FEAT2D

Finite Element Analysis Tools

User Manual

Release 1.3

H. Blum J. Harig S. Müller P. Schreiber S. Turek
Universität Heidelberg, Institut für Angewandte Mathematik
Im Neuenheimer Feld 293, D–69120 Heidelberg

Heidelberg, 1995

Contents

Short description

The progam package FEAT2D is a general purpose subroutine system for the numerical solution of partial differential equations by the finite element method. FEAT2D is designed to handle problems in two space dimensions, namely

- Scalar elliptic boundary value problems,
- Elliptic systems of equations,
- Time dependent problems.

FEAT2D is not a user oriented system, it only provides subroutines for several main steps in a finite element program. The user should be familiar with the mathematical formulation of the discrete problems. The data structure of FEAT2D is transparent so that modifications or augmentations of the program package are very easy, for instance the implementation of new elements. For details in the design and data structures of finite element codes the reader is referred to the books of Axelsson/Barker [1] and Schwarz [3], [4].

There are several main groups of subroutines in the system:

- Generation of subdivisions of the domain Ω
- Assembly of global matrices related to a bilinear form $a(\phi_i, \phi_j)$
- Assembly of right hand side vectors related to a linear form $l(\phi_i)$
- Implementation of standard boundary conditions
- Solution of the resulting linear systems of equations by iterative solvers
- Error analysis for test problems where the exact solution is known

Moreover sample programs for standard applications are included. These can be used as starting programs which can be modified for the actual application. Most FEAT2D subroutines can be used outside FEAT2D as well since the special memory management is separated from the working program units.

1. General Concepts and Notation

1.1. Groups of subprograms

The subprograms of FEAT2D are grouped with respect to their tasks. The names of all programs in one group start with the same key letter. Here, we give a short overview of the different tasks.

`A` Calculation of matrices corresponding to bilinear forms for triangular and quadrilateral elements
Support of several storage techniques and calculation of the corresponding pointer vectors

`B` Unused

`C` Cubature formulas
Defined on 1- and 2-d reference elements

`D` Reserved for finite difference applications

`E` Element library

`F` Unused

`G` Normalized Input/Output for use of graphic packages

`H` Unused

`I` Iterative solvers of linear systems of equations
Versions for solution, smoothing iterations and preconditioning

`J` Reserved for eigenvalue problems

`K` Unused

`L` Basic linear algebra applications
Matrix-vector and vector-vector operations

`M` Multigrid components
Prolongations and restrictions

`N` Auxiliary routines
Correspondence of local and global quantities

`O` Input/Output

`P` Reserved for postprocessing routines
Comparison to exact solutions etc.

`Q` Unused

`R` Reorganization
Compression of zero entries in stiffness matrices
Reordering of matrices with respect to different storage techniques

`S` Generation of subdivisions

`T` Unused

`U` Unused

`V` Calculation of vectors corresponding to linear forms

`W` Error handling
Selective dump of contents of `COMMON` blocks and arrays allocated on the workspace

`X` Direct communication with `Z`-routines
Preparation of parameter lists for routines `A-W`
Check of size and data type of arrays allocated on the workspace

`Y` External subroutines
Preparation of parameter lists for routines `A-W`

`Z` Handling of the pseudodynamic memory management
Machine dependent system routines

1.2. Symbolic names and reserved starting letters

All routines are written in standard `FORTRAN 77`. Correspondingly, the names of variables and arrays are restricted to 6 alphanumeric characters. According to the standard `FORTRAN` convention, names starting with characters `I-N` are of type `INTEGER`. All other starting letters, except `V` and `B`, implicitly denote `DOUBLE PRECISION` quantities. Variables starting with one of the following letters have a specialized meaning.

- `N...` Number of ...
  Global or local constants or effective dimension of arrays
- `NN...` Maximum dimension of arrays (usually defined in `PARAMETER` statements)
- `K...` `INTEGER` arrays, usually of problem dependent dimension, mainly used as pointer vectors, for description of triangulations, etc.
- `I...` Local variables and subscripts of arrays frequently used as `DO` loop variables
- `L...` Array descriptors in pseudodynamic memory management
- `M...` Input/Output parameters, unit numbers
- `J...` Left for free use as auxiliary variables
- `V...` Single precision array
- `D...` Double precision array
- `B...` Logical (boolean) variables and arrays

<u>Example</u>

Array name:	`DAUX` (`DOUBLE PRECISION`) or `KAUX` (`INTEGER`)
Maximum dimension:	`NNAUX`
Effective dimension:	`NAUX`
Reference to a single element:	`IAUX` (`JAUX` may be used in inner `DO`-loops)

1.3. Pseudodynamic memory management

FEAT2D makes use of a pseudodynamic memory management for most of the arrays in use. It provides a workspace vector containing in particular those arrays whose dimension depends on the problem and, moreover, local auxiliary vectors. The workspace vector `DWORK` is allocated on the `BLANK COMMON` and is of type `DOUBLE PRECISION`.

The `BLANK COMMON`

```
      PARAMETER (NNARR=299)

      COMMON NWORK,IWORK,IWMAX,L(NNARR),DWORK(1)
```

Explanation

`NWORK`	Effective dimension of `DWORK`
`IWORK`	Pointer to the last element of `DWORK` in use
`IWMAX`	Maximum number of (`DOUBLE PRECISION`) elements of `DWORK` used during the program
`L`	`INTEGER` array containing the starting address of the arrays allocated on `DWORK`, corresponding to data type (see below)
`NNARR`	Maximum number of arrays on `DWORK` (set to `299` in the present version)
`DWORK`	Workspace vector

In the main program the dimension of `DWORK` must be set to `NNWORK`, the overall maximum:

```
      PARAMETER (NNARR=299,NNWORK=......)

      COMMON NWORK,IWORK,IWMAX,L(NNARR),DWORK(NNWORK)
```

The value of `NNWORK` is defined by the user.

It is possible to allocate arrays of different data type on `DWORK`. In the present version, the following types are supported:

Datatype 1	`DOUBLE PRECISION`
Datatype 2	`REAL`
Datatype 3	`INTEGER`

The arrays on `DWORK` are addressed via their number (token) which ranges between `1` and `NNARR`. The auxiliary routines which manage those arrays write messages or notes to the units `MSYS` or `MPROT` and error messages to unit `MERR`. Besides the number of the array the user may provide array names of the type `CHARACTER*6` for these messages. In the following, we describe the usage of the auxiliary routines which are placed in group `Z`.

