R YAL AIR CRAFT ESTABLISHMENT
BEDFORD.

C.P. No. 1223

C.P. No. 1223

PROCUREMENT EXECUTIVE, MINISTRY OF DEFENCE

AERONAUTICAL RESEARCH COUNCIL

CURRENT PAPERS

Fortran Subroutines for Finite Element Analysis

by

B. C. Merrifield

Structures Dept., R.A.E., Farnborough

LONDON: HER MAJESTY'S STATIONERY OFFICE

1972

PRICE *80 p* NET

UDC 531.25 : 518.5 : 531.258

CP No.1223 *
August 1971

FORTRAN SUBROUTINES FOR FINITE ELEMENT ANALYSIS

by

B. C. Merrifield

SUMMARY

Twelve subroutines, written in ICL 1900 Fortran are presented for matrix and other operations which are commonly encountered in the finite element analysis of structures. Although the subroutines have been developed specifically to deal with problems concerning flat plates they clearly possess some wider generality.

Details are given of each subroutine including its method of use and its complete listing.

---

* Replaces RAE Technical Report 71156 - ARC 33734

CONTENTS

Drgs. 011/96920 to 96922

1 INTRODUCTION

Twelve subroutines are presented for various matrix and other operations commonly used in the finite element method of structural analysis and specifically for the analysis of flat plates. They are written in 1900 Fortran which is the ICL implementation of ASA Fortran.

The subroutines are the result of experience in Structures Department, RAE with the development of several research-type computer programs for the finite element analysis of plate bending problems where it is important to take advantage of symmetry and bandedness in the matrices to reduce computer storage requirements and execution time.

The subroutines simplify the development of finite element programs, and include many operations which are non-standard and make use of information peculiar to finite element analyses. The use of double suffix arrays is generally avoided because of the time penalty[1] which is associated with their use.

The Report commences with a brief description of the kinds of operations and matrices which are encountered in finite element analyses; this is followed by a description and illustrative example of the use of each subroutine.

The subroutines, which are listed in full in the Appendix, have been tested and used in current finite element programs but there must, however, be instances where their efficiency and generality can be improved.

2 GENERAL NOTATION

[ ]	matrix or row vector
{ }	column vector
[k]	element stiffness matrix
[K]	stiffness matrix for the whole structure
n	outward pointing normal from the boundary
{q}	column vector of generalised displacements
{Q}	column vector of generalised loads
s	distance measured around the boundary in the clockwise sense
x,y	rectangular Cartesian coordinates

The programming notation which is used in the subroutines is defined in section 4.

3 MATRIX OPERATIONS IN THE FINITE ELEMENT METHOD

The finite element method requires that the plate be divided by imaginary lines into a number of 'finite elements' which are then assumed to be connected at a discrete number of points on their boundaries. Fig.1 shows a simple example of such a plate divided into triangular finite elements which are connected at the corners (nodes) and mid-points of the sides. The plate is described by specifying the node and mid-point numbers of each element together with the coordinates of the nodes with respect to a given set of axes.

Finite element methods can be divided into two main categories, displacement (stiffness) methods or force (flexibility) methods. Since both of these involve the same type of operations our subsequent references can be to the displacement method only. It is pertinent to consider briefly the kind of operation encountered and the type of matrix, i.e. symmetric, banded, square etc. A description of the finite element method is given in the book by Zienkiewicz[2] where the square, symmetric element stiffness matrix [k] is given as

$$[k] = \int_{\substack{\text{element}\\ \text{area}}} \int [B]^T [D][B] \, dx \, dy \qquad (1)$$

where [D] is a square symmetric matrix called the elasticity matrix which contains the material properties of the plate and [B] is a rectangular matrix containing functions of x and y. Frequently, however, the matrix [B] is expressed as the product of various rectangular matrices, e.g. the element stiffness matrix of a recently developed finite element is given by

$$[k] = \int_{\substack{\text{element}\\ \text{area}}} \int [T]^T [F]^T [C]^T [D][C][F][T] \, dx \, dy \qquad (2)$$

where [T] and [C] are rectangular transformation matrices of constants and [F] is a rectangular matrix of so-called shape functions.

The individual stiffness matrices for each element are then compounded into a stiffness matrix [K] for the whole structure. This (global) stiffness matrix is also symmetric and all the non-zero terms are ideally contained within a band surrounding the leading diagonal. The width of this band depends critically on the way in which the elements are numbered and it is evident that it is required to number the elements in such a way that the greatest difference between the node or mid-point numbers in any one element is as small as possible. In simple regular structures the optimum node numbering is frequently obvious but in large complicated structures this is far from the case. Attention is drawn here to work on methods of bandwidth reduction[3,4] because of the considerable advantages which accrue in the saving of storage space for problems involving large numbers of elements. The bandwidth also influences strongly the execution time for the solution of the final stiffness equations for the unknown generalised displacements {q} where

$$[K]\ \{q\} = \{Q\} \quad . \tag{3}$$

This computation invariably accounts for a substantial part of the total computing time.

The flow chart of a typical finite element program is shown in Fig.2 and the subroutines corresponding to the steps in the chart are as follows

FE INPUT FE PARAMETERS	Program Parameters
MAT ATBA INTEGRAL MAT ATBA	Element stiffness matrix assembly
ADD TO BANDMAT ADD TO VECTOR	Add element contribution to global stiffness matrix and right hand side
ALTER BANDMAT SOLVE CONSTRAINED BANDMAT SOLVE BANDMAT	Impose boundary conditions and solve

Thus, the element stiffness matrix (equation 2) can be built up using MAT ATBA for the product $[C]^T[D][C]$ (= [E] say), INTEGRAL MAT ATBA for the area integral of $[F]^T[E][F]$ (= [G] say) and MAT ATBA again for the product $[T]^T[G][T]$.

4 DESCRIPTION OF SUBROUTINES

Each subroutine is described in detail but the following generalities and notation apply to them all.

The individual subroutines are self contained with the two exceptions of SOLVE CONSTRAINED BANDMAT which calls SOLVE BANDMAT and FE INPUT which calls FE PARAMETERS. Every identifier used in the subroutines is either locally declared or is an argument of the subroutine. Extensive use is made of dynamic dummy arrays which, in 1900 Fortran, cannot be incorporated in COMMON storage specified in the subroutine. The user can, however, construct his program in such a way as to make use of the COMMON facility.

If a two-dimensional array A in the calling segment is used to store a m(rows) by n(cols) matrix then the following program notation is employed

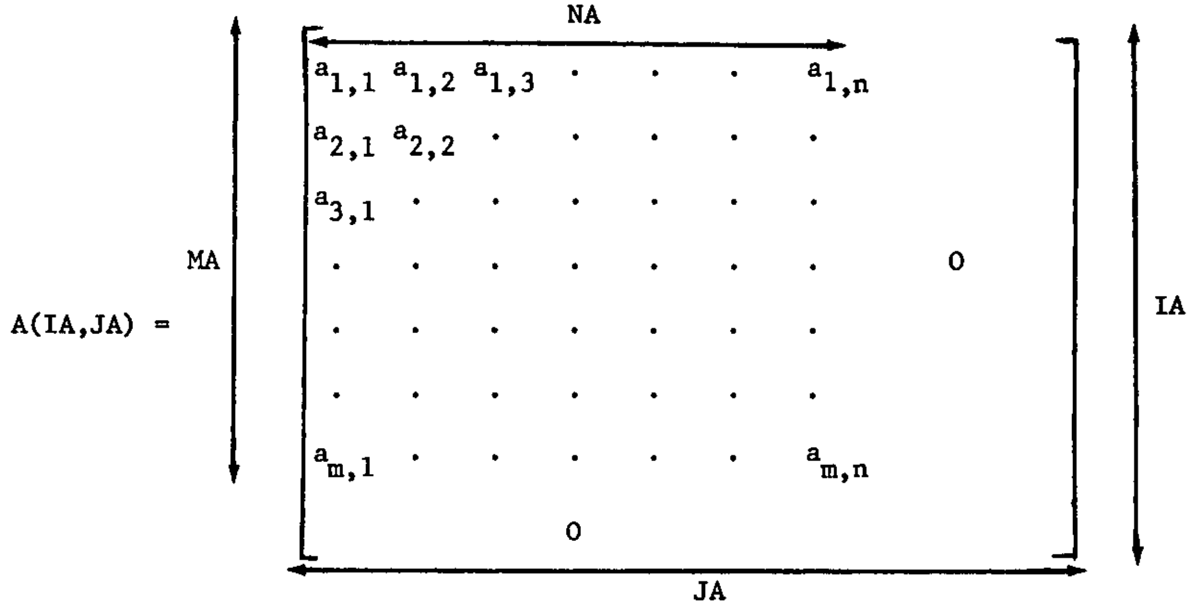
IA is the first dimension of A ($\geqslant$ m),
JA is the second dimension of A ($\geqslant$ n),
MA = m,
NA = n.

Similarly IB is the first dimension of array B, IC is the first dimension of array C and so on. In general the prefices M and N refer to the dimensions of the actual matrix while I and J refer to the overall dimensions of the array in which the matrix is stored. This notation allows an array A to be declared in the calling segment as A (IA,JA) and to be used for varying sizes of matrices so that, for instance, several examples with differing dimensions may be run at one time.

The following diagrams illustrate this notation and the storage of the two types of matrices, banded symmetric and rectangular (or square), within the calling segment. The m x n rectangular matrix [A] where

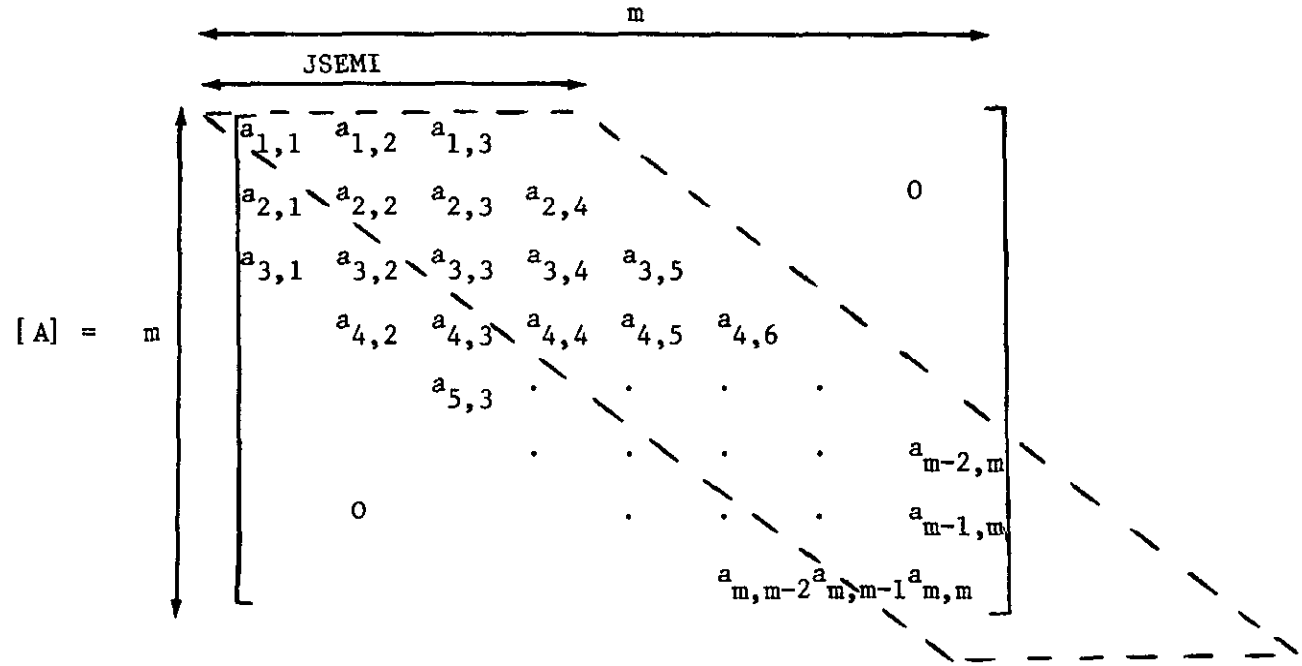
$$[A] = \overbrace{\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdot & \cdot & \cdot & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{3,1} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{m,1} & \cdot & \cdot & \cdot & \cdot & \cdot & a_{m,n} \end{bmatrix}}^{n} \quad (m \text{ rows})$$

is located in the array A(IA,JA), IA≥m, JA≥n as follows

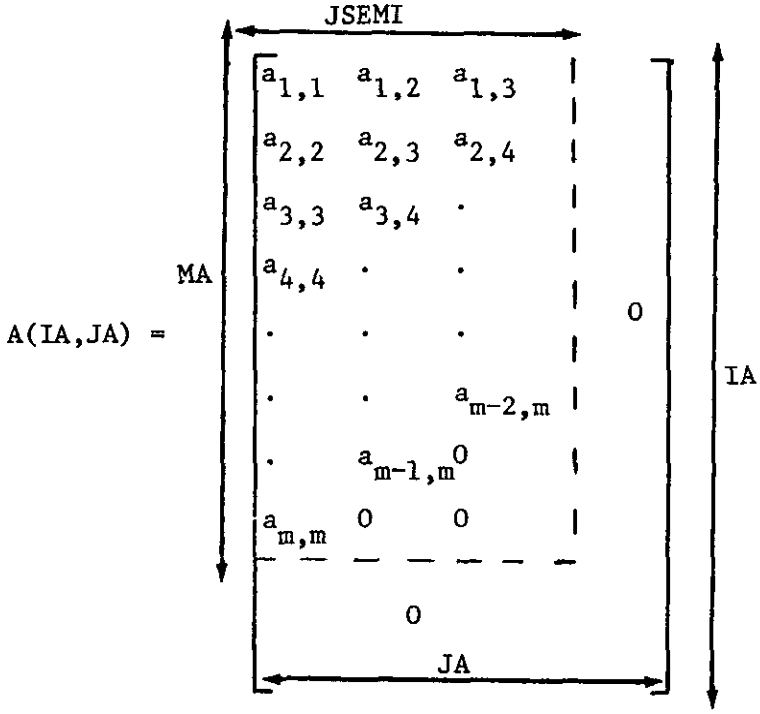


It should be noted that, for convenience, the elements of the above matrix [A] are located in the array by a scheme which is at variance with standard Fortran practice. If [A] occupies the whole of the array, i.e. IA = MA, JA = NA, then both storage schemes coincide and A is dimensioned A(MA,NA).

