST,NEND
      COMMON /EQL/ KURPA
      COMMON /PARA/ L0,L1
      IF(NEW .EQ. 0) GOTO 1
C   ---   NEW EQUATION
      L= LADDR(LDES-1)
      M= IEQ
      DO 2 J=1,LDES
      L= L+1
      M=M+1
      ELPA(M)= ELPA(L)
      ELPA(L)= 0.
    2 CONTINUE
      IF(LDES .EQ. NEND) GOTO 7
      N1= LDES + 1
      K= 0
      DO 3 J=N1,NEND
      K=K + 1
      L= L + LDES
      M= M + 1
      ELPA(M)= ELPA(L)
      ELPA(L)= 0.
      L= L + K
    3 CONTINUE
```

```
    7   CONTINUE
        NDEQN= IEQ + KURPA
        IF(M .EQ. NDEQN) GOTO 5
        M1= M+1
        DO 6 I=M1,NDEQN
        ELPA(I)= 0.
    6   CONTINUE
        M= NDEQN
    5   CONTINUE
        ELPA(M + 1) = LDES
        ELPA(M + 2)= KURPA
        RETURN
C   ---   OLD EQUATION
    1   CONTINUE
        L= LDES + L0
        M= IEQ + NST - 1
        NREST= NST
        DO 4 J=NST,NEND
        M= M+1
        ELPA(M)= ELPA(L)
        ELPA(L)= 0.
        L= L + NREST
        NREST= NREST + 1
    4   CONTINUE
        RETURN
        END
        SUBROUTINE GAUSS(NEW,LDES,IEQ)
C-------------------------------------------------------
C
C       MODIFIES ALL EQUATIONS OF PARTITION NPA
C   (ELIMINATION OF VARIABLE LDES)
C
C   IEQ   ...   ADDRESS OF EQUATION BUFFER POINTER
C
C-----------------------------------------------------
        COMMON ELPA(1)
        COMMON /EIGEN/ DET,NEG
        COMMON /EQL/ KURPA
        COMMON /PARA/ L0,L1
        COMMON /PARTL/ NST,NEND
        NDIAG= IEQ + LDES
        PIVOT= ELPA(NDIAG)
        ELPA(NDIAG)= 0.
C   CHECK FOR SINGULARITY AND WORK OUT DETERMINANT
        IF(NEW .EQ. 0) GOTO 2
        PIVO= ABS(PIVOT)
        DET= DET + ALOG10(PIVO)
        IF(PIVO .LT. 1.E-20) CALL ERROR(PIVOT,LDES,2)
        IF(PIVOT .GT. 0.) GOTO 2
        NEG= NEG + 1
        CALL ERROR(PIVOT,LDES,1)
    2   CONTINUE
        L= L0
        MI= IEQ + NST- 1
        DO 1 I=NST,NEND
        MI= MI + 1
        CONS= ELPA(MI )
        IF(CONS .EQ. 0.) GOTO 3
        CONS= CONS/PIVOT
        M= IEQ
C   CALL VECTOR PROCESSOR
```