1. Allocation of a new array on DWORK

```
SUBROUTINE ZNEW(ILONG,ITYPE,LNR,ARR)
```

Parameters Input

ILONG	I*4	Length of the array For ILONG=0 the whole free space on DWORK is allocated.
ITYPE	I*4	Data type of the array For ITYPE>0 the data type (1,2, or 3) is chosen as above and the array is initialized by 0. For ITYPE<0 the data type -ITYPE is chosen but the array is not initialized.
ARR	C*6	Array name, used for messages only.

Output

LNR	I*4	New number of the array.

2. Shortening or deletion of an array on DWORK

```
SUBROUTINE ZDISP(ILONG,LNR,ARR)
```

Parameters Input

ILONG	I*4	Number of elements to be left ILONG=0 removes the whole array and releases the number LNR.
LNR	I*4	Number of array
ARR	C*6	Array name (for messages only)

Output

LNR	I*4	Set to 0 if ILONG=0

3. Further service routines

Fill array with number LNR (name ARR) with 0 (corresponding to data type).

```
SUBROUTINE ZCLEAR(LNR,ARR)
```

Copy array with number LNR1 (name ARR1) onto array with number LNR2 (name ARR2).

```
SUBROUTINE ZCPY(LNR1,ARR1,LNR2,ARR2)
```

Change data type of array with number `LNR` (name `ARR`) to type `ITYPE`.

```
      SUBROUTINE ZCTYPE(ITYPE,LNR,ARR)
```

Return free space on `DWORK` with respect to data type `ITYPE` on variable `IFREE`.

```
      SUBROUTINE ZFREE(ITYPE,IFREE)
```

Return number of elements of array with number `LNR` on variable `LENGTH`.

```
      SUBROUTINE ZLEN(LNR,LENGTH)
```

Return number of `DOUBLE PRECISION` elements used for array with number `LNR` on variable `LENGTH`.

```
      SUBROUTINE ZLEN8(LNR,LENGTH)
```

Return data type (1,2, or 3) of array with number `LNR` on variable `LTYPE`.

```
      SUBROUTINE ZTYPE(LNR,LTYPE)
```

4. Reference to an array on `DWORK`

In order to pass the effective starting address of an array to a subprogram one has to include the following `DIMENSION` and `EQUIVALENCE` statement into the calling routine.

```
      DIMENSION VWORK(1),KWORK(1)

      EQUIVALENCE (DWORK(1),VWORK(1),KWORK(1))
```

Moreover, the `BLANK COMMON` must be available. Then, an array with number `LARRAY` is addressed as follows:

Datatype 1	`DWORK(L(LARRAY))`
Datatype 2	`VWORK(L(LARRAY))`
Datatype 3	`KWORK(L(LARRAY))`

5. Examples

In this section, we discuss a sample program in order to demonstrate the usage of the above mentioned routines.

```
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      PARAMETER (NNARR=299,NNWORK=10000)
      DIMENSION VWORK(1),KWORK(1)
      COMMON NWORK,IWORK,IWMAX,L(NNARR),DWORK(NNWORK)
      EQUIVALENCE (DWORK(1),VWORK(1),KWORK(1))
C
1     CALL ZNEW(10,1,LX,'DX    ')
2     CALL ZNEW(20,-3,LI,'KI    ')
3     CALL FILL1(DWORK(L(LX)),10)
4     CALL FILL2(KWORK(L(LI)),20)
5     CALL ZCPY(LI,'KI    ',LI1,'KI1   ')
6     CALL ZTYPE(LI1,ITYPE)
7     CALL ZCTYPE(LX,2,'DX    ')
8     CALL ZDISP(3,LX,'VX    ')
9     CALL ZDISP(0,LI,'KI    ')
10    CALL ZLEN(LX,ILEN)
11    CALL ZLEN8(LX,ILEN8)
12    CALL ZCLEAR(LI1)
13    WRITE(*,*) (VWORK(L(LX)+IX),IX=0,2)
```

Explanation

Line 1: A new array is allocated, data type `DOUBLE PRECISION`, length `10`. The array is initialized by `0D0`. `ZNEW` returns the number of the array on `LX`. If the routine `ZNEW` is called the first time, this number equals `1`.

Line 2: A new array is allocated, data type `INTEGER`, length `20`. The array is not initialized. `ZNEW` returns the number `2` on `LI`.

Line 3: A user provided routine `FILL1` is called to assign values to the (`DOUBLE PRECISION`) array `LX` (`DX`).

Line 4: A user provided routine `FILL2` is called to assign values to `INTEGER` array `LI` (`KI`).

Line 5: A new array of type `INTEGER` is allocated with the same length (`20`) as `LI1` and `LI` is copied onto the new array. `ZCPY` returns the value `3` on `LI1` for the new array.

Line 6: The data type of the array `LI1` is checked. `ITYPE` returns the value `3`.

Line 7: The data type of the array `LX` (`DX`) is changed from `DOUBLE PRECISION` to `REAL`. This may be used to save storage locations or to provide output for graphic packages which usually work with single precision data.

Line 8: The length of the array `LX` is reduced to `3`. The remaining space on the workspace vector is released.

Line 9: Array `LI` is completely removed from `DWORK`.

Line 10: The length of array `LX` is checked. `ILEN` returns the value `3`.

Line 11: The number of `DOUBLE PRECISION` storage locations used for array `LX` is checked. `ILEN8` returns the value `2`.

Line 12: The array `LI1` is filled with (`INTEGER`) zeroes.

Line 13: The `3` components of array `LX (VX)` are printed. The starting address of the array would be `DWORK(L(LX))` if the data type were `1` and `KWORK(L(LX))` for data type equal to `3`.

1.4. Description of the domain Ω

In this subsection, we explain how a two-dimensional bounded domain must be described. FEAT2D only needs information on the boundary $\partial\Omega$. It is assumed that the boundary consists of finitely many simply closed curves which are given in parametrized form. The orientation should be chosen such that the interior of the domain is locally on the left of the boundary curves. The parameter for each boundary component starts from `0`.

The user provided subprograms to describe the parametrization are the following:

```
DOUBLE PRECISION FUNCTION PARX(T,IBCT)

DOUBLE PRECISION FUNCTION PARY(T,IBCT)

DOUBLE PRECISION FUNCTION TMAX(IBCT)
```

Parameters Input

`T`	Parameter of boundary point
`IBCT`	Number of boundary component, boundary components are enumerated from `1` to `NBCT`.