A m by m banded symmetric matrix with a semi-bandwidth (JSEMI) of 3, where semi-bandwidth = (bandwidth + 1)/2, of the form



is located in the rectangular array A(IA,JA) where IA≥m, JA≥JSEMI as follows



With the exception of subroutines FE INPUT, FE PARAMETERS and SOLVE CONSTRAINED BANDMAT all dummy arguments are declared as one-dimensional arrays within the subroutines. Hence an array declared A(IA,JA) in the calling segment is automatically stored column by column in the one-dimensional array A(IAJA) (where IAJA = IA*JA) within the subroutine and so reference to element A(11) in the subroutine corresponds, if IA = 10 say, to element A(1,2) in the calling segment. To dimension these arrays the additional notation

IAJA = IA*JA

IAIA = IA*IA

JAJA = JA*JA

MANA = MA*NA

IAJSEMI = IA*JSEMI

is required, where * denotes multiplication, with similar expressions for MAMA, IBJB etc.

The following integer notation is used

NEL = number of elements

NAM = number of nodes and mid-points

NEQ = number of equations

NNODE = number of nodes

NCONC = number of concentrated loads

NF = number of degrees of freedom at a node (NF≥0)

MF = number of degrees of freedom at a mid-point

JEL = number of nodes and mid-points required to define one element, but if MF is negative (see below) JEL must be set equal to -MF*JEL

A negative value of MF is used to signal a special type of input for the node numbers in a method which makes use of an extended interpolation and which will be described in a later Report; locations are left for pseudo mid-point numbers which are not, however, prescribed at input. The following notation refers to real numbers

NU = Poisson's ratio

D = flexural rigidity

Qo = intensity of uniformly distributed load

Additional notation is defined within the subroutine descriptions.

4.1 Subroutine FE INPUT

The subroutine reads and prints the data necessary to describe the elemental divisions, material properties and applied loading. The subroutine FE PARAMETERS (see section 4.2) is called to evaluate additional parameters used in subsequent calculations. The subroutine requires the following arrays to be dimensioned in the calling segment.

```
REAL X(IXY), Y(IXY), CMAG(ICON)
INTEGER ELNO(IEL,JEL), IDENT(ID), CNODE(ICON)
```

where IXY≥NAM, ICON≥NCONC≥1, IEL≥NEL and ID≥NAM. Note, even if there are no concentrated loads CMAG and CNODE must be dimensioned. The instruction

```
CALL FE INPUT (NEL, Qo, NCONC, CNODE, CMAG, ELNO, X, Y, IEL, JEL, IDENT,
ID, NF, MF, NEQ, NAM, NNODE, JSEMI, NU, D, IXY, ICON)
```

then results in the reading (in free format) of the following data

(i) CAPTION, a single card containing any required heading in columns 1 to 72.

(ii) NEL, NCONC, NU, Qo, D (Note, NU must be declared real in the calling segment, and Qo is subsequently assumed zero if its absolute value is less than 0.0000001).

(iii) CNODE(I),CMAG(I) an integer array and a real array respectively containing the node numbers and magnitudes of each applied concentrated load. (I = 1,2,.....NCONC)
Omit if NCONC = 0.

(iv) ELNO(I,J), an integer array containing the node numbers and, if appropriate, the mid-point numbers taken in sequence around each element starting at a node. The array is read a row at a time, one finite element to a row. (I = 1,2,.....NEL, J = 1,2,.....JEL. Note, if MF is negative then J = 1,1-MF,1-2*MF,.....JEL+MF+1).

(v) NODE, X(NODE), Y(NODE). The real arrays X and Y are filled by reading NNODE cards, each card containing an integer (NODE) equal to a node number and two real numbers equal to the X and Y coordinates of that node.

The subroutine FE PARAMETERS is called to evaluate additional parameters and on return the pertinent information is printed. All parameters, input and calculated, are returned through the argument list to the calling segment. Note that a negative value of MF requires a special FE PARAMETERS subroutine.

For the purpose of illustration, consider the square plate shown in Fig.1. The plate is divided into 8 elements, hence NEL = 8, and assuming the bending deflection w and its derivatives $\partial w/\partial x$, $\partial w/\partial y$ to be the

generalised displacements at a node and $\partial w/\partial n$ to be the generalised displacement at a mid-point, then NF = 3 and MF = 1. There are six nodes and mid-points to each element, hence JEL = 6. In the calling segment the dimension statements are

```
REAL X(100),Y(100),CMAG(10), NU
INTEGER ELNO(20,6), IDENT(130), CNODE(10)
```

which allows for a case with up to 10 concentrated loads, 20 elements and 100 nodes and mid-points. The calling statement is then

```
CALL FE INPUT (NEL, Qo, NCONC, CNODE, CMAG, ELNO, X, Y, 20, 6, IDENT,
130, 3, 1, NEQ, NAM, NNODE, JSEMI, NU, D, 100, ICON)
```

The subroutine reads the following data for the square plate of unit length side with Poisson's ratio 0.3 and flexural rigidity 1.0, under a concentrated load of magnitude 1.0 applied at node 13.

```
CAPTION CARD
 8     1       0.3      0.0      1.0
13     1.0
 1       2       3       8      13       9
 3       4       5       7      13       8
 5       6      15      14      13       7
15      16      25      17      13      14
25      24      23      18      13      17
23      22      21      19      13      18
21      20      11      12      13      19
11      10       1       9      13      12
 1    -0.5    -0.5
 3     0.0    -0.5
 5     0.5    -0.5
11    -0.5     0.0
13     0.0     0.0
15     0.5     0.0
21    -0.5     0.5
23     0.0     0.5
25     0.5     0.5
```

The print-out from the subroutine is then as follows

Print of CAPTION CARD

NUMBER OF ELEMENTS	=	8
NUMBER OF NODES	=	9
NUMBER OF NODES AND MID-POINTS	=	25
NUMBER OF EQUATIONS	=	43
SEMI BANDWIDTH OF STIFFNESS MATRIX	=	23
POISSONS RATIO	=	0.3000
FLEXURAL RIGIDITY	=	1.0000
INTENSITY OF UD LOAD	=	0.0000

CONCENTRATED LOAD

NODE MAGNITUDE

13 1.0

ELNO						NODE	X	Y
1	2	3	8	13	9	1	-0.5	-0.5
3	4	5	7	13	8	3	0.0	-0.5
5	6	15	14	13	7	5	0.5	-0.5
15	16	25	17	13	14	11	-0.5	0.0
25	24	23	18	13	17	13	0.0	0.0
23	22	21	19	13	18	15	0.5	0.0
21	20	11	12	13	19	21	-0.5	0.5
11	10	1	9	13	12	23	0.0	0.5
						25	0.5	0.5

4.2 Subroutine FE PARAMETERS

The subroutine evaluates parameters which are required in subsequent calculations. More explicitly, given the integers NEL, IEL, JEL, ID, NF, MF and the integer array ELNO, the instruction

CALL FE PARAMETERS (NEL, ELNO, IEL, JEL, IDENT, ID, NF, MF, NEQ, NAM, NNODE, JSEMI)

results in the calculation of the integer array IDENT which relates node and mid-point numbers with positions in the global stiffness matrix, together with the integers NEQ, NAM, NNODE, JSEMI, and returns them through the argument list. The subroutine requires the following arrays to be dimensioned in the calling segment

INTEGER ELNO(IEL, JEL), IDENT(ID)

The semi-bandwidth of the stiffness matrix, JSEMI, is calculated by evaluating the semi-bandwidth associated with each element (a function of its maximum and minimum node or mid-point numbers) and taking the greatest of these.

To illustrate, consider the same example as in section 4.1. In the calling segment the arrays are dimensioned

INTEGER ELNO(20,6), IDENT(100)

which allows for 20 elements and 100 nodes and mid-points. ELNO contains the node and mid-point numbers as follows

NEL (=8)

1	2	3	8	13	9
3	4	5	7	13	8
5	6	15	14	13	7
15	16	25	17	13	14
25	24	23	18	13	17
23	22	21	19	13	18
21	20	11	12	13	19
11	10	1	9	13	12
0	0	0	0	0	0
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

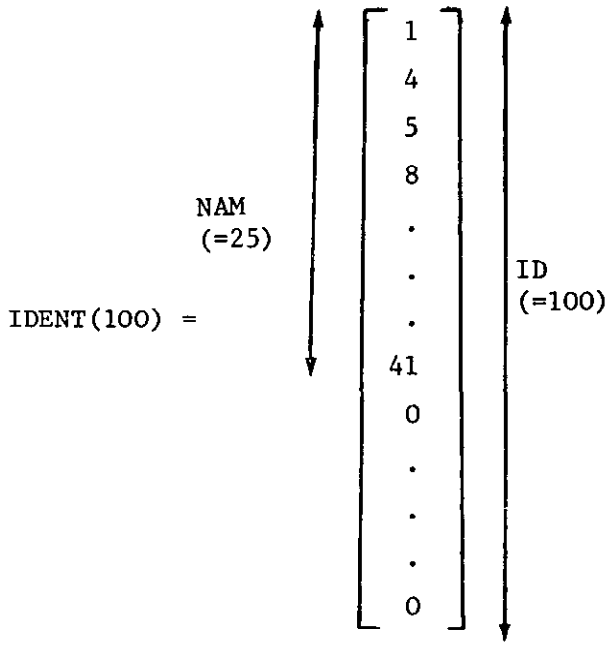
JEL (=6)

IEL (=20)

The calling statement for this case is then

CALL FE PARAMETERS (8, ELNO, 20, 6, IDENT, 100, 3, 1, NEQ, NAM, NNODE, JSEMI)

and the subroutine returns with the values NAM = 25, NEQ = 43, NNODE = 9, JSEMI = 23 and the array IDENT which contains



which states for example, that the variation associated with the generalised displacement $\partial w/\partial n$ at mid-point 2 occupies row 4 in the square global stiffness matrix, or that the variations associated with w, $\partial w/\partial x$, $\partial w/\partial y$ at node 3 occupy rows 5, 6 and 7 respectively.

4.3 Subroutine MAT ATBA

The subroutine evaluates the matrix equation

$$[C] = [A]^T[B][A]$$

where $[A]$ is a m by m rectangular matrix stored in the calling segment in the real array A(MA,NA) and $[B]$ is a m by m symmetric matrix stored in the real array B(MA,MA). The subroutine finally multiplies the n by n symmetric matrix product $[A]^T[B][A]$ by the given real number CONSTANT before returning the result to the real array C(NA,NA). Note that the arrays A,B,C, have the same dimensions as the corresponding matrices $[A]$,$[B]$,$[C]$, and that, although symmetric, the complete matrix $[B]$ is required. The subroutine is used to perform a matrix operation which is frequently required in setting up the element stiffness matrices. The matrices involved are of a constant size for a given type of element so that no allowance is made in dimensioning the arrays for variations between different data cases. The calling statement is

CALL MAT ATBA (A, MANA, MA, B, MAMA, C, NANA, CONSTANT)

As an example consider equation (2) where it is required to evaluate the matrix product $[C]^T[D][C]$. The three by three symmetric elasticity matrix $[D]$ is stored in the array D(3,3), $[C]$ is a three by six transformation matrix stored in the array C(3,6) and the result is to be stored in the six by six array CTDC(6,6). Assuming that CONSTANT in this case is 2.3 then the calling statement is

CALL MAT ATBA(C, 18, 3, D, 9, CTDC, 36, 2.3) .