```
      CALL SVECT(ELPA,CONS,L,M,I)
      GOTO 1
    3 CONTINUE
      L= L + I
    1 CONTINUE
      ELPA(NDIAG)= PIVOT
C     MOVE BUFFER POINTER TO END OF EQUATION
      IEQ= IEQ + KURPA + 2
      RETURN
      END
      SUBROUTINE UNCOD(LCDEST,KUREL,LUDEST,INDABL,MAXPA,NDPAR)
C-----------------------------------------------------------------
C   UNCODES DESTINATION VECTOR LCDEST AND
C   UPDATES SPACE INDICATOR INDABL
C-----------------------------------------------------------------
      DIMENSION LCDEST(KUREL)
      DIMENSION LUDEST(KUREL)
      COMMON /EQL/ KURPA
      DIMENSION INDABL(MAXPA)
      DO 1 K=1,KUREL
      IRDY= 0
      CALL CODEST(K,NSTRES,LDES,LCDEST,KUREL)
      IF(LDES .EQ. 0) GOTO 3
      IF(NSTRES .NE. 0 .AND. NSTRES.NE. 1) GOTO 2
C   ---   VARIABLE LDES CAN BE ELIMINATED
      IRDY= 1
    2 CONTINUE
      NPA= IABS(INDABL(LDES))
      INDABL(LDES)= NPA
    3 CONTINUE
      IF(IRDY .EQ. 1) LDES= -LDES
      LUDEST(K)= LDES
    1 CONTINUE
CURRENT EQUATION LENGTH
      M= MAXPA
    5 CONTINUE
      IF(INDABL(M) .GT. 0) GOTO 4
      M= M-1
      GOTO 5
    4 CONTINUE
      KURPA= M
      NDPAR= INDABL(KURPA)
      RETURN
      END
      SUBROUTINE PALI(MAXPAR,MAXPA,INDABL,LPAL,NDPAR)
C-----------------------------------------------------------------
C   SETS UP PARTITION LIMIT ARRAY LPAL
C   DEPENDING ON SIZE OF FRONT BUFFER LFRBUF
C   AND MAXPA ( MAXIMUM FRONT WIDTH )
C   TOTAL NUMBER OF PARTITIONS REQUIRED: NDPAR
C-----------------------------------------------------------------
      DIMENSION INDABL(MAXPA),LPAL(MAXPAR)
      COMMON /BUFSZE/ NELZ,LFRBUF,LEQBUF
      LPAL(1)= 0
      DO 1 NPA=1,MAXPAR
      FAC= 2* LPAL(NPA) + 1
      FAC1= 2*LFRBUF
      LDSPD= SQRT(.25*FAC*FAC + FAC1) - .5*FAC
      LPAL(NPA + 1)= LPAL(NPA) + LDSPD
      IF(LPAL(NPA + 1) .GE. MAXPA) GOTO 2
    1 CONTINUE
```

```
      CALL ERROR(0.,NPA,4)
    2 CONTINUE
      LPAL(NPA+1)= MAXPA
      NDPAR= NPA
C-------------------------------------------------------------------
C   SET UP ARRAY INDABL ; CODING: '+'-VE=OCCUPIED;'-'VE=FREE
C-------------------------------------------------------------------
      DO 3 M=1,MAXPA
      DO 4 NPA=1,NDPAR
      IF(M .LE. LPAL(NPA+1)) GOTO 5
    4 CONTINUE
    5 CONTINUE
      INDABL(M)= -NPA
    3 CONTINUE
      RETURN
      END
      SUBROUTINE ASSEMB(LDES,LUDEST,NV,KUREL)
C-------------------------------------------------------------------
C   ASSEMBLES EQUATION LDES INTO CURRENT FRONT PARTITION
C
C-------------------------------------------------------------------
      COMMON ELPA(1)
      DIMENSION LUDEST(KUREL)
      LL= LADDR(LDES-1)
      DO 1 K=1,KUREL
      II= LUDEST(K)
      IF(II) 2,1,3
    2 CONTINUE
      II= -II
    3 CONTINUE
      IF(II .GT. LDES) GOTO 1
      L= LADST(K,NV)
      LF=LL + II
      ELPA(LF)= ELPA(LF) + ELPA(L)
    1 CONTINUE
      RETURN
      END
      FUNCTION LADDR(M)
C-----------------------------------------------------------
C   COMPUTES THE ADDRESS OF COEFF M,M IN CURRENT PARTITION
C-----------------------------------------------------------
      COMMON /PARA/ LO,L1
      COMMON /PARTL/ NST,NEND
      NS= NST -1
      MR= M-NS
      LADDR= MR*NS + MR*(MR+1)/2+ LO
      RETURN
      END
      FUNCTION LADST(I,J)
C-----------------------------------------------------------
C   COMPUTES ADRESS OF COEFF I,J IN ELEMENT STIFFNESS MATRIX
C-----------------------------------------------------------
      II= MAXO(I,J)
      JJ= MINO(I,J)
      LADST= JJ + II*(II-1)/2.
      RETURN
      END
      SUBROUTINE EMPDI(IEQ,NBLK)
C-------------------------------------------------------------
C   EMPTIES EQUATION BUFFER ONTO DISK WHEN FULL
C
```