<u>Explanation</u>

`PARX` `PARY`	`X`- and `Y`-coordinates of boundary point with parameter `T` on boundary component `IBCT`

`TMAX`	Maximum parameter for each boundary component, first positive value for which
	`PARX(0D0,IBCT).EQ.PARX(TMAX(IBCT),IBCT).AND.`
	`PARY(0D0,IBCT).EQ.PARY(TMAX(IBCT),IBCT)`
	for `IBCT=1` to `NBCT`

<u>Example</u>

Circular annulus, outer radius equals `1`, inner radius equals `0.5`

```
      DOUBLE PRECISION FUNCTION ANNX(T,IBCT)
      DOUBLE PRECISION T,R1,R2,PI2
      PARAMETER (R1=1D0,R2=.5D0,PI2=6.28...D0)
C
      IF (IBCT.EQ.1) ANNX=R1*COS(T)
      IF (IBCT.EQ.2) ANNX=R2*COS(PI2-T)
      END
```

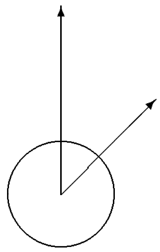

Figure 1.1: Circular annulus

```
      DOUBLE PRECISION FUNCTION ANNY(T,IBCT)
      DOUBLE PRECISION T,R1,R2,PI2
      PARAMETER (R1=1D0,R2=.5D0,PI2=6.28...D0)
C
      IF (IBCT.EQ.1) ANNY=R1*SIN(T)
      IF (IBCT.EQ.2) ANNY=R2*SIN(PI2-T)
      END

      DOUBLE PRECISION FUNCTION ANNMAX(IBCT)
      DOUBLE PRECISION PI2
      PARAMETER (PI2=6.28...D0)
C
      IF (IBCT.EQ.1) ANNMAX=PI2
      IF (IBCT.EQ.2) ANNMAX=PI2
      END
```

1.5. Description of the subdivision

Subdivisions in FEAT2D can be described in different ways. The first one provides information on subdivisions that are globally, not necessarily uniform, refined. A second possibility, especially with regard to parallelized applications, is the administration of piecewise uniform macro-triangulations.

Global subdivisions

The variables and arrays described in this section contain all the information needed for the generation of the finite element stiffness matrices and right hand side vectors and for the implementation of boundary conditions. The information on the subdivision is passed to subprograms by the following two `COMMON` blocks, the first containing scalar information (dimensions) and the second containing numbers of arrays describing the subdivision.

```
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /TRIAA/  LCORVG,LCORMG,LVERT,LMID,LADJ,LVEL,LMEL,LNPR,LMM,
     *                LVBD,LEBD,LBCT,LVBDP,LMBDP
```

Explanation

a) Contents of `/TRIAD/` – Dimensions

- `NEL` Total number of elements
- `NVT` Total number of vertices
- `NMT` Total number of edges (midpoints)
  `NMT` is set to `0` if no information about edges is generated, e.g., numbers and/or coordinates of the midpoints. Information on midpoints is generated by the routines `XS2M/S2M`.
- `NVE` Number of vertices per element
  `NVE` is set to `3` for purely triangular meshes and is set to `4` for quadrilateral meshes and meshes formed by triangles and quadrilaterals. In the present version no meshes of mixed type are supported.
- `NVEL` Maximum number of elements meeting at one vertex
  `NVEL` is set to `0` if this information is not available. `NVEL` is set by the routines `XS2V/S2V`.
- `NBCT` Total number of boundary components,
  i.e., the number of simply closed curves describing the boundary of the domain (see Section 1.4).
- `NVBD` Total number of vertices on the boundary (sum over all boundary components). `NVBD` is calculated by routine `SIVBD`.

b) Contents of `/TRIAA/` – Array descriptors

We give a list of the array descriptors and the corresponding arrays together with their effective dimension.

`LCORVG` `DIMENSION DCORVG(2,NVT)`
Array containing information about the coordinates of the vertices
a) Interior vertices
`DCORVG(1,IVT)` – `X`-coordinate of vertex `IVT`
`DCORVG(2,IVT)` – `Y`-coordinate of vertex `IVT`
b) Boundary vertex during grid generation
`DCORVG(1,IVT)` – Parameter of boundary vertex. The cartesian coordinates can be obtained using the routines `PARX`, `PARY`, described in the preceding section.
`DCORVG(2,IVT)` – Parameter of the edge midpoint following the boundary vertex `IVT`. This information is provided if `NMT`$\neq$`0`.
After the generation of the grid `DCORVG(.,IVT)` is overwritten by the cartesian coordinates of vertex `IVT`. The parametrization information can be saved on arrays `KVBDP` and `KMBDP`.

`LCORMG` `DIMENSION DCORMG(2,NMT)`
Array containing the cartesian coordinates of the midpoints of edges.
This information is needed only for isoparametric meshes using quadratic transformation onto the reference element. In the present version of FEAT2D `LCORMG` is set to `0`.

`LVERT` `DIMENSION KVERT(NNVE,NEL)`
Array containing the numbers of vertices for each element in counterclockwise sense.
Convention: For triangular elements, the last component `KVERT(NNVE,.)` is set to `0`.

`LMID` `DIMENSION KMID(NNVE,NEL)`
Array containing the numbers of midpoints of edges for each element in counterclockwise sense; the midpoints are given numbers ranging from `NVT+1` to `NVT+NMT`.
Convention: For triangular elements, the last component `KMID(NNVE,.)` is set to `0`. If finite elements possessing degrees of freedom on edges are employed, the array `KMID` must be generated. This is done by the routines `XS2M/S2M`.

`LADJ` `DIMENSION KADJ(NNVE,NEL)`
Array containing the numbers of the neighboring elements for each element of the subdivision.
Same convention for enumeration as for midpoints of edges.
For triangular elements, the last component `KADJ(NNVE,.)` is set to `0`. The array `KADJ` is needed for the generation of subdivisions and, e.g., for the implementation of routines for multigrid prolongation and restriction.

`LVEL` `DIMENSION KVEL(NVEL,NVT)`
Array containing the numbers of the elements meeting at a vertex.
Remember that `NVEL` is the maximum number of elements meeting at one of the vertices. If the number of elements at a vertex `IVT` is smaller than `NVEL`, in particular at the boundary, the remaining entries of `KVEL` are filled with `0`. The array `KVEL` is generated by routine `S2V`.