4.4 Subroutine INTEGRAL MAT ATBA

The subroutine evaluates the equation

$$[C] = \int_{\text{element area}} \int [A]^T[B][A] \, dx \, dy$$

where $[A]$ is a m by n rectangular matrix, $[B]$ is a m by m symmetric matrix of constants and $[C]$ is a n by n matrix. The elements of the matrix $[A]$ are functions of x and y, e.g. x, y, xy (or of area coordinates[2] like L_1, L_2, $L_2 L_3$) and hence [A] can be written as

$$[A] = [A_1]f_1(x,y) + [A_2]f_2(x,y) + \ldots\ldots[A_i]f_i(x,y) + \ldots\ldots[A_K]f_K(x,y)$$

where the $[A_i]$ are m by n matrices of constants and K is an integer equal to the total number of different functions. An integer NTYPES denotes the largest number of $f_i(x,y)$ required to define an element of $[A]$. In the calling segment the real array A(MANA, NTYPES) holds the amplitudes of the $f_i(x,y)$ so that these components of the element a_{ij} in $[A]$ are stored in A(i',1), A(i',2),..... A(i',NTYPES) with i' = (i-1)*NA+j. The integer array ATYPE(MANA, NTYPES) holds type numbers (integers 1,2.....K) stored in the same way, which identify the $f_i(x,y)$ as they occur in A(MANA,NTYPES), so that type number 1 indicates function type $f_1(x,y)$ and so on. The type numbers must be in ascending numerical order but followed by any zeros. The array A must be arranged in sympathy with ATYPE. Within the subroutine the arrays A and ATYPE are declared as one-dimensional arrays A(MANANTYPES) and ATYPE(MANANTYPES) where MANANTYPES = MA*NA*NTYPES.

In the calling segment the square matrix $[B]$ is a matrix of constants stored in the real array B(MA,MA) and the result $[C]$ is stored in the real array C(NA,NA). The real array INT(K,K) contains the symmetric matrix of

integrals over the element area so that the i,jth element holds the area integral of $f_i(x,y)f_j(x,y)$ (i,j = 1,2.....K). Within the subroutine INT is declared as a one-dimensional array INT(KK) where KK = K*K. The subroutine uses two one-dimensional real arrays SPARE1(100) and SPARE2(100) as working space, but if MANA is greater than 100 these arrays must be re-dimensioned accordingly.

The subroutine is used in setting up the element stiffness matrices and the calling statement is

CALL INTEGRAL MAT ATBA(A, MANANTYPES, B, MAMA, C, NANA, INT, KK, ATYPE).

Since the matrices involved are again a constant size the corresponding arrays are dimensioned accordingly.

To illustrate the use of the subroutine consider the case where [A] is the matrix

$$\begin{bmatrix} 0.0 & 2.5 + 3.2x & 0.0 & 0.0 & 4.1 \\ 2.1 - 0.9x + 1.7y & 0.0 & 6.3x + 1.2y & 2.4y & 5.3x \\ 0.0 & 0.0 & 0.0 & 4.2 - 2.7x & -2.6y \end{bmatrix}$$

Here there are three $f_i(x,y)$ namely $f_1(x,y)$ = constant, $f_2(x,y)$ = x and $f_3(x,y)$ = y and so K = 3. All three types are required to specify element $a_{2,1}$ and so NTYPES = 3. Since MA = m = 3 and NA = n = 5 the arrays A and ATYPE are as follows

$$A(15,3) = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 2.5 & 3.2 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 4.1 & 0.0 & 0.0 \\ 2.1 & -0.9 & 1.7 \\ 0.0 & 0.0 & 0.0 \\ 6.3 & 1.2 & 0.0 \\ 2.4 & 0.0 & 0.0 \\ 5.3 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 4.2 & -2.7 & 0.0 \\ -2.6 & 0.0 & 0.0 \end{bmatrix} \qquad ATYPE(15,3) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 2 & 3 \\ 0 & 0 & 0 \\ 2 & 3 & 0 \\ 3 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 2 & 0 \\ 3 & 0 & 0 \end{bmatrix}$$

Thus the second rows (say) indicate that element $a_{1,2}$ of the matrix [A] consists of two terms, one of function type 1 with coefficient 2.5 (i.e. 2.5) and one of function type 2 with coefficient 3.2 (i.e. 3.2x). In short, each row of array A and the corresponding row of array ATYPE, define one element of the matrix [A].

The matrix INT is the three by three matrix of area integrals

$$\text{INT}(3,3) = \begin{bmatrix} \iint \text{const}\ dx\ dy & \iint x\ dx\ dy & \iint y\ dx\ dy \\ \iint x\ dx\ dy & \iint x^2\ dx\ dy & \iint xy\ dx\ dy \\ \iint y\ dx\ dy & \iint xy\ dx\ dy & \iint y^2\ dx\ dy \end{bmatrix}$$

and B is some three by three array of constants. The calling statement is

```
CALL INTEGRAL MAT ATBA (A, 45, B, 9, C, 25, INT, 9, ATYPE)
```

and the result is returned by the real array C(5,5).

4.5 Subroutine ADD TO BANDMAT

The purpose of this subroutine is to form a global stiffness matrix [K], see equation (3), from the individual stiffness matrices [k]. The subroutine adds the elements of the m by m symmetric matrix [B] into positions in the banded symmetric matrix [A] as specified by the one-dimensional integer array IPOSITION. In the calling segment the matrices [B] and [A] are stored respectively in the real arrays B(IB, JB) and A(IA, JA) as already described, while the integer array IPOSITION is dimensioned IB where IB≥MB. Note that only the upper triangle of [B] is strictly required. If the prescribed semi-bandwidth JSEMI is insufficient an error message to this effect is output and control returned to the calling segment with JSEMI set equal to zero. Because of the symmetry of the matrices [B] and [A], rows and columns of [B] are related to rows and columns of [A] by a single one-dimensional array IPOSITION. Thus, IPOSITION(i) = p indicates that row (column) i in [B] corresponds to row (column) p in [A]. A negative value of p is taken to indicate that the sign of both the ith row and column of [B] is to be changed before the addition takes place. The value p = o indicates that the ith row and column of [B] contribute nothing to [A]. The subroutine is called by the statement

```
CALL ADD TO BANDMAT (A, IAJSEMI, IA, B, IBJB, IB, MB, IPOSITION)
```

As an illustrative example consider the triangle of Fig.1 which is denoted by node and mid-point numbers 1, 2, 3, 8, 13, 9 taken in the clockwise direction. It is assumed that the deflection w, and its derivatives $\partial w/\partial x$ and $\partial w/\partial y$ are the generalised displacements at the nodes and that the normal slope $\partial w/\partial n$ is the generalised displacement at the mid-points. For compatibility of normal slope between elements the convention is now adopted that if the second node number in the clockwise direction along one side of the element is greater than the first node number on that side then the sign of the directional derivative $\partial w/\partial n$ at that mid-point is to be changed. Assume that a 12 by 12 element stiffness matrix [B] is stored in the array B(15, 20), and suppose that the banded symmetric global stiffness matrix [A] is dimensioned A(45, 23). Then, for this element, the integer array IPOSITION(12) contains {1 2 3 4 5 6 7 14 21 22 23 -15}, the minus sign indicating that the signs of the 12th row and column of [B] are to be changed before adding. The calling statement

CALL ADD TO BANDMAT (A, 1035, 45, B, 300, 15, 12, IPOSITION)

then returns the array A with, for example, B(1, 1) added to A(1, 1), B(1, 2) to A(1, 2), ... -B(8, 12) to A(14, 2), ... -(-B(12, 12)) to A(15, 1).

4.6 Subroutine ADD TO VECTOR

This subroutine is used to form the global column vector {Q} of equation (3) from the contributions of each element in a similar way to that used in subroutine ADD TO BANDMAT to form the global stiffness matrix. The subroutine multiplies the column vector {B} by a given real number CONSTANT and adds the result into positions in the column vector {A} as specified by the integer array IPOSITION. In the calling segment the column vectors {B} and {A} are stored respectively in the one-dimensional real arrays B(IB) and A(IA) where IB≥MB and IA≥MA. The array IPOSITION is the same as that used in ADD TO BANDMAT and again if any element of IPOSITION is negative the sign of the corresponding element of {B} is changed before adding, also a zero value in IPOSITION indicates a zero contribution to {A}. The calling statement for the subroutine is

CALL ADD TO VECTOR (A, MA, B, MB, IPOSITION, CONSTANT)

To illustrate, consider the previous example (section 4.5) where in this case [A] and [B] are column vectors, then the calling statement

CALL ADD TO VECTOR (A, 43, B, 12, IPOSITION, 1.0)

adds, for example, B(1) to A(1), B(2 to A(2),...B(8) to A(14), ... -B(12) to A(15).

4.7 Subroutine ALTER BANDMAT

Having derived the global matrices [K] and {Q} of equation (3) it is now necessary to impose the boundary conditions. Those considered here take the form of explicit constraints involving only a single variable at the nodes or mid-points, for example, w = CONST at node i, and it is required to modify [K] and {Q} accordingly. Other types of boundary conditions are considered in section 4.12. Thus, in the subroutine, given a row (column) number I (an integer), the Ith column of the symmetric banded matrix [A] is multiplied by a given real number CONST and subtracted from the column vector {B}. In the calling segment the matrix [A] is stored in the real array A(IA,JSEMI) as already described and the column vector {B} in the real array B(IB) where IB≥MB. The Ith row and column of [A] are then replaced by zeros except for the leading diagonal element which is given the value 1.0 and the Ith element of {B} is replaced by CONST. The calling statement for the subroutine is

CALL ALTER BANDMAT (A, IAJSEMI, IA, B, MB, I, CONST) .

To illustrate, consider the following simple example. Let the symmetric banded matrix

$$[A] = \begin{bmatrix} 1.9 & 2.1 & -5.7 & 0.0 & 0.0 \\ 2.1 & 3.4 & 1.5 & 3.3 & 0.0 \\ -5.7 & 1.5 & 2.2 & 4.5 & 2.8 \\ 0.0 & 3.3 & 4.5 & 5.6 & -1.8 \\ 0.0 & 0.0 & 2.8 & -1.8 & 4.7 \end{bmatrix}$$

be stored in the real array A(5, 3) as

$$A(5,3) = \begin{bmatrix} 1.9 & 2.1 & -5.7 \\ 3.4 & 1.5 & 3.3 \\ 2.2 & 4.5 & 2.8 \\ 5.6 & -1.8 & 0.0 \\ 4.7 & 0.0 & 0.0 \end{bmatrix} .$$

If initially {B}, stored in the real array B(5), is zero, I = 3 and CONST = 2.0 then the calling statement

CALL ALTER BANDMAT (A, 15, 5, B, 5, 3, 2.0)

returns the arrays A and B with the following values

$$A(5,3) = \begin{bmatrix} 1.9 & 2.1 & 0.0 \\ 3.4 & 0.0 & 3.3 \\ 1.0 & 0.0 & 0.0 \\ 5.6 & -1.8 & 0.0 \\ 4.7 & 0.0 & 0.0 \end{bmatrix}, \qquad B(5) = \begin{bmatrix} 11.4 \\ -3.0 \\ 2.0 \\ -9.0 \\ -5.6 \end{bmatrix}.$$

4.8 Subroutine SQUARE BANDMAT

Given a symmetric banded matrix [A] with semi-bandwidth JSEMI, stored in the calling segment in the real array A(IA, JA) as already described, the subroutine evaluates $[A]^2$ which is returned in the array A thereby overwriting the original matrix. The squaring of a matrix of semi-bandwidth JSEMI results in a matrix of semi-bandwidth JSEMI2 = 2*JSEMI-1 so it is important to note that in this case it is necessary to set JA≥JSEMI2. The subroutine is used to form a positive definite symmetric banded matrix from a non-singular symmetric banded matrix and the calling statement is

CALL SQUARE BANDMAT (A, IAJSEMI2, IA, JSEMI, MA)

where IAJSEMI2 = IA*JSEMI2.

As an illustrative example let

$$[A] = \begin{bmatrix} 1.0 & 2.0 & & & & & & & & \\ 2.0 & 2.0 & 3.0 & & & & & & & \\ & 3.0 & 3.0 & 4.0 & & & & & & \\ & & 4.0 & 4.0 & 5.0 & & & & & \\ & & & 5.0 & 5.0 & 6.0 & & & & \\ & & & & 6.0 & 6.0 & 7.0 & & & \\ & & & & & 7.0 & 7.0 & 8.0 & & \\ & 0 & & & & & 8.0 & 8.0 & 9.0 & \\ & & & & & & & 9.0 & 9.0 & 10.0 \\ & & & & & & & & 10.0 & 10.0 \end{bmatrix}$$

so that JSEMI = 2 and hence JSEMI2 = 3. The matrix [A] is stored in the array A as follows

$$A(10,3) = \begin{bmatrix} 1.0 & 2.0 & 0.0 \\ 2.0 & 3.0 & 0.0 \\ 3.0 & 4.0 & 0.0 \\ 4.0 & 5.0 & 0.0 \\ 5.0 & 6.0 & 0.0 \\ 6.0 & 7.0 & 0.0 \\ 7.0 & 8.0 & 0.0 \\ 8.0 & 9.0 & 0.0 \\ 9.0 & 10.0 & 0.0 \\ 10.0 & 0.0 & 0.0 \end{bmatrix}$$

where here IA = 10, JA = 3. The calling statement

CALL SQUARE BANDMAT (A, 30, 10, 2, 10)

returns the array A so that it contains the upper half band of $[A]^2$ as follows

$$A(10,3) = \begin{bmatrix} 5.0 & 6.0 & 6.0 \\ 17.0 & 15.0 & 12.0 \\ 34.0 & 28.0 & 20.0 \\ 57.0 & 45.0 & 30.0 \\ 86.0 & 66.0 & 42.0 \\ 121.0 & 91.0 & 56.0 \\ 162.0 & 120.0 & 72.0 \\ 209.0 & 153.0 & 90.0 \\ 262.0 & 190.0 & 0.0 \\ 200.0 & 0.0 & 0.0 \end{bmatrix}$$

4.9 Subroutine MULT BANDMAT VECTOR

The subroutine post-multiplies the symmetric banded matrix [A] by a column vector {B} where, in the calling segment, [A] is stored in the real array A(IA, JA) and {B} in the real array B(IB). The array B is overwritten by the result. Within the subroutine the real array SPARE(100) is used as working space and must be re-dimensioned accordingly if MB>100. The calling statement is

CALL MULT BANDMAT VECTOR (A, IAJSEMI, IA, B, MB).