```
C   IEQ   ...   BUFFER POINTER
C   NBLK   ...   NUMBER OF BLOCKS WRITTEN
C-----------------------------------------------------------
       COMMON ELPA(1)
       COMMON /PARA/ L0,L1
       COMMON /EQL/ KURPA
       COMMON /INCORE/ NBLKA,NBLKE
       COMMON /BLOKL/ LBLK
       COMMON /FILES/ NF6,NDIM6,NF7,NDIM7,NF8,NDIM8,NF9,NDIM9
       COMMON /IOCONV/ IREAD,IWRIT
       COMMON /BUFSZE/ NELZ,LFRBUF,LEQBUF,LFRBBL,LEQBBL
CHECK IF ANOTHER EQUATION FITS
       NDEQN= IEQ + KURPA + 2 - L1
       IF(NDEQN .LT. LEQBUF) RETURN
C   ---   DOES NOT FIT >> WRITE BUFFER ONTO DISK
       NBLOKS= (IEQ-L1)/LBLK
       CALL BLKIO(IWRIT,NF7,NBLK + 1,NBLOKS,ELPA,L1+1)
C   MOVE LAST BLOCK AT THE BEGINNING OF BUFFER IF NOT COMPLETELY FULL
       LEQE= IEQ
       LEQ= NBLOKS *LBLK + L1
       IEQ= L1
       IF(LEQ .EQ. LEQE) GOTO 10
       LEQ= LEQ + 1
       DO 1 I=LEQ,LEQE
       IEQ= IEQ + 1
       ELPA(IEQ)= ELPA(I)
    1  CONTINUE
   10  CONTINUE
       NBLK= NBLK + NBLOKS
       NBLKA= NBLK + 1
       NBLKE= NBLK + LEQBBL
       RETURN
       END
       SUBROUTINE RESBUF(IEQ,NBLK,NNBLK,LIEQ,ICYCL)
C-------------------------------------------------------------------
C   RESETS BUFFER POINTER IEQ TO LIEQ IN BLOCK NNBLK
C   AND SWAPS(ICYCL=1) OR READS(ICYCL=2) BLOCKS IF NECESSARY
C
C---------------------------------------------------------------
       COMMON ELPA(1)
       COMMON /PARA/ L0,L1
       COMMON /BLOKL/ LBLK
       COMMON /EQL/ KURPA
       COMMON /INCORE/ NBLKA,NBLKE
       COMMON /FILES/ NF6,NDIM6,NF7,NDIM7,NF8,NDIM8,NF9,NDIM9
       COMMON /IOCONV/ IREAD,IWRIT
       COMMON /BUFSZE/ NELZ,LFRBUF,LEQBUF,LFRBBL,LEQBBL
       COMMON /ENDQN / LASTBL
C-----------------------------------------------------------------
C   IS EQUATION STILL IN CORE ?
C---------------------------------------------------------------
       IF(NNBLK .LT. NBLKA) GOTO 2
       LEQ= LIEQ + KURPA + 2
       LEQB= (LEQ-1)/LBLK
       NDBLK= NNBLK + LEQB
       IF(NDBLK .LE. NBLKE) GOTO 1
C-----------------------------------------------------------------
C   NO   -   SWAP BLOCKS
C----------------------------------------------------------------
    2  CONTINUE
       IF(ICYCL .EQ. 2) GOTO 4
```