`LMEL` `DIMENSION KMEL(2,NEL)`
Array containing the numbers of the elements meeting at an edge.

This number is 2 for interior edges and 1 on the boundary. The array KMEL is generated by routine S2M.

LNPR DIMENSION KNPR(NVT+NMT)
Array containing information about the location of vertices and edges (nodal properties)

KNPR(IP) = 0	if IP is the number of an interior vertex or an interior edge and the adjacent elements are regularly refined (see below).
= IBCT	if IP is the number of a vertex on boundary component IBCT.
= IV	if IP is the number of an edge on the boundary; in this case, IV denotes the number of the boundary vertex preceding IV in the mathematically positive sense.
= -IEL	if IP is the number of an interior vertex or edge, which is an irregular node for the adjacent element IEL.

For triangular elements, IEL is obtained by "green" subdivision, for quadrilaterals, IP is a "blind" node of element IEL.

LMM DIMENSION KMM(2,NBCT)
Array containing the numbers of the vertices with minimum and maximum parameter for each boundary component.
KMM is used during refinements of the mesh only.

LVBD DIMENSION KVBD(NVBD)
Array containing the numbers of the vertices on the boundary.
The numbers are sorted, first, with respect to the boundary component, and then, within a boundary component, with respect to the parameter. All numbers are stored on a one-dimensional array. KVBD is generated by the routines XSVEB/SVEB.

LEBD DIMENSION KEBD(NVBD)
Same as KVBD, but for the elements adjacent to the boundary.
To each element IVT in KVBD, there corresponds an entry in KEBD which is just the number of that element containing the boundary edge following the vertex IVT. KEBD is generated by the routines XSVEB/SVEB.

LBCT DIMENSION KBCT(NBCT+1)
Pointer vector for the arrays KVBD and KEBD.
KBCT(IBCT) points to the position of the first entry in these arrays belonging to boundary component IBCT and KBCT(NBCT+1) is set to NVBD+1. KBCT is generated by the routines XSVEB/SVEB.

LVBDP DIMENSION DVBDP(NVBD)
Array containing the parameter values of boundary vertices. DVBDP is generated by the call of routine S.D03 in XS.0X (see Section 3.2).

LMBDP DIMENSION DMBDP(NVBD)
Array containing the parameter values of boundary midpoints. DMBDP is generated by the call of routine S.D03 in XS.0X (see Section 3.2).

Classification of the above quantities

a) Essential descriptors for a subdivision (always to be provided)
Scalars: `NEL`, `NVT`, `NVE`, `NBCT`
Arrays: `DCORVG`, `KVERT`, `KNPR(NVT)`

b) Optionally, for treatment of elements using information on midpoints of edges
Scalars: `NMT`
Arrays: `KMID`, `KNPR(NVT+NMT)`

c) Optionally, for generation of subdivision
Scalars: –
Arrays: `KADJ`, `KMM`

d) Optionally, for (simple) treatment of boundary conditions
Scalars: `NVBD`
Arrays: `KVBD`, (`KEBD`), `KBCT`

e) Optionally, for evaluation of boundary integrals
Scalars: `NVBD`
Arrays: `KVBD`, (`KEBD`), `KBCT`, `KVBDP`

f) Optionally, for several postprocessing algorithms
Scalars: `NVEL`
Arrays: `KVEL`, (`KMEL`)

g) Optionally, for treatment of isoparametric elements
Scalars: `NMT`
Arrays: `DCORMG`

Further, note that usually all entries of `KNPR` are greater or equal `0`, unless special care is taken to deal with irregular meshes, in particular in the case of quadrilateral elements ("blind nodes").

Example

Domain: Unit square $\Omega = (0,1)^2$ – Parametrization of the boundary from `0D0` to `4D0`.

Contents of variables and arrays described above.

Scalars

```
NEL    4   NVT    5   NMT    8   NVE   3
NVEL   4   NBCT   1   NVBD   4
```

Arrays

```
DCORVG
 0.0D0   .5D0
 1.0D0  1.5D0
 2.0D0  2.5D0                        DCORVG(1,IVT)  DCORVG(2,IVT)
 3.0D0  3.5D0
  .5D0   .5D0
```

If no information on midpoints is generated the second components in the first four lines are set to `0D0`.

`DCORMG` (only generated for isoparametric elements)

```
 0.50D0  0.00D0
 0.75D0  0.25D0
 0.25D0  0.25D0
 1.00D0  0.50D0                       DCORMG(1,IVT)  DCORMG(2,IVT)
 0.75D0  0.75D0
 0.50D0  1.00D0
 0.25D0  0.75D0
 0.00D0  0.50D0
```

```
KVERT
 1  2  5  0
 2  3  5  0                           KVERT(IVE,IEL), IVE=1,...,NNVE
 3  4  5  0
 4  1  5  0
```

```
KMID
  6   7   8  0
  9  10   7  0                        KMID(IVE,IEL), IVE=1,...,NNVE
 11  12  10  0
 13   8  12  0
```

```
KADJ
 0  2  4  0
 0  3  1  0                           KADJ(IVE,IEL), IVE=1,...,NNVE
 0  4  2  0
 0  1  3  0
```

```
KVEL
 1  4  0  0
 2  1  0  0
 3  2  0  0                           KVEL(NVEL,IVT), IVT=1,...,NVT
 4  3  0  0
 1  2  3  4
```

```
KMEL
 1  0
 2  1
 1  4
 2  0                                 KMEL(2,IMT), IMT=1,...,NMT
 2  3
 3  0
 3  4
 4  0
```

```
KNPR
 1  1  1  1  0
 1  0  0  2  0                   KNPR(IVMT), IVMT=1,...,NVT+NMT
 3  0  4

KMM
 1  4                            KMM(IMM,NBCT), IMM=1,2

KVBD
 1  2  3  4                      KVBD(IVBD), IVBD=1,...,NVBD

KEBD
 1  2  3  4                      KEBD(IVBD), IVBD=1,...,NVBD

KBCT
 1  5                            KBCT(IBCT), IBCT=1,...,NBCT+1
```

The numbers of the midpoints and the contents of the optional arrays coincide with that generated by the routines S2M, S2V, and SVEB on the basis of the essential arrays DCORVG, KVERT, and KMM.