As an example consider the array A(10,3) given in section 4.8, then if B(10) is the array {1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0} the calling statement

CALL MULT BANDMAT VECTOR (A, 30, 10, B, 10)

post-multiplies the array A by array B and returns the result through B as follows B(10) = {5.0 15.0 31.0 53.0 81.0 115.0 155.0 201.0 253.0 190.0}.

4.10 Subroutine POSDEF MATINV

The subroutine finds the inverse of the positive definite symmetric matrix [A] stored in the calling segment in the real array A(MA, MA). The method used is that of Choleski in which [A] is expressed as the product of an upper and a lower triangular matrix where, because of symmetry, the lower is the transpose of the upper. The inverse is returned in the array A thereby overwriting the original matrix. Although [A] is symmetric it is assumed that the full matrix is supplied in the array A. The calling statement is

CALL POSDEF MATINV (A, MAMA).

To illustrate, the calling statement

CALL POSDEF MATINV (A, 144)

overwrites the 12 by 12 symmetric positive definite matrix [A] stored in the real array A(12, 12) by its inverse $[A]^{-1}$.

It should be noted that the inversion of a symmetric positive definite matrix is now available in the ICL Scientific Subroutines Library[5]. This library subroutine, written in PLAN uses the Gaussian elimination method and requires only the lower triangle of [A] stored as a one-dimensional array; it is found to be up to six times faster than the present subroutine which is therefore included only to complete the package.

4.11 Subroutine SOLVE BANDMAT

This subroutine is used to solve the banded system of simultaneous equations given by equation (3) to obtain the unknown generalised displacements. The method used is due to Martin and Wilkinson[6] and is most efficient if the bandwidth is small compared with the order of the matrix. The subroutine solves the matrix equation

$$[A][X] = [B]$$

where [A] is a symmetric positive definite banded matrix and [B] is a matrix of right hand sides. In the calling segment [B] is stored in the real array B(IA, JB) where JB is the number of right hand sides, if JB = 1 then [B] can be stored in the one-dimensional array B(IA). The solution matrix is

returned in B thereby overwriting the original matrix. An integer NCHANGE is also required to be prescribed, its value depending on the way in which [A] is stored. If the upper half band of [A] is stored as already described then NCHANGE can be any integer greater than zero, if the lower half band is stored then NCHANGE must be set equal to zero. During the solution the original matrix [A] is destroyed. If [A] is singular, or not positive definite, the subroutine outputs an error message FAILED IN INVERSION and control is returned to the calling segment with A undefined. The calling statement is

CALL SOLVE BANDMAT (A, IAJSEMI, IA, MA, B, IAJB, NCHANGE).

To illustrate, if [A] is a 20 by 20 symmetric positive definite matrix with semi-bandwidth 5, the upper half band of which is stored in the array A(30, 5) and [B] is a column of right hand sides in the array B(30) then the calling statement

CALL SOLVE BANDMAT (A, 150, 30, 20, B, 30, 1)

returns the solution through the array B(30).

A more general subroutine is available in the ICL Scientific Subroutines Library for the solution of banded simultaneous equations but, because of its generality, it requires the whole band to be stored.

4.12 Subroutine SOLVE CONSTRAINED BANDMAT

This subroutine is used to solve the banded system of simultaneous equations given by equation (3) when the solution is subject to linear constraints which are provided by the boundary conditions and are of a more complicated form than those which can be applied with subroutine ALTER BANDMAT. The method which is used to apply the constraints without increasing the bandwidth or losing the symmetry of [K] is due to Morley[7] and the subroutine SOLVE BANDMAT is called for the solution of the final equations. More explicitly, when the matrix equation

$$[A]\{X\} = \{B\}$$

is derived by way of a variational process, as in equation (3), the subroutine obtains the solution for {X} subject to the imposition of linear constraints

$$[C]\{X\} = \{D\} \quad .$$

The matrix [A] is a symmetric positive definite banded matrix and is stored in the calling segment in the real array A(IA,JA), {B} is a column vector stored in the real array B(IA), {D} is a column vector stored in the real

array D(IA) and [C] is a tridiagonal matrix consisting of unit diagonal elements and/or square submatrices of order two of the form

$$[C] = \begin{bmatrix} 1.0 & & & & & \\ & 1.0 & & & & \\ & & \ddots & & & \\ & & & \begin{bmatrix} C_{i,i} & C_{i,i+1} \\ C_{i+1,i} & C_{i+1,i+1} \end{bmatrix} & & \\ & & & & \ddots & \\ & & & & & \begin{bmatrix} C_{j,j} & C_{j,j+1} \\ C_{j+1,j} & C_{j+1,j+1} \end{bmatrix} \\ & & & & & 1.0 \\ & & & & & \ddots \end{bmatrix}$$

The square submatrices are stored in the real array C(IA, 2) as follows

$$C(IA,2) = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \\ \vdots & \vdots \\ C_{i,1} & C_{i,2} \\ C_{i+1,1} & C_{i+1,2} \\ \vdots & \vdots \\ C_{j,1} & C_{j,2} \\ C_{j+1,1} & C_{j+1,2} \\ 0.0 & 0.0 \\ \vdots & \vdots \\ \vdots & \vdots \\ \vdots & \vdots \end{bmatrix} \quad \updownarrow \; IA$$

The subroutine requires an accompanying integer array IDENT(NAM) described in section 4.2 and an integer array CONROW(IA) which is filled as follows

CONROW(I) = 0 if row I is not a constraint row

CONROW(I) = 1 if row I is an isolated constraint row, which involves only one variable

CONROW(I) = 2 if row I is a constraint row forming a submatrix with row I+1, i.e. involves two variables

If both rows I and I+1 forming a submatrix are constraint rows then CONROW(I) = CONROW(I+1) = 2. It should be noted that the one-dimensional array SPARE(200) is used as additional working space and must be re-dimensioned accordingly if MA>200. The calling statement is

CALL SOLVE CONSTRAINED BANDMAT (A, B, C, D, IA, JSEMI, CONROW, IDENT, MA, NAM).

To illustrate the use of the subroutine consider the simple case of a single triangle with nodes and mid-points numbered as shown in Fig.3. The kinematic boundary conditions for two sides are also shown where w is the out of plane deflection and $\partial w/\partial n$ is the normal slope. The generalised displacements at a node are $w, \partial w/\partial x, \partial w/\partial y$ and at a mid-point $\partial w/\partial n$. The angle γ_{64} for example, is the angle included by the intersection of the outward pointing normal n to the side joining nodes 6 and 4 with the Ox axis. The distance around the boundary of the element is measured in the clockwise sense by s. The integer array IDENT for this case contains {1 4 5 6 9 10}.

To satisfy the boundary conditions along side 6-4, i.e. $w = 0$, the values of w and the slope $\partial w/\partial s$ are prescribed at each end. The condition $w = 0$ can be prescribed directly at nodes 6 and 4 but the slopes $\partial w/\partial s$ are obtained from

$$\frac{\partial w}{\partial s} = -\sin \gamma \frac{\partial w}{\partial x} + \cos \gamma \frac{\partial w}{\partial y} \quad .$$

Consider node 6 (noting that IDENT(6) = 10): the condition $w_6 = 0$ is directly prescribed by setting CONROW(10) = 1 and D(10) = 0.0, while the additional condition $\partial w_6/\partial s = 0$, which is given by the equation

$$0.0 = -\sin \gamma_{64} \frac{\partial w_6}{\partial x} + \cos \gamma_{64} \frac{\partial w_6}{\partial y}$$

can be put in either row 11 or row 12 of [C]. The convention adopted here is that the first of the two rows is used unless it has been previously filled, so that, in this case we set CONROW(11) = 2, D(11) = 0.0, C(11,1) = $-\sin \gamma_{64}$ and C(11,2) = $\cos \gamma_{64}$. Similarly at node 4, where IDENT(4) = 6, we set CONROW(6) = 1 and D(6) = 0.0 to prescribe $w_4 = 0.0$ and CONROW(7) = 2, D(7) = 0.0, C(7,1) = $-\sin \gamma_{64}$, C(7,2) = $\cos \gamma_{64}$ to prescribe $\partial w_4/\partial s = 0$.

To satisfy the boundary condition along side 4-1, i.e. $\partial w/\partial n = 0$, the value of $\partial w/\partial n$ is prescribed at the points 4,2 and 1. At the mid-point, 2, $\partial w_2/\partial n = 0$ is directly prescribed by setting CONROW(4) = 1 and D(4) = 0.0 but at nodes 4 and 1 the $\partial w/\partial n$ is given by the equation

$$\frac{\partial w}{\partial n} = \cos \gamma \frac{\partial w}{\partial x} + \sin \gamma \frac{\partial w}{\partial y} \quad .$$

Hence at node 4 the equation

$$0.0 = \cos \gamma_{41} \frac{\partial w_4}{\partial x} + \sin \gamma_{41} \frac{\partial w_4}{\partial y}$$

is put in row 8 (row 7 is already filled) so that CONROW(8) = 2, D(8) = 0.0, C(8,1) = $\cos \gamma_{41}$ and C(8,2) = $\sin \gamma_{41}$. Similarly at node 1 where IDENT(1) = 1, CONROW(2) = 2, D(2) = 0.0, C(2,1) = $\cos \gamma_{41}$ and C(2,2) = $\sin \gamma_{41}$.

In this case, therefore, the constraint equations can be written

$$\begin{bmatrix}
1.0 & & & & & & & & & & & \\
\cos \gamma_{41} & \sin \gamma_{41} & & & & & & & & & & \\
0.0 & 1.0 & & & & & & & & & & \\
 & & 1.0 & & & & & & & & & \\
 & & & 1.0 & & & & & & & & \\
 & & & & 1.0 & & & & & & & \\
 & & & & & -\sin \gamma_{64} & \cos \gamma_{64} & & & & & \\
 & & & & & \cos \gamma_{41} & \sin \gamma_{41} & & & & & \\
 & & & & & & & 1.0 & & & & \\
 & & & & & & & & 1.0 & & & \\
 & & & & & & & & & -\sin \gamma_{64} & \cos \gamma_{64} & \\
 & & & & & & & & & 0.0 & 1.0 &
\end{bmatrix}
\begin{bmatrix} w_1 \\ \partial w_1/\partial x \\ \partial w_1/\partial y \\ \partial w_2/\partial n \\ \partial w_3/\partial n \\ w_4 \\ \partial w_4/\partial x \\ \partial w_4/\partial y \\ \partial w_5/\partial n \\ w_6 \\ \partial w_6/\partial x \\ \partial w_6/\partial y \end{bmatrix}
=
\begin{bmatrix} w_1 \\ 0.0 \\ \partial w_1/\partial y \\ 0.0 \\ \partial w_3/\partial n \\ 0.0 \\ 0.0 \\ 0.0 \\ \partial w_5/\partial n \\ 0.0 \\ 0.0 \\ \partial w_6/\partial y \end{bmatrix}$$

so that the arrays C and CONROW are as follows

$$C(12) = \begin{bmatrix} 0.0 & 0.0 \\ \cos\gamma_{41} & \sin\gamma_{41} \\ 0.0 & 1.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \\ -\sin\gamma_{64} & \cos\gamma_{64} \\ \cos\gamma_{41} & \sin\gamma_{41} \\ 0.0 & 0.0 \\ 0.0 & 0.0 \\ -\sin\gamma_{64} & \cos\gamma_{64} \\ 0.0 & 1.0 \end{bmatrix} \qquad CONROW(12) = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 1 \\ 0 \\ 1 \\ 2 \\ 2 \\ 0 \\ 1 \\ 2 \\ 0 \end{bmatrix}$$

Then given an array A(12,12) containing the global stiffness matrix and an array B(12) containing the right hand sides the calling statement

```
CALL SOLVE CONSTRAINED BANDMAT (A, B, C, D, 12, 12, CONROW, IDENT, 12, 6)
```

returns the unknown displacements through the array B.

5 CONCLUSIONS

The subroutines which are described provide building blocks which enable finite element programs to be quickly assembled and tested taking full advantage of the symmetry and bandedness of the matrices. The subroutines can be divided into two main groups, those concerned with operations on rectangular or square matrices and those concerned with operations on banded matrices. The former group may be used to set up the element stiffness matrices while the latter serve to set up and solve the global equations.