```
      NBLKEN= NBLKE
      IF(NBLKEN .LT. LASTBL) GOTO 3
      NBLKEN= LASTBL
    3 CONTINUE
      NBLOKS= NBLKEN-NBLKA + 1
      CALL BLKIO(IWRIT,NF7,NBLKA,NBLOKS,ELPA,L1+1)
    4 CONTINUE
      NBLOKS= LEQBBL
      LIMBLK= LASTBL - NNBLK + 1
      IF(NBLOKS .GT. LIMBLK) NBLOKS= LIMBLK
      CALL BLKIO(IREAD,NF7,NNBLK,NBLOKS,ELPA,L1+1)
      NBLKA= NNBLK
      NBLKE= NNBLK + NBLOKS - 1
      NBLK= NBLKA - 1
      IEQ= LIEQ + L1
      RETURN
    1 CONTINUE
      IEQ= (NNBLK-NBLKA)*LBLK + LIEQ + L1
      RETURN
      END
      SUBROUTINE SWAPF(NPA,NPAC,HBWR)
C-----------------------------------------------------------------
C   SWAPS FRONT PARTITIONS IN AND OUT OF CORE AS REQUIRED
C
C       NPA    ...   NEW PARTITION
C       NPAC    ...   CURRENT PARTITION
C-----------------------------------------------------------------
      COMMON ELPA(1)
      DIMENSION HBWR(1)
      COMMON /PARA/ L0,L1,L2
      COMMON /FILES/ NF6,NDIM6,NF7,NDIM7,NF8,NDIM8,NF9,NDIM9
      COMMON /IOCONV/ IREAD,IWRIT
      COMMON /BLOKL/ LBLK
      IF(NPA .EQ. NPAC) RETURN
      NBLKS8= NDIM8/LBLK
      IF(NPAC .EQ. 0) GOTO 1
      NFROM= (NPAC-1)*NBLKS8 + 1
      CALL BLKIO(IWRIT,NF8,NFROM,NBLKS8,ELPA,L0+1)
      HBWR(NPAC)= 1
    1 CONTINUE
      IF(NPA .EQ. 0) RETURN
      NFROM= (NPA-1)*NBLKS8 + 1
      IF(HBWR(NPA) .EQ. 1) CALL BLKIO(IREAD,NF8,NFROM,NBLKS8,ELPA,
      IF(HBWR(NPA) .EQ. 0) CALL CLEAR(ELPA,L0+1,L1)           L0+1)
      NPAC= NPA
      RETURN
      END
      SUBROUTINE CODEST(K,NSTRES,LDES,LCDEST,KUREL)
C---------------------------------------------------------------------
C   INTERPRETS CODED ELEMENT DESTINATIONS
C---------------------------------------------------------------------
      DIMENSION LCDEST(KUREL)
      LDES= LCDEST(K)
      DO 2 NSTRES= 1,32000
      IF(LDES .LT. 1000) GOTO 4
      LDES= LDES - 1000
    2 CONTINUE
    4 CONTINUE
      NSTRES= NSTRES - 2
      RETURN
      END
```

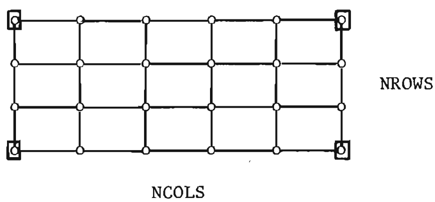
```
      SUBROUTINE CLEAR(ARRAY,NST,NEN)
      DIMENSION ARRAY(NEN)
      DO 1 N=NST,NEN
      ARRAY(N)= 0.
    1 CONTINUE
      RETURN
      END
      SUBROUTINE ICLAR(IARR,NEN)
      DIMENSION IARR(NEN)
      DO 1 N=1,NEN
      IARR(N)= 0
    1 CONTINUE
      RETURN
      END
      SUBROUTINE ERROR(F,I,N)
      GOTO (1,2,3) ,N
    1 CONTINUE
      PRINT 2000,F,I
 2000 FORMAT(/' *** NEGATIVE PIVOT (',E15.5,') AT DESTINATION',I5)
      RETURN
    2 CONTINUE
      PRINT 2001,F,I
 2001 FORMAT(//' SINGULARITY CHECK:'/
     1 ' NEAR ZERO OR ZERO PIVOT (',E15.5,') AT DESTINATION',I5)
      STOP
    3 CONTINUE
      PRINT 2002,I
 2002 FORMAT(//' *** DIMENSION OF ELPA (',I5,') TOO SMALL')
      STOP
    4 CONTINUE
      PRINT 2003,I
 2003 FORMAT(//' *** MAXIMUM NUMBER OF PARTITIONS (',I5,') ECCEEDED')
      STOP
      RETURN
      END
      SUBROUTINE SAVES(LDES,IEQ,ICYCL,JES)
C-------------------------------------------------------------
C   TO SAVE ON COMPUTATION AND SWAPPING TIME
C   FOR ZERO COEFFICIENTS
C--------------------------------------------------------
      COMMON ELPA(1)
      COMMON /PARTL/ NST,NEND
      COMMON /PARA/ LO,L1,L2
      IF(ICYCL .EQ. 2) GOTO 1
      L= LDES + LO
      NREST= NST
      DO 2 J=NST,NEND
      IF(ELPA(L) .NE. 0.) GOTO 3
      L= L + NREST
      NREST=NREST + 1
    2 CONTINUE
      JES=1
      RETURN
    3 CONTINUE
      JES= 0
      RETURN
    1 CONTINUE
      MI= IEQ + NST-1
      DO 4 I= NST,NEND
      MI= MI+1
      IF(ELPA(MI) .NE. 0.) GOTO 5
```