Macro–triangulations

The variables and arrays described in this section contain all the information needed for the administation of macro-triangulations. The final grid results from a uniform subdivision of the macro-elements in, for example, NFINE**2 elements. The structure of the two COMMON blocks belonging to that task is very similar to the already described COMMON blocks /TRIAA/ and /TRIAD/.

```
      COMMON /MACROD/ NMAEL,NMAVT,NMAEDG,NMAVE,NMAVEL,NMABCT,NMAVBD
      COMMON /MACROA/ LMACVG,LMACMG,LMAVT,LMAMID,LMAADJ,LMAVEL,LMAMEL,
     *                LMANPR,LMAMM,LMAVBD,LMAEBD,LMABCT,LMAVBP,LMAMBP,
     *                LMAVE
```

Explanation

a) Contents of /MACROD/ – Dimensions

NMAEL	Total number of macro-elements
NMAVT	Total number vertices of the macro-elements
NMAEDG	Total number of macro-edges

`NMAVE` Number of vertices per macro-element
`NMAVE` is set to `3` for purely triangular meshes and is set to `4` for quadrilateral meshes and meshes formed by triangles and quadrilaterals. In the present version no meshes of mixed type are supported.

`NMAVEL` Maximum number of macro-elements meeting at one vertex
`NMAVEL` is set to `0` if this information is not available. `NMAVEL` is set by the routines `XSMA2V/SMA2V`.

`NMABCT` Total number of boundary components,
i.e., the number of simply closed curves describing the boundary of the domain (see Section 1.4). `NMAVBD` is calculated by routine `XSMAS`.

`NMAVBD` Total number of macro-vertices on the boundary (sum over all boundary components). `NMAVBD` is calculated by routine `XSMAS`.

b) Contents of `/MACROA/` – Array descriptors

We give a list of the array descriptors and the corresponding arrays together with their effective dimension.

`LMACVG` `DIMENSION DMACVG(2,NMAVT)`
Array containing information about the coordinates of the macro-vertices
`DMACVG(1,IVT)` – X-coordinate of vertex `IVT`
`DMACVG(2,IVT)` – Y-coordinate of vertex `IVT`

`LMACMG` `DIMENSION DMACMG(2,NMAEDG)`
Array containing the cartesian coordinates of the midpoints of macro-edges
This information is needed only for isoparametric meshes using quadratic transformation onto the reference element. In the present version of FEAT2D `LMACMG` is set to `0`.

`LMAVT` `DIMENSION KMAVT(NNVE,NMAEL)`
Array containing the numbers of vertices for each element in counterclockwise sense
Convention: For triangular elements, the last component `KMAVT(NNVE,.)` is set to `0`.

`LMAMID` `DIMENSION KMAMID(NNVE,NMAEL)`
Array containing the numbers of midpoints of macro-edges for each element in counterclockwise sense; the midpoints are given numbers ranging from `1` to `NMAEDG`.
Convention: For triangular elements, the last component `KMAMID(NNVE,.)` is set to `0`.

`LMAADJ` `DIMENSION KMAADJ(NNVE,NMAEL)`
Array containing the numbers of the neighboring elements for each macro-element of the triangulation.
Same convention for enumeration as for midpoints of edges.
For triangular elements, the last component `KMAADJ(NNVE,.)` is set to `0`.

`LMAVEL` `DIMENSION KMAVEL(NMAVEL,NMAVT)`
Array containing the numbers of the macro-elements meeting at a macro-vertex.

Remember that NMAVEL is the maximum number of elements meeting at one of the vertices. If the number of elements at a vertex IVT is smaller than NVEL, in particular at the boundary, the remaining entries of KMAVEL are filled with 0. The array KMAVEL is generated by routine S2V.

LMAMEL DIMENSION KMAMEL(2,NMAEL)
Array containing the numbers of the macro-elements meeting at a macro-edge
This number is 2 for interior edges and 1 on the boundary. The array KMAMEL is generated by routine S2M.

LMANPR DIMENSION KMANPR(NMAVT+NMAEDG)
Array containing information about the location of macro-vertices and macro-edges (nodal properties, see description of KNPR)

LMAMM DIMENSION KMAMM(2,NMABCT)
Array containing the numbers of the macro-vertices with minimum and maximum parameter for each boundary component.

LMAVBD DIMENSION KMAVBD(NMAVBD)
Array containing the numbers of the macro-vertices on the boundary. KMAVBD is generated by the routines XSVEB/SVEB.

LMAEBD DIMENSION KMAEBD(NMAVBD)
Same as KMAVBD, but for the elements adjacent to the boundary.
To each element IVT in KMAVBD, there corresponds an entry in KMAEBD which is just the number of that element containing the boundary edge following the vertex IVT. KMAEBD is generated by the routines XSVEB/SVEB.

LMABCT DIMENSION KMABCT(NMABCT+1)
Pointer vector for the arrays KMAVBD and KMAEBD.
KMABCT(IBCT) points to the position of the first entry in these arrays belonging to boundary component IBCT and KMABCT(NMABCT+1) is set to NMAVBD+1. KMABCT is generated by the routines XSVEB/SVEB.

LMAVBP DIMENSION DMAVBP(NMAVBD)
Array containing the parameter values of boundary vertices. DMAVBP is generated by the call of routine S.D13 in XS.1X (see Section 3.2).

LMAMBP DIMENSION DMAMBP(NMAVBD)
Array containing the parameter values of boundary midpoints. DMAMBP is generated by the call of routine S.D13 in XS.1X (see Section 3.2).

LMAVE DIMENSION KMAVE(2,NMAEDG)
KMAVE(1,IEDGE) and KMAVE(2,IEDGE) are the numbers of the two macro-vertices of macro-edge IEDGE, where the entry KMAVE(1,IEDGE) is smaller than KMAVE(2,IEDGE).

Multigrid data

In this last subsection we present two COMMON blocks that are needed in multigrid applications. They contain the grid information of /TRIAA/ and /TRIAD/ for at most NNLEV refinement levels. The meaning of the variables can be deduced straightforward from the contents of /TRIAA/ and /TRIAD/ (for further information see Section 3.2).

```
      PARAMETER (NNLEV=9)

      COMMON /MGTRD/  KNEL(NNLEV),KNVT(NNLEV),KNMT(NNLEV),
     *                KNVEL(NNLEV),KNVBD(NNLEV)
      COMMON /MGTRA/  KLCVG(NNLEV),KLCMG(NNLEV),KLVERT(NNLEV),
     *                KLMID(NNLEV),KLADJ(NNLEV),KLVEL(NNLEV),
     *                KLMEL(NNLEV),KLNPR(NNLEV),KLMM(NNLEV),
     *                KLVBD(NNLEV),KLEBD(NNLEV),KLBCT(NNLEV),
     *                KLVBDP(NNLEV),KLMBDP(NNLEV)
```

1.6. Storage techniques for matrices

FEAT2D provides routines for the evaluation of finite element matrices and for basic linear algebra operations with respect to several storage techniques, in particular taking care for the sparse structure. Not all techniques described below are supported in the present version. The different storage methods are indicated by a reference character `0...9, A...Z` which again occurs in the name of the corresponding subprograms, for example for calculating stiffness matrices or for forming matrix-vector products.