REFERENCES

No.	Author(s)	Title, etc.
1	A. W. Odell	Execution times on the RAE 1907 computer. RAE Technical Memorandum Space 120 (1969)
2	O. C. Zienkiewicz Y. K. Cheung	The finite element method in structural and continuum mechanics. McGraw Hill (1967)
3	F. A. Akyuz S. Utku	An automatic node-relabelling scheme for bandwidth minimisation of stiffness matrices. AIAA Journal 6, 4, 728-730, (1968)
4	L. S. D. Morley B. C. Merrifield	Bandwidth reduction of label sequences as encountered in finite element calculations. RAE Technical Memorandum Structures 789 (1970)
5	ICL	Scientific subroutines. ICL Technical Publications 4096 First edition May 1968
6	R. S. Martin J. H. Wilkinson	Symmetric decomposition of positive definite band matrices. Numerische Mathematik 7, 355-361 (1965)
7	L. S. D. Morley	Bending of a square plate with central square hole. RAE Technical Report 69031 (1969)

App.A

LISTING OF SUBROUTINES

A.1 FE INPUT

```
      SUBROUTINE FE INPUT(NEL,QO,NCONC,CNODE,CMAG,ELNO,X,Y,IEL,JEL,
     1IDENT,ID,NF,MF,NEQ,NAM,NNODE,JSEMI,NU,D,IXY,ICON)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  THE SUBROUTINE READS AND PRINTS BASIC FINITE ELEMENT DATA AND CALLS
C  SUBROUTINE FE PARAMETERS TO CALCULATE THE ADDITIONAL PARAMETERS
C  REQUIRED WHICH ARE ALSO PRINTED.
C  THE SUBROUTINE READS-  CAPTION,NEL,NCONC,NU,QO,D,CNODE,CMAG,ELNO,X,Y.
C  PRINTS- CAPTION,NEL,NNODE,NAM,NEQ,JSEMI,NU,D,QO,CNODE,CMAG,ELNO,X,Y.
C  WHERE:
C  CAPTION IS A SINGLE CARD WITH ANY REQD. TITLE IN COLS 1-72.
C  NEL=NUMBER OF ELEMENTS,              NCONC=NUMBER OF CONCENTRATED LOADS.
C   NU=POISSONS RATIO,                    QO=INTENSITY OF UD LOAD.
C    D=FLEXURAL RIGIDITY,               NNODE=NUMBER OF NODES.
C  NAM=NUMBER OF NODES AND MIDPOINTS,   NEQ=NUMBER OF EQUATIONS.
C  JSEMI=SEMI-BANDWIDTH OF GLOBAL STIFFNESS MATRIX.
C  CNODE=INTEGER ARRAY CONTAINING NODE NOS. AT WHICH CONC. LOADS APPLIED.
C  CMAG =REAL ARRAY CONTAINING MAGNITUDES OF CONC. LOADS.
C  ELNO =INTEGER ARRAY CONTAINING THE NODE AND MIDPOINT NUMBERS TAKEN IN
C  SEQUENCE AROUND EACH ELEMENT STARTING AT A NODE(ONE ELEMENT TO A ROW).
C  X,Y  =REAL ARRAYS CONTAINING X,Y COORDINATES OF NODES.
C  IDENT=INTEGER ARRAY RELATING NODE AND MIDPOINT NUMBERS TO POSITIONS
C  IN THE GLOBAL STIFFNESS MATRIX. (NOTE, IDENT IS NOT PRINTED)
C     THE ARRAYS ARE DIMENSIONED:
C     CNODE(ICON),CMAG(ICON),ELNO(IEL,JEL),X(IXY),Y(IXY),IDENT(ID),
C     WHERE: ICON>OR=NCONC>OR=1,IXY>OR=NAM,IEL>OR=NEL,ID>OR=NAM,
C     AND JEL=NUMBER OF NODES AND MIDPOINTS REQUIRED TO DESCRIBE
C     ONE ELEMENT.
C  IN ADDITION NF=NUMBER OF DEGREES OF FREEDOM AT A NODE,
C           AND MF=NUMBER OF DEGREES OF FREEDOM AT A MIDPOINT.
C  NB, IF MF<0 A SPECIAL FE PARAMETERS SUBROUTINE IS REQUIRED.
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      REAL CAPTION(9),X(IXY),Y(IXY),CMAG(ICON),NU
      INTEGER ELNO(IEL,JEL),IDENT(ID),CNODE(ICON)
      READ(1,10)CAPTION
      READ(1,20)NEL,NCONC,NU,QO,D
      IF(NCONC.EQ.0)GOTO 8
      DO 1 I=1,NCONC
    1 READ(1,30)CNODE(I),CMAG(I)
    8 J1=JEL
      J2=1
      IF(MF)0,3,3
      J1=JEL+MF+1
      J2=-MF
    3 DO 2 I=1,NEL
    2 READ(1,40)(ELNO(I,J),J=1,J1,J2)
      CALL FE PARAMETERS(NEL,ELNO,IEL,JEL,IDENT,ID,NF,MF,NEQ,NAM,NNODE,
     1JSEMI)
      WRITE(2,50)CAPTION
      WRITE(2,60)
      WRITE(2,70)NEL,NNODE,NAM,NEQ,JSEMI,NU,D
      WRITE(2,80)QO
      IF(NCONC.EQ.0)GOTO 9
      WRITE(2,90)
      DO 4 I=1,NCONC
    4 WRITE(2,100)CNODE(I),CMAG(I)
```

App.A (cont'd)

```
      9 WRITE(2,110)
        J=NNODE
        IF(NNODE.LT.NEL)J=NEL
        DO 5 I=1,J
        IF(I.GT.NEL)GOTO 6
        WRITE(2,120)(ELNO(I,K),K=1,JEL)
        IF(I.GT.NNODE)GOTO 5
        GOTO 7
      6 WRITE(2,130)
      7 READ(1,30)NODE,X(NODE),Y(NODE)
        WRITE(2,140)NODE,X(NODE),Y(NODE)
      5 CONTINUE
        RETURN
C
     10 FORMAT(9A8)
     20 FORMAT(2I0,3F0.0)
     30 FORMAT(I0,2F0.0)
     40 FORMAT(10I0)
     50 FORMAT(1H1///20X,9A8)
     60 FORMAT(1H0,40X,14HDATA PRINT OUT)
     70 FORMAT(1H0,19HNUMBER OF ELEMENTS=,15X,I5/17H NUMBER OF NODES=,18X
       1,I5/31H NUMBER OF NODES AND MIDPOINTS=,4X,I5/
       221H NUMBER OF EQUATIONS=,14X,I5/
       335H SEMIBANDWIDTH OF STIFFNESS MATRIX=,I5/16H POISSONS RATIO=,18X
       4,F6.4/19H FLEXURAL RIGIDITY=,9X,F12.4)
     80 FORMAT(1H0,22H INTENSITY OF U.D.LOAD=,5X,F12.4)
     90 FORMAT(1H0,17HCONCENTRATED LOAD/17H NODE   MAGNITUDE)
    100 FORMAT(1H ,I3,F12.6)
    110 FORMAT(1H0,16X,4HELNO,60X,4HNODE,6X,1HX,8X,1HY)
    120 FORMAT(1H ,1216)
    130 FORMAT(1H )
    140 FORMAT(1H+,77X,I5,2X,2F10.6)
        END
```

A.2 FE PARAMETERS

```
      SUBROUTINE FE PARAMETERS(NEL,ELNO,IEL,JEL,IDENT,ID,NF,MF,NEQ,NAM,
     1NNODE,JSEMI)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  THE SUBROUTINE CALCULATES NEQ,NAM,NNODE,JSEMI,AND IDENT(ID).
C  THE SUBROUTINE REQUIRES:
C     THE INTEGER ARRAY ELNO(IEL,JEL) EACH ROW CONTAINING THE NODE AND
C     MIDPOINT NUMBERS IN SEQUENCE AROUND ONE ELEMENT STARTING AT A NODE.
C     THE INTEGERS NEL,NF,MF.(NOTE  NF,MF >OR= 0)
C     THE ARRAY DIMENSIONS (INTEGERS) ID,IEL,JEL.
C  THE NOTATION IS THAT OF SUBROUTINE FE INPUT.
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      INTEGER ELNO(IEL,JEL),IDENT(ID),DIFF
      K,NAM=0
      DO 2 I=1,NEL
      DO 2 J=1,JEL
    2 IF(ELNO(I,J).GT.NAM)NAM=ELNO(I,J)
C
      NNODE=0
      IDENT(1)=1
      DO 1 I1=1,NAM
      DO 13 I2=1,NEL
      I3=1
    6 IF(NF.EQ.0)GOTO 4
```

App.A (cont'd)

```
      IF(I1.NE.ELNO(I2,I3))GOTO 5
      NNODE=NNODE+1
      IF(I1.NE.NAM)IDENT(I1+1)=IDENT(I1)+NF
      GOTO 1
    5 I3=I3+1
      IF(MF.EQ.0)GOTO 12
    4 IF(I1.NE.ELNO(I2,I3))GOTO 14
      IF(I1.NE.NAM)IDENT(I1+1)=IDENT(I1)+MF
      GOTO 1
   14 I3=I3+1
   12 IF(I3.LE.JEL)GOTO 6
   13 CONTINUE
    1 CONTINUE
      IF(MF.EQ.0)NNODE=NAM
C
      NEQ=NF*NNODE+MF*(NAM-NNODE)
      JSEMI=0
      DO 7 I=1,NEL
      M1=0
      M2=NAM+1
      DO 8 J=1,JEL
      IF(ELNO(I,J).GT.M1)M1=ELNO(I,J)
    8 IF(ELNO(I,J).LT.M2)M2=ELNO(I,J)
      IF(MF.NE.0)GOTO9
      M3=IDENT(M1)+NF-1
      GOTO11
    9 J=1
      K=MF-1
   10 IF(M1.EQ.ELNO(I,J))K=NF-1
      J=J+2
      IF(J.LE.JEL)GOTO10
      M3=IDENT(M1)+K
   11 M4=IDENT(M2)
      DIFF=M3-M4+1
    7 IF(DIFF.GT.JSEMI)JSEMI=DIFF
      RETURN
      END
```

A.3 MAT ATBA

```
      SUBROUTINE MAT ATBA(A,MANA,MA,B,MAMA,C,NANA,CONSTANT)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  THE SUBROUTINE EVALUATES [C]=[A]TRANSPOSE*[B]*[A]*CONSTANT WHERE [A]
C  IS MA BY NA,[B] IS SYMMETRIC MA BY MA,AND [C] IS NA BY NA.
C  THE SUBROUTINE REQUIRES:
C     THE REAL ARRAY A(MA,NA) CONTAINING [A].
C     THE REAL ARRAY B(MA,MA) CONTAINING [B].
C     THE REAL ARRAY C(NA,NA) TO CONTAIN THE RESULT,[C].
C     A REAL NUMBER 'CONSTANT'.
C  NOTE, MANA=MA*NA,MAMA=MA*MA,NANA=NA*NA.
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      REAL A(MANA),B(MAMA),C(NANA)
      NA=0.1+MANA/MA
      I2=-MA
      I1=-NA
      DO 1 I=1,NA
      I1=I1+NA
      I2=I2+MA
      J2=I2-MA
      J1=I1-NA
```

App.A (cont'd)

```
      DO 1 J=I,NA
      J1=J1+NA
      J2=J2+MA
      C(J1+I)=0.0
      DO 1 K=1,MA
      SUML=0.0
      L1=-MA
      DO 2 L=1,MA
      L1=L1+MA
    2 SUML=SUML+A(J2+L)*B(L1+K)
    1 C(J1+I),C(I1+J)=C(J1+I)+A(I2+K)*SUML*CONSTANT
      RETURN
      END
```

A'.4 INTEGRAL MAT ATBA

```
      SUBROUTINE INTEGRAL MAT ATBA(A,MANANTYPES,B,MAMA,C,NANA,INT,KK,
     1ATYPE)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  THE SUBROUTINE EVALUATES THE INTEGRAL,OVER THE ELEMENT AREA,OF
C                    [A]TRANSPOSE*[B]*[A]
C  WHERE [A]=[A1]*F1(X,Y)+[A2]*F2(X,Y)+...[AI]*FI(X,Y)+...[AK]*FK(X,Y),
C  THE [AI] ARE MA BY NA MATRICES OF CONSTANTS AND [B] IS A MA BY MA
C  SYMMETRIC MATRIX OF CONSTANTS. THE SUBROUTINE REQUIRES:
C     THE REAL ARRAY B(MA,MA) CONTAINING THE MATRIX [B].
C     THE REAL ARRAY C(NA,NA) TO CONTAIN THE RESULT (SYMMETRICAL).
C     THE INTEGER NTYPES=THE LARGEST NUMBER OF FI(X,Y) REQUIRED TO
C     DEFINE AN ELEMENT OF THE MATRIX [A].
C     THE REAL ARRAY A(MANA,NTYPES) CONTAINING THE AMPLITUDES OF THE
C     FI(X,Y) STORED ROW BY ROW FROM [A]. THUS THE ELEMENT (I,J) OF [A]
C     IS STORED IN A(ID,1),A(ID,2),...A(ID,NTYPES) WITH ID=(I-1)*NA+J.
C     THE INTEGER ARRAY ATYPE(MANA,NTYPES) IDENTIFIES THE FI(X,Y) AS
C     THEY OCCUR IN A(MANA,NTYPES).THE TYPE NUMBERS IN ATYPE(ID,1),
C     ATYPE(ID,2),...ATYPE(ID,NTYPES) MUST BE IN NUMERICAL ORDER
C     FOLLOWED BY ANY ZEROS.
C     THE ARRAY A MUST BE ARRANGED IN SYMPATHY WITH THE ARRAY ATYPES.
C     THE REAL ARRAY INT(K,K) (SYMMETRIC) THE (I,J)TH ELEMENT OF WHICH
C     CONTAINS THE AREA INTEGRAL OF FI(X,Y)*FJ(X,Y).
C     THE INTEGER K=TOTAL NUMBER OF DIFFERENT FUNCTIONS FI(X,Y).
C  NB. MANANTYPES=MA*NA*NTYPES,NANA=NA*NA,MANA=MA*NA,MAMA=MA*MA,KK=K*K.
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  ---------------------------------------
C  SUBROUTINE FAILS FOR LACK OF STORAGE IN
C  SPARE1,SPARE2 IF MA*NA.GT.100
      REAL SPARE1(100),SPARE2(100)
C  ---------------------------------------
C
      REAL A(MANANTYPES),B(MAMA),C(NANA),INT(KK)
      INTEGER ATYPE(MANANTYPES),COL,ROW
C
      K=0.1+SQRT(1.0*KK)
      MA=0.1+SQRT(1.0*MAMA)
      NA=0.1+SQRT(1.0*NANA)
      MANA=MA*NA
      DO 1 I=1,NANA
    1 C(I)=0.0
      L1=MANANTYPES-MANA+1
      L2=MANA
```