```
    4   CONTINUE
        JES= 1
        RETURN
    5   CONTINUE
        JES=0
        RETURN
        END
        SUBROUTINE SVECT(VECTOR,CONS,I1,I2,N)
C-------------------------------------------------------------
C   FAST VECTOR PROCESSING ROUTINE TO PERFORM :
C
C   VECTOR(I1)= VECTOR(I1) - VECTOR(I2)*CONS
C
C       VECTOR   ...     VECTOR
C       CONS    ...   SCALAR
C       I1    ...   START ADDRESS 1
C       I2    ...   START ADDRESS 2
C       N    ...   NUMBER OF OPERATIONS
C
C   MACHINE DEPENDENT CODING SHOULD BE USED IN ACTUAL IMPLEMENTATION
C   ON A GIVEN MACHINE.
C   CODING SHOWN IS STANDARD FORTRAN
C--------------------------------------------------------------------
        DIMENSION VECTOR(1)
        DO 2 J=1,N
        I1= I1+1
        I2= I2+1
        VECTOR(I1)= VECTOR(I1) - VECTOR(I2)*CONS
    2   CONTINUE
        RETURN
        END
```

```
      SUBROUTINE BLKIO(IRW,LUN,NBLA,NBLOKS,BUFFER,IADD)
C-----------------------------------------------------------------------
C   SUBROUTINE TO READ/WRITE DIRECTLY INTO BUFFER
C
C   IRW   ...   SWITCH FOR READ/WRITE
C                IRW=1   ...   READ
C                IRW=2   ...   WRITE
C   LUN   ...   LOGICAL UNIT NUMBER
C   NBLA   ...   FIRST BLOCK
C   NBLOKS   ...   NUMBER OF BLOCKS TO BE READ/WRITTEN
C   BUFFER   ...   BUFFER
C   IADD   ...   START ADDRES IN BUFFER
C
C----------------------------------------------------------------------
      COMMON /IO/ IPAR(6),ISTAT(2)
      COMMON /BLOKL/ LBLK
      DIMENSION BUFFER(1)
C   INSERT MACHINE DEPENDENT CODING HERE
      RETURN
      END
      SUBROUTINE FILO
C-----------------------------------------------------------------
C   SUBROUTINE TO ASSIGN CHANNEL NUMBERS
C   AND OPEN DIRECT ACCESS AND BLOCKED FILES
C
C   CHANNEL#      SIZE          CONTENTS
C       NF6       NELZ          ELEMENT STIFFNESS
C       NF9    D.O.F. + 2       ELEMENT DESTINATIONS
C       NF7       LBLK          BLOCKED EQUATIONS
C       NF8       LBLK          BLOCKED PARTITIONS
C
C----------------------------------------------------------------
      COMMON /FILES/ NF6,NDIM6,NF7,NDIM7,NF8,NDIM8,NF9,NDIM9
      COMMON /BUFSZE/ NELZ,LFRBUF,LEQBUF
      COMMON /IOCONV/ IREAD,IWRIT
      COMMON /BLOKL/ LBLK
      IREAD= 1
      IWRIT= 2
C   INSERT MACHINE DEPENDENT CODING HERE
      RETURN
      END
```

APPENDIX C - TEST PROGRAM

In the following a test program is listed which can be used to test the subroutines PREFR and SFRONT. The example is a substructure condensation for a regular patch of square 4 node/8 d.o.F. Elements. The stiffness matrix of the Elements is read in and all the nodes except the 4 corner nodes of the super element are condensed out. Results can be obtained for different mesh and buffer sizes



□ substructure nodes

o element nodes