The array containing the entries of the matrix is denoted by `DA` or `VA`, depending on the accuracy. Further, we use the notation `NA` for the number of entries stored in `DA` (`VA`) and `NEQ` for the number of equations. For rectangular matrices (storage techniques `1` and `9`), `NEQ` denotes the number of rows.

Storage technique		Array descriptors	Explanation
0	Matrix not stored	None	All matrix operations are performed by means of `EXTERNAL` subroutines
1	Full matrix	`DA(NEQ,NEQ1)`	Usual full storage, `DA(I,J)` Elements are stored columnwise
2	Full matrix symmetric	`DA(NEQ*(NEQ+1)/2)`	Only upper triangular matrix is stored on a vector, columnwise

3	Sparse band matrix	`DA(NEQ*NDIA)`
		`KDIA(NDIA)`
		`KDIAS(NDIA+1)`
		`NDIA`
			Elements of `NDIA` nonzero subdiagonals are stored onto a matrix, each subdiagonal is stored with length `NEQ`, `KDIA` contains the distance to the main diagonal, the main diagonal is stored first (`KDIA(1)=0`), followed by the lower triangular part (`KDIA(.)<0`)

4	Sparse band matrix symmetric	`DA(NEQ*(NDIA+1)/2)`
		`KDIA(NDIA)`
		`KDIAS(NDIA+1)`
		`NDIA`
			Same as technique 3, but only upper triangular part is stored

5	Skyline technique	`DA(NA)`	
6	Skyline technique symmetric	`DA(NA)`	

7	Compact storage Standard technique for quadratic matrices	`DA(NA)`
		`KCOL(NA)`
		`KLD(NEQ+1)`
			The (nonzero) elements of the matrix are stored, row by row, on the vector `DA`. For each row, the diagonal entry is stored first. `KCOL` contains the column index for each element in `DA`. `KLD(IEQ)` contains the position of the `IEQ`-th diagonal element, i.e. `KLD` points to the start of row `IEQ` in `DA`, `KLD(NEQ+1)=NA+1`

Storage technique		Array descriptors	Explanation
8	Compact storage symmetric	DA(NA) KCOL(NA) KLD(NEQ+1)	Same as technique 7 Only upper triangular matrix stored. For each row, the diagonal entry is stored first.
9	Compact storage sparse rectangular matrix	DA(NA) KCOL(NA) KLD(NEQ+1)	Same as technique 7, but "diagonal" entry is not stored first for each row, NEQ denotes number of rows
A	Operator technique	DA(NA) KCOL(NA) KLD(NOP+1) KOP(NEQ)	Matrix rows with the same elements at the same position with respect to the main diagonal are only stored once (typical for finite difference matrices, constant coefficients). NOP denotes the number of rows stored in DA. DA contains the entries, in each row the diagonal entry is stored first. KCOL contains the distance of an element to the diagonal, KLD points to the start of a new row. KOP(IEQ) contains the number of the row in DA which forms the row IEQ of the real matrix.

Example

As an example, for the different techniques, above, the matrix elements and pointer vectors of the following symmetric "sparse" and "banded" 4×4-matrix A_1 in DOUBLE PRECISION are presented.

$$A_1 = \begin{pmatrix} 1 & 2 & 0 & 7 \\ 2 & 4 & 3 & 0 \\ 0 & 3 & 6 & 5 \\ 7 & 0 & 5 & 8 \end{pmatrix}$$

Technique 1

```
DA:   1D0 2D0 0D0 7D0 2D0 4D0 3D0 0D0
      0D0 3D0 6D0 5D0 7D0 0D0 5D0 8D0
```

Pointer vectors not needed

Technique 2

```
DA:   1D0 2D0 4D0 0D0 3D0 6D0 7D0 0D0 5D0 8D0
```

Pointer vectors not needed

Technique 3

```
DA:     1D0 4D0 6D0 8D0 7D0 2D0 3D0 5D0 2D0 3D0 5D0 7D0
KDIA:    0  -3  -1   1   3
KDIAS:   1   5   6   9  12  13
NDIA:    5
NA:     12
```

Technique 4

```
DA:     1D0 4D0 6D0 8D0 2D0 3D0 5D0 7D0
KDIA:    0   1   3
KDIAS:   1   5   8   9
NDIA:    3
NA:      8
```

Technique 5 and 6 not yet available

Technique 7

DA: 1D0 2D0 7D0 4D0 2D0 3D0 6D0 3D0 5D0 8D0 7D0 5D0

KCOL: $\underline{1}$ 2 4 $\underline{2}$ 1 3 $\underline{3}$ 2 4 $\underline{4}$ 1 3

KLD: 1 4 7 10 13

NA: 12

Underlined numbers indicate new row

Technique 8

DA: 1D0 2D0 7D0 4D0 3D0 6D0 5D0 8D0

KCOL: $\underline{1}$ 2 4 $\underline{2}$ 3 $\underline{3}$ 4 $\underline{4}$

KLD: 1 4 6 8 9

NA: 8

Underlined numbers indicate new row

Modified example for storage technique 9 – rectangular matrix

$$A_2 = \begin{pmatrix} 1 & 2 & 0 & 7 & 9 \\ 2 & 4 & 3 & 0 & 9 \\ 0 & 3 & 6 & 5 & 9 \\ 7 & 0 & 5 & 8 & 9 \end{pmatrix}$$

Technique 9

DA: 1D0 2D0 7D0 9D0 2D0 4D0 3D0 9D0
3D0 6D0 5D0 9D0 7D0 5D0 8D0 9D0

KCOL: $\underline{1}$ 2 4 5 $\underline{1}$ 2 3 5
$\underline{2}$ 3 4 5 $\underline{1}$ 3 4 5