App.A (cont'd)

```
      IF(L1.EQ.1)L2=1
C
      DO 2 L=1,K
C
C  DERIVE SPARE1(MA,NA)=AREA INTEGRAL OF FL(X,Y)*A(MA,NA).
C  SPARE1 STORES COLUMN BY COLUMN.
C
      J1=0
      DO 3 COL=1,NA
      ID=-NA
      DO 3 ROW=1,MA
      J1=J1+1
      ID=ID+NA
      SPARE1(J1)=0.0
      DO 4 JD=1,L1,L2
      J3=ID+JD+COL-1
      J=ATYPE(J3)
      IF(J)0,3,0
      J2=K*(J-1)+L
    4 SPARE1(J1)=SPARE1(J1)+A(J3)*INT(J2)
    3 CONTINUE
C
C  DERIVE SPARE2(MA,NA)=MATRIX MULTIPLICATION B(MA,NA)*SPARE1(MA,NA).
C  SPARE2 STORES ROW BY ROW.
C
      J2=0
      DO 5 ROW=1,MA
      J1=0
      DO 5 COL=1,NA
      J2=J2+1
      SPARE2(J2)=0.0
      J3=-MA
      DO 5 I=1,MA
      J3=J3+MA
      J1=J1+1
    5 SPARE2(J2)=SPARE2(J2)+B(J3+ROW)*SPARE1(J1)
C
C  DERIVE SPARE1(MA,NA)=AL(MA,NA).
C  SPARE1 STORES ROW BY ROW.
C
      DO 6 J1=1,MANA
      SPARE1(J1)=0.0
      DO 7 J2=1,L1,L2
      J3=J2
      IF(ATYPE(J1+J2-1)-L)7,8,6
    7 CONTINUE
      GOTO6
    8 SPARE1(J1)=A(J1+J3-1)
    6 CONTINUE
C
C  DERIVE ADDITION TO C(NA,NA)=SPARE1(MA,NA)TRANSPOSED*SPARE2(MA,NA).
C
      DO 9 ROW=1,NA
      DO 9 COL=ROW,NA
      CD=0.0
      J1=NA*(COL-1)+ROW
      J2=-NA
      DO 10 I=1,MA
      J2=J2+NA
   10 CD=CD+SPARE1(J2+ROW)*SPARE2(J2+COL)
    9 C(J1)=C(J1)+CD
    2 CONTINUE
```

App.A (cont'd)

```
C
C  SYMMETRICAL C(NA,NA).
C
      J1=-NA
      DO 11 I=1,NA
      J1=J1+NA
      J2=-NA
      DO 11 J=1,NA
      J2=J2+NA
   11 C(J1+J)=C(J2+I)
      RETURN
      END
```

A.5 ADD TO BANDMAT

```
      SUBROUTINE ADD TO BANDMAT(A,IAJSEMI,IA,B,IBJB,IB,MB,IPOSITION)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  THE SUBROUTINE ADDS THE ELEMENTS OF THE MB BY MB SYMMETRIC MATRIX [B]
C  INTO POSITIONS IN THE SYMMETRIC BANDED MATRIX [A] AS SPECIFIED BY THE
C  ELEMENTS OF THE INTEGER ARRAY IPOSITION. THE SUBROUTINE REQUIRES:
C     THE REAL ARRAY B(IB,JB) CONTAINING THE MATRIX [B] (IB,JB >OR= MB).
C     NOTE ONLY THE UPPER TRIANGLE OF [B] IS STRICTLY REQUIRED.
C     THE REAL ARRAY A(IA,JSEMI) CONTAINING THE UPPER HALF BAND OF THE
C     MATRIX [A].
C     THE INTEGER ARRAY IPOSITION(MB) THE ITH ELEMENT CONTAINING THE
C     ROW/COLUMN OF A TO WHICH THE ITH ROW/COLUMN OF B CORRESPONDS.
C  IF THE ITH ELEMENT OF IPOSITION IS NEGATIVE THE SIGN OF THE ITH ROW
C  AND COLUMN OF B IS CHANGED BEFORE ADDING. A ZERO VALUE INDICATES THAT
C  THE ITH ROW AND COLUMN OF B CONTRIBUTE NOTHING TO A.
C  NOTE, IBJB=IB*JB,IAJSEMI=IA*JSEMI.
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      REAL A(IAJSEMI),B(IBJB)
      INTEGER IPOSITION(MB),ROWA,ROWB,COLA,COLB
      JSEMI=0.1+IAJSEMI/IA
      DO 1 ROWB=1,MB
      DO 1 COLB=ROWB,MB
      I=(COLB-1)*IB+ROWB
      ROWA=IPOSITION(ROWB)
      IF(ROWA)0,1,4
      ROWA=-ROWA
      B(I)=-B(I)
    4 COLA=IPOSITION(COLB)
      IF(COLA)0,1,5
      COLA=-COLA
      B(I)=-B(I)
    5 NCOL=COLA-ROWA+1
      IF(COLA.LT.ROWA)NCOL=ROWA-COLA+1
      IF(NCOL.GT.JSEMI)GOTO2
      J=(NCOL-1)*IA+ROWA
      IF(COLA.LT.ROWA)J=J-ROWA+COLA
      A(J)=B(I)+A(J)
    1 CONTINUE
      RETURN
    2 WRITE(2,3)NCOL
    3 FORMAT(1H ,44HSEMIBANDWIDTH INSUFFICIENT REQUIRES AT LEAST,I5)
      JSEMI=0
      RETURN
      END
```

App.A (cont'd)

A.6 ADD TO VECTOR

```
      SUBROUTINE ADD TO VECTOR(A,MA,B,MB,IPOSITION,CONSTANT)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  THE SUBROUTINE ADDS THE ELEMENTS OF THE MB BY 1 COLUMN VECTOR [B],
C  MULTIPLIED BY A GIVEN CONSTANT,INTO POSITIONS IN THE MA BY 1 COLUMN
C  VECTOR [A] SPECIFIED BY THE ELEMENTS OF THE INTEGER ARRAY IPOSITION
C  THE SUBROUTINE REQUIRES:
C     THE REAL ARRAY B(MB) CONTAINING THE COLUMN VECTOR [B].
C     THE REAL ARRAY A(MA) CONTAINING THE COLUMN VECTOR [A].
C     THE INTEGER ARRAY IPOSITION(MB) THE ITH ELEMENT CONTAINING THE
C     ROW OF A TO WHICH THE ITH ROW OF B CORRESPONDS.
C     THE REAL NUMBER 'CONSTANT'.
C  IF THE ITH ELEMENT OF IPOSITION IS NEGATIVE THE SIGN OF THE ITH ROW
C  OF B IS CHANGED BEFORE ADDING. A ZERO VALUE INDICATES THAT THE ITH
C  ROW OF B CONTRIBUTES NOTHING TO A.
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      DIMENSION A(MA),B(MB),IPOSITION(MB)
      DO 2 I=1,MB
      J=IPOSITION(I)
      IF(J)0,2,1
      B(I)=-B(I)
      J=-J
    1 A(J)=A(J)+CONSTANT*B(I)
    2 CONTINUE
      RETURN
      END
```

A.7 ALTER BANDMAT

```
      SUBROUTINE ALTER BANDMAT(A,IAJSEMI,IA,B,MB,I,CONST)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  THE SUBROUTINE MULTIPLIES THE ITH COLUMN OF THE SYMMETRIC BANDED
C  MATRIX [A] BY A GIVEN CONSTANT,SUBTRACTS THIS COLUMN FROM THE COLUMN
C  VECTOR [B] AND REPLACES THE ITH ROW AND COLUMN OF [A] BY ZEROS EXCEPT
C  FOR THE LEADING DIAGONAL ELEMENT WHICH IS SET EQUAL TO 1.0.
C  THE SUBROUTINE REQUIRES:
C     THE INTEGER I= THE REQUIRED ROW/COLUMN NUMBER.
C     THE REAL ARRAY A(IA,JSEMI) CONTAINING THE UPPER HALF BAND OF [A].
C     THE REAL ARRAY B(MB) CONTAINING THE COLUMN VECTOR [B].
C     THE REAL NUMBER 'CONST'.
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      REAL A(IAJSEMI),B(MB)
      JSEMI=0.1+IAJSEMI/IA
      I3=0
      DO 2 J=2,JSEMI
      I1=I-J+1
      I2=I+J-1
      I3=I3+IA
      IF(I1.GE.1)B(I1)=B(I1)-A(I3+I1)*CONST
    2 IF(I2.LE.MB)B(I2)=B(I2)-A(I3+I)*CONST
      B(I)=CONST
      I3=-IA
      DO 1 J=1,JSEMI
      I1=I-J+1
      I3=I3+IA
      IF(I1.GE.1)A(I3+I1)=0.0
    1 A(I3+I)=0.0
      A(I)=1.0
      RETURN
      END
```

App.A (cont'd)

A.8 SQUARE BANDMAT

```
      SUBROUTINE SQUARE BANDMAT(A,IAJSEMI2,IA,JSEMI,MA)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  THE SUBROUTINE SQUARES THE SYMMETRIC BANDED MATRIX [A].
C  THE SUBROUTINE REQUIRES:
C     THE REAL ARRAY A(IA,JSEMI2) CONTAINING THE UPPER HALF BAND OF [A]
C     IN COLS 1 TO JSEMI.JSEMI2=2*JSEMI-1=SEMIBANDWIDTH OF [A] SQUARED.
C  NOTE, [A] IS OVERWRITTEN.
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      REAL A(IAJSEMI2)
      INTEGER ROW,COL,R,C
      JSEMI2=IAJSEMI2/IA+0.1
      J7=IA*JSEMI
      IROW=IA
      ICOL=JSEMI2+1
      L=0
      J5=-IA
      DO 2 ROW=1,MA
      J5=J5+IA
      J8=J5-IA
      DO 2 COL=ROW,MA
      J8=J8+IA
      J=COL-ROW+1
      IF(J.GT.JSEMI2)GOTO2
      IF(L.GE.1)GOTO9
      ICOL=ICOL-1
      IF(ICOL.GE.2+IA-IROW)GOTO6
      IROW=IROW-1
      ICOL=JSEMI2
    6 IF(IROW.GT.IA-JSEMI2+1)GOTO7
      L=1
      IROW=1
      ICOL=0
    9 ICOL=ICOL+1
      IF(ICOL.LE.JSEMI2)GOTO7
      ICOL=1
      IROW=IROW+1
    7 B=0.0
      J3=ICOL*IA-IA+IROW
      J6=-IA
      DO 2 KCOL=1,MA
      J6=J6+IA
      I1=J5-J6+KCOL
      K2=J6-J8+COL
      IF(ROW.LT.KCOL)I1=J6-J5+ROW
      IF(KCOL.LT.COL)K2=J8-J6+KCOL
    3 IF(I1.GT.J7.OR.K2.GT.J7)GOTO2
      A(J3)=A(I1)*A(K2)+B
      B=A(J3)
    2 CONTINUE
      L=0
      I1=MA
      J1=2
      DO 1 I=1,MA
      DO 1 J=1,JSEMI2
      IF(L.EQ.1)GOTO11
      ICOL=JSEMI2+1-J
      IROW=MA-JSEMI2+2-I
      IF(I.LE.MA-JSEMI2+1)GOTO8
      IF(L.EQ.0)IROW=IA-JSEMI2+1
   11 IF(L.NE.0.AND.ICOL.LE.JSEMI2-1)GOTO10
      IROW=IROW+1
      ICOL=1+IA-IROW
```

App.A (cont'd)

```
   10 L=1
      ICOL=ICOL+1
    8 J1=J1-1
      IF(J1.GE.1)GOTO1
      I1=I1-1
      J1=MA+1-I1
      IF(J1.GT.JSEMI2)J1=JSEMI2
    1 IF(I1.GE.1.OR.IROW.LE.IA)A(IA*J1-IA+I1)=A(IA*ICOL-IA+IROW)
      RETURN
      END
```