```
C----------------------------------------------------------
C   PROGRAM TO TEST OUT SUBROUTINE SFRONT
C-----------------------------------------------------------
      COMMON ELPA(5000),MAXPA,NELEMZ,LDEST(62)
      COMMON /BUFSZE/ NELZ,LFRBUF,LEQBUF,LFRBBL,LEQBBL
      COMMON /MESH/ NROWS,NCOLS,NODES
      COMMON /BLOKL/ LBLK
      COMMON /FILES/ NF6,NDIM6,NF7,NDIM7,NF8,NDIM8,NF9,NDIM9
      NCR= 7
      CALL ASSIGN(NCR,'INPUT')
      NELZ= 36
      READ(NCR,1000) NROWS,NCOLS
 1000 FORMAT(16I5)
      NELEMZ= NROWS*NCOLS
      NODES= (NCOLS+1)*(NROWS+1)
      NDOFS= NODES*2
      READ(NCR,1000) LFRBUF,LEQBUF,LBLK
      CALL FILO
      READ(NCR,1000) ISTIF
      IF(ISTIF .NE. 1) GOTO 1
      READ(NCR,1001) (ELPA(N),N=1,NELZ)
 1001 FORMAT(8F10.0)
      WRITE (NF6'1) (ELPA(I),I=1,NELZ)
      GOTO 2
    1 CONTINUE
      READ(NF6'1) (ELPA(I),I=1,NELZ)
    2 CONTINUE
      PRINT 3000
 3000 FORMAT(1H1//' *** SUBSTRUCTURE CONDENSATION EXAMPLE ***')
      PRINT 3001,NROWS,NCOLS
 3001 FORMAT(//' NUMBER OF ELEMENT ROWS=',I5/
     1      ' NUMBER OF ELEMENT COLUMNS=',I5)
      PRINT 3002,NELEMZ,NODES,NDOFS
 3002 FORMAT(' NUMBER OF ELEMENTS=',I5/
     1        ' NUMBER OF NODES   =',I5/
     1        ' NUMBER OF D.O.F.  =',I5)
      PRINT 3003,LFRBUF,LEQBUF
 3003 FORMAT(' SIZE OF FRONT BUFFER   =',I5/
     1        ' SIZE OF EQUATION BUFFER=',I5)
      PRINT 3004,(K,K=1,8)
 3004 FORMAT(//' ELEMENT STIFFNESS MATRIX:'//4X,20I6)
      LA= 1
      LE= 1
      DO 200 K=1,8
      PRINT 3005,K,(ELPA(L),L=LA,LE)
      LA=LE+1
      LE= LA+K
  200 CONTINUE
 3005 FORMAT(1X,I3,20F6.2)
      CALL PREFR
      CALL SFRONT
      STOP
      END
```

```
      SUBROUTINE GETELN(NELEM,KUREL,LVABL)
C----------------------------------------------------------------
C   THIS IS A DUMMY SUBROUTINE FOR TESTING SFRONT
C   IT CREATES CONNECTIVITY DATA FOR A REGULAR
C   ASSEMBLY OF 4 NODE/8 D.O.F. ELEMENTS
C
C   NROWS   ...   NUMBER OF ELEMENT ROWS
C   NCOLS   ...   NUMBER OF ELEMENT COLUMNS
C   NODES   ...   NUMBER OF NODES
C
C----------------------------------------------------------------
      DIMENSION LVABL(8)
      COMMON /MESH/ NROWS,NCOLS,NODES
      KUREL= 8
      NROW= (NELEM-1)/NCOLS + 1
      NCOL= NELEM - NCOLS*(NROW-1)
      NCOL1= NCOLS + 1
      LVABL(1)= NCOL + (NROW-1)*NCOL1
      LVABL(2)= NCOL + 1 + (NROW-1)*NCOL1
      LVABL(3)= NCOL + 1 + NROW*NCOL1
      LVABL(4)= NCOL + NROW*NCOL1
      DO 1 N=5,8
      LVABL(N)= LVABL(N-4) + NODES
    1 CONTINUE
      RETURN
      END
      SUBROUTINE GETSUN(KUREL,LVABL)
C------------------------------------------------------------
C   THIS IS A DUMMY SUBROUTINE TO TEST THE SUBSTRUCTURING
C   CAPABILITIES OF SFRONT.
C   IT CREATES CONNECTIVITY DATA FOR A
C   4 NODE/8 D.O.F. SUPERELEMENT
C
C------------------------------------------------------------
      DIMENSION LVABL(8)
      COMMON /MESH/ NROWS,NCOLS,NODES
      KUREL= 8
      NCOL1= NCOLS + 1
      LVABL(1)= 1
      LVABL(2)= NCOL1
      LVABL(3)= NODES
      LVABL(4)= NCOL1*NROWS + 1
      DO 1 N=5,8
      LVABL(N)= LVABL(N-4) + NODES
    1 CONTINUE
      RETURN
      END
```

```
*** SUBSTRUCTURE CONDENSATION EXAMPLE ***

NUMBER OF ELEMENT ROWS=     2
NUMBER OF ELEMENT COLUMNS=    10
NUMBER OF ELEMENTS=   20
NUMBER OF NODES   =   33
NUMBER OF D.O.F.  =   66
SIZE OF FRONT BUFFER   =  512
SIZE OF EQUATION BUFFER=  256