KLD: 1 5 9 13 9

NA: 16

Modified example for technique A – Operator technique

Row 2 and 3 of the matrix A_3 have the same structure and are stored only once.

$$A_3 = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 2 & 1 \end{pmatrix}$$

Technique A

DA: 2D0 1D0 4D0 1D0 1D0 1D0 2D0

KCOL: $\underline{0}$ 1 $\underline{0}$ -1 1 $\underline{0}$ -1

KLD: 1 3 6 8

KOP: 1 2 2 3

NA: 7

NOP: 3

1.7. Messages, error handling and consistency checks

The COMMON blocks described below contain the necessary information to display protocol and error messages. The DATA statements are part of the BLOCK DATA subprogram ZVALUE and indicate the default values (see Section 3.1 for more information on ZVALUE).

```
CHARACTER SUB*6,FMT*15,CPARAM*120

COMMON /OUTPUT/ M,MT,MKEYB,MTERM,MERR,MPROT,MSYS,MTRC,IRECL8
COMMON /ERRCTL/ IER,ICHECK
COMMON /CHAR/   SUB,FMT(3),CPARAM

DATA M/2/,MT/2/,MKEYB/5/,MTERM/6/,IER/0/,ICHECK/1/
DATA MERR/11/,MPROT/12/,MSYS/13/,MTRC/14/,IRECL8/128/
```

1. Messages

The FORMAT statements for all messages displayed by FEAT2D are contained in the file FEAT.MSG. The messages are divided in three groups.

- Protocol messages of informative character (.PROT),
- System messages, e.g.messages about allocation, deletion, and resizing of arrays on DWORK (.SYS),
- Error messages, usually leading to termination of the program (.ERR).

Corresponding to the different groups, the messages are sent to the output files connected with the actual unit numbers MPROT, MSYS, and MERR. These files are opened and linked to the correct unit numbers by the system initialization routine ZINIT. The user may decide not to split the information onto several files and overwrite the default values for the output units. In addition, the messages can appear on the screen.

The user can control the amount of output by choosing the values for the output levels M and MT in the COMMON block /OUTPUT/. The level M refers to the output to file and MT refers to the output to the standard output device MTERM. The message file FEAT.MSG contains two level numbers, separately for each message. The corresponding message is sent only if M resp. MT are greater than or equal these values.

For example, system messages are displayed on the screen only if MT is at least 3, but they are sent to MSYS already for M>1. Error messages are sent to MERR even for M=0. In most cases, it is sufficient to choose M=MT=1 or even 0.

2. Error handling

If an internal error occurs in a FEAT2D subprogram the error routine `WERR` is invoked. `WERR` calls the subprogram `OERR` for error messages and sets the error indicator in the `COMMON` block `/ERRCTL/` to the (negative) number of the error. If any FEAT2D routine returns a negative error indicator one should immediately stop the program. In some programs, `IER` may also be set to some positive value. For example, the value `IER=1` in an iterative solution algorithm indicates that the desired accuracy has not been achieved. In this case, the user may decide to continue the program. Before it starts working, each subprogram overwrites the variable `SUB` in the `COMMON` block `/CHAR/` by its own name.

```
SUB='nnnnnn'
```

Therefore the user can decide in which subprogram the error occurred even if no messages have been displayed. For the next release of FEAT2D it is planned to improve the error handling. For example, it will be possible to dump the names and size of all arrays on the workspace at the moment when the error occurred.

3. Subprogram tracing and consistency checks

The second parameter `ICHECK` in the error control block `/ERRCTL/` is used to decide which levels of checks are performed during the program. For `ICHECK=1`, only elementary consistency checks take place. For example, the data type of arrays passed to a subprogram is checked or the area and orientation of each element is controlled during the calculation of a finite element matrix. Moreover, `ICHECK` controls the optional tracing of the FEAT2D subprograms. The name and the date of their version is written to the file connected with the unit `MTRC`.

The user may want his own subprograms to be traced in the same manner as it is done for the FEAT2D routines. Then, the first executable statements of a program unit should look as follows:

```
SUB='nnnnnn'
IF (ICHECK.GE.997) CALL OTRC('nnnnnn','date')
IER=0
```

Here, nnnnnn stands for the name of the routine and date is a string containing the date of the last update in the form mm/dd/yy. Clearly the `COMMON` blocks `/ERRCTL/` and `/CHAR/` must also be defined. FEAT2D routines are traced in this way for values `ICHECK=997`, `998`, or `999`. For the value `997`, only the most important subprograms are traced, for the value `999` even elementary auxiliary subroutines. If the user decides to trace only his own subprograms he has to use values for `ICHECK` smaller than `997`. For fully tested programs one should choose `ICHECK=0`.

1.8. BLAS routines

Many of the subprograms of the subgroup `L` deal with elementary linear algebra like forming the dot product or calculating linear combinations of two vectors. In order to speed up execution time on many of the larger machines, in particular on vector computers, these tasks are realized by calling the Basic Linear Algebra Subprograms (BLAS). Since most of the matrix operations in the package are for extremely sparse matrices, here we only use the BLAS routines of level 1 (vector operations).

Here, we give a short list of the subprograms needed in the package which should be replaced by coded routines if they are available.

`DOUBLE PRECISION`

`DCOPY`	Copies a vector
`DSCAL`	Scales a vector by a constant
`DAXPY`	Forms linear combination $y := y + ax$
`DNRM2`	Calculates the mean square norm of a vector
`IDAMAX`	Finds the index of the (first) element with maximum modulus
`DDOT`	Forms the inner product of two vectors

`REAL`

`SCOPY`	
`SSCAL`	
`SAXPY`	Same tasks as above
`SNRM2`	
`ISAMAX`	
`SDOT`	

For the remaining basic linear algebra tasks which are not directly supported by the BLAS like clearing a vector (filling with zeroes, see Section 1.3) we use unrolled loops unless we work on vector machines. For adapting FEAT2D to a new machine typically only the `L`-routines have to be optimized.

2. Common Program Segments

In this section we give an example for standard declarations and give an explanation of the internally used `COMMON` blocks.

2.1. Type declarations

```
      IMPLICIT DOUBLE PRECISION (A,C-H,O-U,W-Z),LOGICAL(B)
      CHARACTER SUB*6,FMT*15,CPARAM*120
```

The quantities of type `CHARACTER` are used in `COMMON /CHAR/`, described below.