A.9 MULT BANDMAT VECTOR

```
      SUBROUTINE MULT BANDMAT VECTOR(A,IAJSEMI,IA,B,MB)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  THE SUBROUTINE POSTMULTIPLIES THE SYMMETRIC BANDED MATRIX [A] BY THE
C  COLUMN VECTOR [B].  THE SUBROUTINE REQUIRES:
C     THE REAL ARRAY A(IA,JSEMI) CONTAINING THE UPPER HALF BAND OF [A].
C     THE REAL ARRAY B(MB) CONTAINING THE COLUMN VECTOR [B].
C  THE ARRAY B IS OVERWRITTEN BY THE RESULT.  (IAJSEMI=IA*JSEMI)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  ---------------------------------------------------------
C  SUBROUTINE FAILS FOR LACK OF STORAGE IN SPARE IF MB>100
      REAL SPARE(100)
C  ---------------------------------------------------------
      REAL A(IAJSEMI),B(MB)
      JSEMI=0.1+IAJSEMI/IA
      DO 2 I=1,MB
    2 SPARE(I)=0.0
      J1=IAJSEMI-IA+1
      DO 1 I=1,MB
      J2=0
      DO 1 J=1,J1,IA
      J2=J2+1
      J3=I-J2+1
      J4=I+J2-1
      IF(J3.GE.1)SPARE(J3)=SPARE(J3)+A(J+J3-1)*B(I)
      IF(J.EQ.1)GOTO1
      IF(J4.LE.MB)SPARE(J4)=SPARE(J4)+A(J+I-1)*B(I)
    1 CONTINUE
      DO 3 I=1,MB
    3 B(I)=SPARE(I)
      RETURN
      END
```

A.10 POSDEF MATINV

```
      SUBROUTINE POSDEF MATINV(A,MAMA)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  THE SUBROUTINE FINDS THE INVERSE OF THE MA BY MA SYMMETRIC POSITIVE
C  DEFINITE MATRIX [A].  THE SUBROUTINE REQUIRES:
C     THE REAL ARRAY A(MA,MA) CONTAINING THE COMPLETE MATRIX [A].
C  A IS OVERWRITTEN BY ITS INVERSE. (MAMA=MA*MA)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      REAL A(MAMA)
      INTEGER ROW,COL
      MA=0.1+SQRT(1.0*MAMA)
```

App.A (cont'd)

```
      J1=MAMA-MA+1
      J2=0
      DO 1 COL=1,J1,MA
      J2=J2+1
      DO 1 ROW=1,MA
      J3=ROW+COL-1
      B=0.0
      IF(J2.EQ.1.OR.ROW-J2.LT.0)GOTO4
      J4=ROW
      DO 2 M=1,J2-1
      B=B+A(J4)*A(J2+J4-ROW)
    2 J4=J4+MA
    4 IF(ROW-J2)1,3,0
      A(J3)=(A(J3)-B)/A(COL+J2-1)
      GOTO1
    3 A(J3)=SQRT(A(J3)-B)
    1 CONTINUE
      J2=0
      DO 5 COL=1,J1,MA
      J2=J2+1
      J5=-MA
      DO 5 ROW=1,MA
      J3=ROW+COL-1
      J5=J5+MA
      B=0.0
      IF(ROW-J2.LE.0)GOTO6
      J4=J3
      DO 7 M=J2,ROW-1
      B=B+A(J4)*A(COL+M-1)
    7 J4=J4+MA
    6 IF(ROW-J2)5,8,0
      A(J3)=-B/A(ROW+J5)
      GOTO5
    8 A(J3)=1.0/A(COL+J2-1)
    5 CONTINUE
      J2=0
      DO 9 COL=1,J1,MA
      J4=0
      J2=J2+1
      DO 9 ROW=1,J1,MA
      J4=J4+1
      B=0.0
      IF(J4-J2)11,0,0
      DO 10 M=J4,MA
   10 B=B+A(ROW+M-1)*A(COL+M-1)
      GOTO12
   11 A(COL+J4-1)=A(ROW+J2-1)
      GOTO9
   12 A(COL+J4-1)=B
    9 CONTINUE
      RETURN
      END
```

A.11 SOLVE BANDMAT

```
      SUBROUTINE SOLVE BANDMAT(A,IAJSEMI,IA,MA,B,IAJB,NCHANGE)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  THE SUBROUTINE SOLVES THE MATRIX EQUATION [A]*[X]=[B] WHERE [A] IS A
C  MA BY MA SYMMETRIC BANDED MATRIX. THE SUBROUTINE REQUIRES:
C     THE REAL ARRAY A(IA,JSEMI) CONTAINING A HALF BAND OF [A],IA>OR=MA.
C     THE REAL ARRAY B(IA,JB) CONTAINING THE MATRIX OF RHS,[B],WHERE
C     JB IS THE NUMBER OF RHS.
```

App.A (cont'd)

```
C     THE INTEGER NCHANGE=1 IF ARRAY A CONTAINS THE UPPER HALF BAND
C     OF [A],AND =0 IF A CONTAINS THE LOWER HALF BAND.
C  THE ARRAY B IS OVERWRITTEN BY THE SOLUTION.
C  NOTE  IAJB=IA*JB , IAJSEMI=IA*JSEMI.
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      REAL A(IAJSEMI),B(IAJB)
      JSEMI=0.1+IAJSEMI/IA
      JB=0.1+IAJB/IA
      IF(NCHANGE.EQ.0)GOTO4
      DO 6 I=1,MA
      I1=MA-I+1
      DO 6 J=1,JSEMI
      I2=J*IA
    6 IF(I1-J+1.GE.1)A(IAJSEMI-I2+I1)=A(I2-IA+I1-J+1)
    4 L=JSEMI-1
      L1=L*IA
      AA=1.0
      KA=0
      DO 1 I=1,MA
      IF(I-L)0,0,2
      IB=L-I+1
      GOTO3
    2 IB=0
    3 ID=I-L+IB
      DO 1 IH=IB+1,L+1
      J=IH-1
      IE=J-1
      J1=J*IA+I
      IC=L-J+IB
      YA=A(J1)
      IF(IB-IE)0,0,20
      DO 5 IJ=IB+1,IE+1
      IG=IJ-1
      YA=YA-A(IG*IA+I)*A(IC*IA+ID)
    5 IC=IC+1
   20 IF(J-L)7,0,7
      AA=AA*YA
      IF(YA)10,0,10
      KA=0
      GOTO8
   10 IF(ABS(AA)-1)9,0,0
      AA=0.0625*AA
      KA=KA+4
      GOTO10
    9 IF(ABS(AA)-0.0625)0,11,11
      AA=16*AA
      KA=KA-4
      GOTO9
   11 IF(YA)8,0,0
      A(J1)=1.0/SQRT(YA)
      GOTO1
    7 A(J1)=YA*A(L1+ID)
      ID=ID+1
    1 KA=KA
      ID=L-1
      DO 16 J=1,JB
      DO 19 I=1,MA
      J1=(J-1)*IA
      IF(I-L)0,0,13
      IB=L-I+1
      GOTO14
   13 IB=0
   14 IC=I
      YA=B(J1+I)
```

App.A (cont'd)

```
      IF(IB-ID)0,0,19
      DO 15 IH=IB+1,ID+1
      IE=IH-1
      KA=ID+IB-IE
      IC=IC-1
   15 YA=YA-A(KA*IA+I)*B(J1+IC)
   19 B(J1+I)=YA*A(L1+I)
      DO 16 IE=1,MA
      I=MA-IE+1
      IF(MA-I-L)0,0,21
      IB=L-MA+I
      GOTO17
   21 IB=0
   17 YA=B(J1+I)
      IC=I
      IF(IB-ID)0,0,16
      DO 18 IJ=IB+1,ID+1
      IG=IJ-1
      KA=ID+IB-IG
      IC=IC+1
   18 YA=YA-A(KA*IA+IC)*B(J1+IC)
   16 B(J1+I)=YA*A(L1+I)
      RETURN
    8 KA=KA
      WRITE(2,30)
   30 FORMAT(1H ,19HFAILED IN INVERSION)
      IC=0
      RETURN
      END
```

A.12 SOLVE CONSTRAINED BANDMAT

```
      SUBROUTINE SOLVE CONSTRAINED BANDMAT(A,B,C,D,IA,JSEMI,CONROW,
     1IDENT,MA,NAM)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  WHEN THE MATRIX EQUATION [A]*[X]=[B] IS DERIVED BY WAY OF A
C  VARIATIONAL PROCESS THE SUBROUTINE OBTAINS THE SOLUTION FOR [X]
C  SUBJECT TO THE LINEAR CONSTRAINTS [C]*[X]=[D],WHERE [A] IS SYMMETRIC
C  POSITIVE DEFINITE BANDED,[B] AND [D] ARE COLUMN VECTORS AND [C] IS
C  A TRIDIAGONAL MATRIX COMPOSED OF UNIT DIAGONAL ELEMENTS AND/OR
C  SQUARE SUBMATRICES OF ORDER TWO. THE SUBROUTINE REQUIRES:
C     THE REAL ARRAY A(IA,JSEMI) CONTAINING THE UPPER HALF BAND OF [A].
C     THE REAL ARRAY B(IA) CONTAINING THE COLUMN VECTOR [B] (IA>OR=MA).
C     THE REAL ARRAY C(IA,2) CONTAINING THE 2 BY 2 SUBMATRICES OF [C].
C     THE REAL ARRAY D(IA) CONTAINING THE COLUMN_VECTOR [D].
C     THE INTEGER ARRAY IDENT(NAM).
C     THE INTEGER ARRAY CONROW(IA) WHICH IS FILLED AS FOLLOWS:
C     CONROW(I)=0 IF ROW I IS NOT A CONSTRAINT ROW,
C     CONROW(I)=1 IF ROW I IS AN ISOLATED CONSTRAINT ROW,
C     CONROW(I)=2 IF ROW I IS A CONSTRAINT ROW FORMING A SUBMATRIX WITH
C     ROW I+1.IF BOTH I AND I+1 ARE CONSTRAINT ROWS THEN
C     CONROW(I)=CONROW(I+1)=2.
C     THE SUBROUTINE 'SOLVE BANDMAT'.
C  THE ARRAY B IS OVERWRITTEN BY THE RESULTS.
C  NOTE, NAM=TOTAL NUMBER OF NODES AND MIDPOINTS.
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  ------------------------------------------------------
C  SUBROUTINE FAULTS FOR LACK OF STORAGE IN SPARE IF MA>200.
      REAL SPARE(200)
C  ------------------------------------------------------
C
```

App.A (cont'd)

```
      REAL A(IA,JSEMI),B(IA),C(IA,2),D(IA)
      INTEGER CONROW(IA),IDENT(NAM),COL
C
C  INVERT CONSTRAINT MATRIX C.
C
      DO 5 I=1,NAM
      J=IDENT(I)
   11 K=CONROW(J)
      IF(K.EQ.0.OR.K.EQ.1)GOTO1
      IF(CONROW(J)-CONROW(J+1))0,0,2
    4 DET=C(J,1)*C(J+1,2)-C(J,2)*C(J+1,1)
      IF(ABS(DET).GT.0.00001)GOTO6
      C(J+1,1),D(J+1)=0.0
      C(J+1,2)=1.0
      CONROW(J)=2
      CONROW(J+1)=0
    2 IF(ABS(C(J,1)).GT.ABS(C(J,2)))GOTO7
      C(J+1,1)=C(J,1)
      C(J+1,2)=C(J,2)
      C(J,1)=1.0
      C(J,2)=0.0
      CONROW(J+1)=3
      CONROW(J)=0
      D(J+1)=D(J)
      D(J)=0.0
      GOTO 9
    7 C(J,2)=-C(J,2)/C(J,1)
      C(J,1)=1.0/C(J,1)
      J=J+1
      GOTO1
    9 C(J+1,1)=-C(J+1,1)/C(J+1,2)
      C(J+1,2)=1.0/C(J+1,2)
      J=J+1
      GOTO1
    6 AA=1.0/DET
      C(J,2)=-C(J,2)*AA
      C(J+1,1)=-C(J+1,1)*AA
      AB=C(J,1)*AA
      C(J,1)=C(J+1,2)*AA
      C(J+1,2)=AB
      J=J+1
    1 IF(J.GE.MA)GOTO 5
      IF(I.EQ.NAM)GOTO 12
      IF(J+1.NE.IDENT(I+1))GOTO 12
      GOTO5
   12 J=J+1
      GOTO11
    5 CONTINUE
C
C  PERFORM TRIPLE MATRIX MULTIPLICATION
C
      DO 10 I=1,MA
   10 SPARE(I)=0.0
      J=0
   31 J=J+1
      K=CONROW(J)+1
      GOTO(13,13,14,15)K
   15 J=J-1
   14 DO 16 I=1,MA
      COL=J-I+1
      IF(I.GE.J)GOTO16
      IF(COL-JSEMI)17,0,16
      A(I,COL)=C(J,1)*A(I,COL)
```