ELEMENT STIFFNESS MATRIX:

         1      2      3      4      5      6      7      8
  1   5.00
  2  -2.50   5.00
  3  -2.50   0.00   5.00
  4   0.00  -2.50  -2.50   5.00
  5   1.25   1.25  -1.25  -1.25   5.00
  6  -1.25  -1.25   1.25   1.25   0.00   5.00
  7  -1.25  -1.25   1.25   1.25  -2.50  -2.50   5.00
  8   1.25   1.25  -1.25  -1.25  -2.50  -2.50   0.00   5.00

MAXIMUM FRONT WIDTH=   30
MAX. NO. OF PARTITIONS=    1



CONDENSED SUBSTRUCTURE STIFFNESS MATRIX:

         1      2      3      4      5      6      7      8
  1   1.12
  2  -0.08   1.12
  3  -0.74  -0.29   1.12
  4  -0.29  -0.74  -0.08   1.12
  5   0.21  -0.01  -0.21   0.01   2.05
  6   0.01  -0.21  -0.01   0.21   0.01   2.05
  7  -0.21   0.01   0.21  -0.01  -0.05  -2.01   2.05
  8  -0.01   0.21   0.01  -0.21  -2.01  -0.05   0.01   2.05

LOG10 OF DETERMINANT=        55.9191
NO OF NEGATIVE PIVOTS=    0
```

*** SUBSTRUCTURE CONDENSATION EXAMPLE ***

```
NUMBER OF ELEMENT ROWS=     2
NUMBER OF ELEMENT COLUMNS=    10
NUMBER OF ELEMENTS=   20
NUMBER OF NODES    =   33
NUMBER OF D.O.F.   =   66
SIZE OF FRONT BUFFER    =  256
SIZE OF EQUATION BUFFER=  256
```

ELEMENT STIFFNESS MATRIX:

	1	2	3	4	5	6	7	8
1	5.00							
2	-2.50	5.00						
3	-2.50	0.00	5.00					
4	0.00	-2.50	-2.50	5.00				
5	1.25	1.25	-1.25	-1.25	5.00			
6	-1.25	-1.25	1.25	1.25	0.00	5.00		
7	-1.25	-1.25	1.25	1.25	-2.50	-2.50	5.00	
8	1.25	1.25	-1.25	-1.25	-2.50	-2.50	0.00	5.00

```
MAXIMUM FRONT WIDTH=   30
MAX. NO. OF PARTITIONS=    2
```

CONDENSED SUBSTRUCTURE STIFFNESS MATRIX:

	1	2	3	4	5	6	7	8
1	1.12							
2	-0.08	1.12						
3	-0.74	-0.29	1.12					
4	-0.29	-0.74	-0.08	1.12				
5	0.21	-0.01	-0.21	0.01	2.05			
6	0.01	-0.21	-0.01	0.21	0.01	2.05		
7	-0.21	0.01	0.21	-0.01	-0.05	-2.01	2.05	
8	-0.01	0.21	0.01	-0.21	-2.01	-0.05	0.01	2.05

```
LOG10 OF DETERMINANT=        55.9191
NO OF NEGATIVE PIVOTS=     0
```

APPENDIX D - NOMENCLATURE

Symbol	Meaning
c_{ij}	Equation coefficient
ℓ	Address of coefficient in long vector ELPA
ℓ_o	Start address in ELPA of space reserved for the Front matrix
m	Front width
$\Delta\ell_N$	Number of Equations in partition N
k_N	"destination" of first equation in partition N
n_p	Size of storage space for the Front matrix
k, i	Coefficient indexes
F_i	Coefficient on right hand side
x_n	Variable
R_i	Residual
K_{ij}	Structure stiffness coefficient
Δx_j	error on Solution for x_j

APPENDIX E - REFERENCES

1. IRONS, Bruce M. "A Frontal Solution Technique for Finite Element Analysis", Int. Jnl. Num. Meth. Engng., Vol. 2, pp. 5-32, 1970.

2. LIGHT, M.F. and LUXMORE, "Application of the Front Solution to two and three-dimensional Elasto-plastic Crack Problems", Int. Jnt. Num. Meth. Engng., Vol. 11, pp. 393-395, 1977.

3. HOOD, P. "Frontal Solution Program for Unsymmetric Matrices", Int. Jnl. Num. Meth. Engng., Vol. 10, pp. 379-399, 1976.

4. NATARAJAN, R. "Front Solution Program for Transmission Tower Analysis", Computers and Structures, Vol. 5, pp. 59-64, 1975.

5. ALIZADEH, A. and WILL, G.T. "A Substructural Frontal Solver and its Application to Localized Material Non-linearity", Computer and Structures, Vol. 10, pp. 225-231, 1979.

6. FELIPA, C.A. "Solution of Linear Equations with Skyline-Stored Symmetric Matrix", Computers and Structures, Vol. 5, pp. 13-29, 1975.

CIVIL ENGINEERING RESEARCH REPORTS

CE No.	Title	Author(s)	Date
1	Flood Frequency Analysis: Logistic Method for Incorporating Probable Maximum Flood	BRADY, D.K.	February, 1979
2	Adjustment of Phreatic Line in Seepage Analysis by Finite Element Method	ISAACS, L.T.	March, 1979
3	Creep Buckling of Reinforced Concrete Columns	BEHAN, J.E. & O'CONNOR, C.	April 1979
4	Buckling Properties of Monosymmetric I-Beams	KITIPORNCHAI, S. & TRAHAIR, N.S.	May, 1979
5	Elasto-Plastic Analysis of Cable Net Structures	MEEK, J.L. & BROWN, P.L.D.	November, 1979
6	A Critical State Soil Model for Cyclic Loading	CARTER, J.P., BOOKER, J.R. & WROTH, C.P.	December, 1979
7	Resistance to Flow in Irregular Channels	KAZEMIPOUR, A.K. & APELT, C.J.	February, 1980
8	An Appraisal of the Ontario Equivalent Base Length	O'CONNOR, C.	February, 1980
9	Shape Effects on Resistance to Flow in Smooth Rectangular Channels	KAZEMIPOUR, A.K. & APELT, C.J.	April, 1980
10	The Analysis of Thermal Stress Involving Non-Linear Material Behaviour	BEER, G. & MEEK, J.L.	April, 1980
11	Buckling Approximations for Laterally Continuous Elastic I-Beams	DUX, P.F. & KITIPORNCHAI, S.	April, 1980
12	A Second Generation Frontal Solution Program	BEER, G.	May, 1980
13	Combined Stiffness for Beam and Column Braces	O'CONNOR, C.	May, 1980
14	Beaches:- Profiles, Processes and Permeability	GOURLAY, M.R.	June, 1980

CURRENT CIVIL ENGINEERING BULLETINS

4 *Brittle Fracture of Steel — Performance of ND18 and SAA A1 structural steels: C. O'Connor (1964)*

5 *Buckling in Steel Structures — 1. The use of a characteristic imperfect shape and its application to the buckling of an isolated column: C. O'Connor (1965)*

6 *Buckling in Steel Structures — 2. The use of a characteristic imperfect shape in the design of determinate plane trusses against buckling in their plane: C. O'Connor (1965)*

7 *Wave Generated Currents — Some observations made in fixed bed hydraulic models: M.R. Gourlay (1965)*

8 *Brittle Fracture of Steel — 2. Theoretical stress distributions in a partially yielded, non-uniform, polycrystalline material: C. O'Connor (1966)*

9 *Analysis by Computer — Programmes for frame and grid structures: J.L. Meek (1967)*

10 *Force Analysis of Fixed Support Rigid Frames: J.L. Meek and R. Owen (1968)*

11 *Analysis by Computer — Axisymetric solution of elasto-plastic problems by finite element methods: J.L. Meek and G. Carey (1969)*

12 *Ground Water Hydrology: J.R. Watkins (1969)*

13 *Land use prediction in transportation planning: S. Golding and K.B. Davidson (1969)*

14 *Finite Element Methods — Two dimensional seepage with a free surface: L.T. Isaacs (1971)*

15 *Transportation Gravity Models: A.T.C. Philbrick (1971)*

16 *Wave Climate at Moffat Beach: M.R. Gourlay (1973)*

17 *Quantitative Evaluation of Traffic Assignment Methods: C. Lucas and K.B. Davidson (1974)*

18 *Planning and Evaluation of a High Speed Brisbane-Gold Coast Rail Link: K.B. Davidson, et al. (1974)*

19 *Brisbane Airport Development Floodway Studies: C.J. Apelt (1977)*

20 *Numbers of Engineering Graduates in Queensland: C. O'Connor (1977)*