App.A (concl'd)

```
      GOTO16
   17 AA=A(I,COL)
      A(I,COL)=C(J,1)*AA+C(J+1,1)*A(I,COL+1)
      A(I,COL+1)=C(J,2)*AA+C(J+1,2)*A(I,COL+1)
   16 CONTINUE
      DO 18 COL=3,JSEMI
      AA=A(J,COL)
      A(J,COL)=C(J,1)*AA+C(J+1,1)*A(J+1,COL-1)
   18 A(J+1,COL-1)=C(J,2)*AA+C(J+1,2)*A(J+1,COL-1)
      A(J+1,JSEMI)=C(J+1,2)*A(J+1,JSEMI)
      A1=A(J,1)*C(J,1)+A(J,2)*C(J+1,1)
      A2=A(J,1)*C(J,2)+A(J,2)*C(J+1,2)
      A3=A(J,2)*C(J,1)+A(J+1,1)*C(J+1,1)
      A4=A(J,2)*C(J,2)+A(J+1,1)*C(J+1,2)
      A(J,1)=C(J,1)*A1+C(J+1,1)*A3
      A(J,2)=C(J,1)*A2+C(J+1,1)*A4
      A(J+1,1)=C(J,2)*A2+C(J+1,2)*A4
      J=J+1
   13 IF(J.LT.MA)GOTO 31
      J=0
   30 J=J+1
      K=CONROW(J)+1
      GOTO(19,19,20,21)K
   21 J=J-1
   20 AA=B(J+1)
      B(J+1)=C(J,2)*B(J)+C(J+1,2)*AA
      B(J)=C(J,1)*B(J)+C(J+1,1)*AA
      J=J+1
   19 IF(J.LT.MA)GOTO 30
      DO 22 I=1,MA
      IF(CONROW(I))0,22,0
      DO 22 J=1,JSEMI
      J3=I-J+1
      J4=I+J-1
      IF(J3.GE.1)SPARE(J3)=SPARE(J3)+A(J3,J)*D(I)
      IF(J.EQ.1)GOTO22
      IF(J4.LE.MA)SPARE(J4)=SPARE(J4)+A(I,J)*D(I)
   22 CONTINUE
      DO 23 I=1,MA
      IF(CONROW(I))0,23,0
      B(I)=0.0
      SPARE(I)=0.0
      DO 24 J=1,JSEMI
      IF(I-J+1.GE.1)A(I-J+1,J)=0.0
      A(I,J)=0.0
   24 A(I,1)=1.0
   23 CONTINUE
      DO 25 I=1,MA
   25 B(I)=B(I)-SPARE(I)+D(I)
      IAJSEMI=IA*JSEMI
      CALL SOLVE BANDMAT(A,IAJSEMI,IA,MA,B,IA,1)
      J=0
   29 J=J+1
      K=CONROW(J)+1
      GOTO(26,26,28,27)K
   27 J=J-1
   28 AA=B(J+1)
      B(J+1)=C(J+1,1)*B(J)+C(J+1,2)*AA
      B(J)=C(J,1)*B(J)+C(J,2)*AA
      J=J+1
   26 IF(J.LT.MA)GOTO 29
      RETURN
      END
```

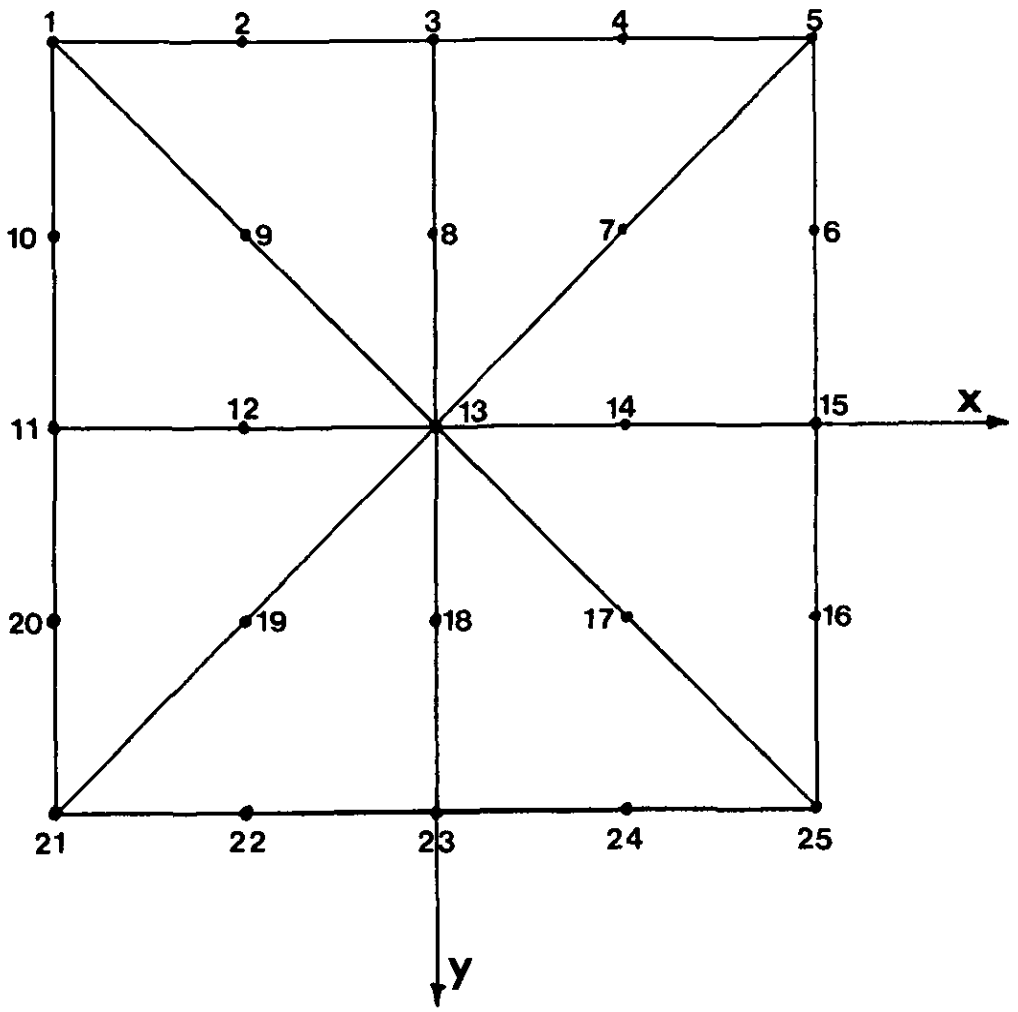


Fig.1 Square plate divided into triangular finite elements

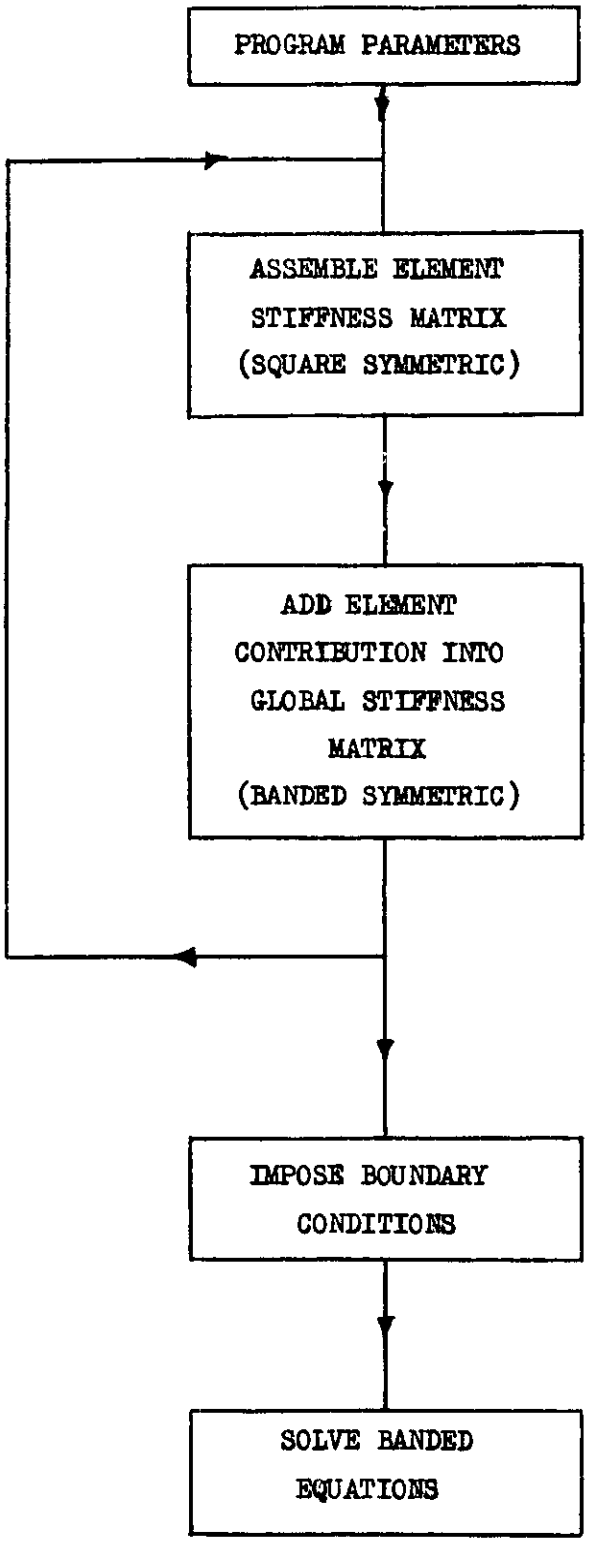


Fig.2 Flow chart of a typical finite element analysis program

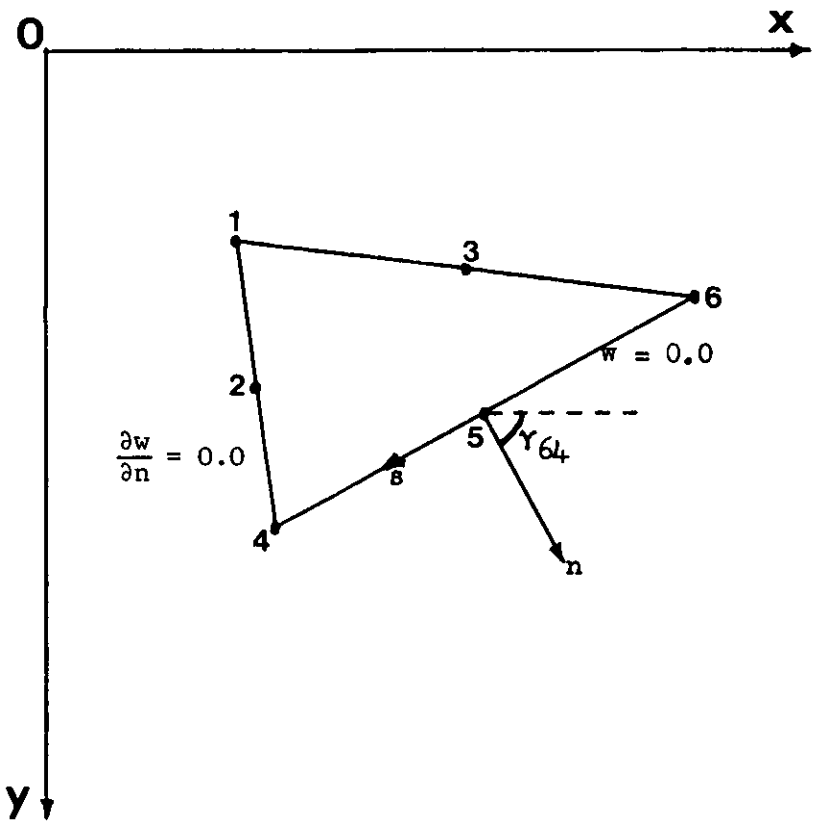


Fig.3 Notation and kinematic boundary conditions for illustrative example

Printed in England for Her Majesty's Stationery Office by the Royal Aircraft Establishment, Farnborough Dd 503427 K4

ARC CP No 1223
August 1971

531 25
518 5
531 258

Merrifield, B C

FORTRAN SUBROUTINES FOR FINITE ELEMENT ANALYSIS

Twelve subroutines, written in ICL 1900 Fortran are presented for matrix and other operations which are commonly encountered in the finite element analysis of structures. Although the subroutines have been developed specifically to deal with problems concerning flat plates they clearly possess some wider generality

Details are given of each subroutine including its method for use and its complete listing

DETACHABLE ABSTRACT CARDS

ARC CP No 1223
August 1971

531 25
518 5
531 258

Merrifield, B C

FORTRAN SUBROUTINES FOR FINITE ELEMENT ANALYSIS

Twelve subroutines, written in ICL 1900 Fortran are presented for matrix and other operations which are commonly encountered in the finite element analysis of structures Although the subroutines have been developed specifically to deal with problems concerning flat plates they clearly possess some wider generality

Details are given of each subroutine including its method for use and its complete listing

DETACHABLE ABSTRACT CARDS

ARC CP No 1223
August 1971

531 25
518 5
531 258

Merrifield, B C

FORTRAN SUBROUTINES FOR FINITE ELEMENT ANALYSIS

Twelve subroutines, written in ICL 1900 Fortran are presented for matrix and other operations which are commonly encountered in the finite element analysis of structures Although the subroutines have been developed specifically to deal with problems concerning flat plates they clearly possess some wider generality

Details are given of each subroutine including its method for use and its complete listing

Cut here

Cut here

These abstract cards are inserted in Technical Reports for the convenience of Librarians and others who need to maintain an Information Index

Detached cards are subject to the same Security Regulations as the parent document, and a record of their location should be made on the inside of the back cover of the parent document

© *Crown copyright 1972*

Published by
HER MAJESTY'S STATIONERY OFFICE

To be purchased from
49 High Holborn, London WC1 V 6HB
13a Castle Street, Edinburgh EH2 3AR
109 St Mary Street Cardiff CF1 1JW
Brazennose Street, Manchester M60 8AS
50 Fairfax Street, Bristol BS1 3DE
258 Broad Street, Birmingham B1 2HE
80 Chichester Street, Belfast BT1 4JY
or through booksellers

C.P. No. 1223

SBN 11 470781 2