2.2. `PARAMETER` statements

```
      PARAMETER (NNWORK=300000)
      PARAMETER (NNARR=299,NNLEV=9)
      PARAMETER (NNVE=4,NNBAS=21,NNCUBP=36,NNAB=21,NNDER=6)
```

<u>Explanation</u>

`NNWORK`	Maximum number of `DOUBLE PRECISION` elements on the workspace Only used in the main program for dimensioning and initialization
`NNARR`	Maximum number of arrays allocated on the workspace (see description of pseudodynamic memory management, Section 1.3)
`NNLEV`	Maximum number of multigrid levels – Set to `9`
`NNVE`	Maximum number of vertices per element – Set to `4`
`NNBAS`	Maximum number of local degrees of freedom Set to `21` for the quintic Argyris element
`NNCUBP`	Maximum number of cubature points in numerical integration formulas – Set to `36`
`NNAB`	Maximum number of additive terms in integrands of bilinear forms or linear forms (see description of routines `AB..` and `VB.`, Sections 3.2 and 3.2)
`NNDER`	Maximum number of combinations of derivatives applied to basis functions, `NNDER=6` means derivatives up to the order 2 (see description of element routines `E...`, Section 3.2)

2.3. EQUIVALENCE statement

```
      DIMENSION VWORK(1),KWORK(1)

      EQUIVALENCE (DWORK(1),VWORK(1),KWORK(1))
```

Used only in X-, Y-, and Z-routines, and possibly in the main program to keep arrays of type DOUBLE PRECISION, REAL, and INTEGER on the same workspace vector (see the description of the pseudodynamic memory management in Section 1.3).

2.4. COMMON blocks

Here, we give a complete list and explanation of all the COMMON blocks used for internal communication of the FEAT2D routines. Not all of the blocks are defined in each subprogram.

In particular, the block /TABLE/ containing the essential information about the workspace is only known to the subroutines of the group Z, described in Section 1.3. It should never be defined in user subprograms.

List of COMMON blocks

```
      IMPLICIT DOUBLE PRECISION (A,C-H,O-U,W-Z),LOGICAL(B)
      CHARACTER SUB*6,FMT*15,CPARAM*120
C
      PARAMETER (NNARR=299,NNLEV=9)
      PARAMETER (NNVE=4,NNBAS=21,NNCUBP=36,NNAB=21,NNDER=6)
C
      COMMON          NWORK,IWORK,IWMAX,L(NNARR),DWORK(1)
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /TRIAA/  LCORVG,LCORMG,LVERT,LMID,LADJ,LVEL,LMEL,LNPR,LMM,
     *                LVBD,LEBD,LBCT,LVBDP,LMBDP
      COMMON /MACROD/ NMAEL,NMAVT,NMAEDG,NMAVE,NMAVEL,NMABCT,NMAVBD
      COMMON /MACROA/ LMACVG,LMACMG,LMAVT,LMAMID,LMAADJ,LMAVEL,LMAMEL,
     *                LMANPR,LMAMM,LMAVBD,LMAEBD,LMABCT,LMAVBP,LMAMBP,
     *                LMAVE
      COMMON /MGTRD/  KNEL(NNLEV),KNVT(NNLEV),KNMT(NNLEV),
     *                KNVEL(NNLEV),KNVBD(NNLEV)
      COMMON /MGTRA/  KLCVG(NNLEV),KLCMG(NNLEV),KLVERT(NNLEV),
     *                KLMID(NNLEV),KLADJ(NNLEV),KLVEL(NNLEV),
     *                KLMEL(NNLEV),KLNPR(NNLEV),KLMM(NNLEV),
     *                KLVBD(NNLEV),KLEBD(NNLEV),KLBCT(NNLEV),
     *                KLVBDP(NNLEV),KLMBDP(NNLEV)
      COMMON /OUTPUT/ M,MT,MKEYB,MTERM,MERR,MPROT,MSYS,MTRC,IRECL8
      COMMON /ERRCTL/ IER,ICHECK
```

```
      COMMON /CHAR/   SUB,FMT(3),CPARAM
      COMMON /ELEM/   DX(NNVE),DY(NNVE),DJAC(2,2),DETJ,
     *                DBAS(NNBAS,NNDER),BDER(NNDER),KVE(NNVE),IEL
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
      COMMON /COAUX1/ KDFG(NNBAS),KDFL(NNBAS),IDFL
      COMMON /COAUX2/ DBAS1(NNBAS,NNDER,3),KDFG1(NNBAS,3),
     *                KDFL1(NNBAS,3),IDFL1(3)
      COMMON /TABLE/  KTYPE(NNARR),KLEN(NNARR),KLEN8(NNARR),IFLAG
```

Explanation

The blank COMMON contains workspace information and is described in detail in Section 1.3 of this manual.

/TRIAA/ contains the numbers of all arrays, describing the (current) triangulation.

/TRIAD/ contains all information about dimensions of the (current) triangulation. For more details see Sections 1.4 and 1.5.

The same kind of information for macro-triangulations is contained in the COMMON blocks /MACROD/ and /MACROA/ (see detailed description in Section 1.5).

For multigrid applications the data of the COMMON blocks /TRIAA/ and /TRIAD/ is contained in the COMMON blocks /MGTRA/ and /MGTRD/, respectively, for each of the at most NNLEV levels (see Section 3.2).

/OUTPUT/ contains information on output level and I/O units, see Section 1.7.

/ERRCTL/ The variable IER contains the current error indicator, ICHECK is used to control consistency checks and tracing of the subprograms, see Section 1.7.

/CHAR/ contains common quantities of type CHARACTER: SUB is the name of the last routine called (used for error tracing), FMT contains formats for normalized output of arrays and subdivisions (routines XORA and XOWA), and CPARAM is used to pass arguments to the message routines OMSG and OERR.

Note: Only quantities in the COMMON blocks /OUTPUT/, /ERRCTL/ or /CHAR/ should be changed in user provided programs.

/ELEM/ contains information of the geometry of the current element and other information needed for the evaluation of the finite element basis functions, group E. The cartesian coordinates of the vertices of the element in process, IEL, are contained in DX and DY. DJAC denotes the Jacobian of transformation to the reference element and DETJ its determinant at the evaluation point. BDER(I)=.TRUE. means that the derivative I has to be calculated. The convention is

I=1	Function value
I=2	First x-derivative
I=3	First y-derivative
I=4	Second x-derivative
I=5	Mixed xy-derivative
I=6	Second y-derivative

KVE contains the numbers of the vertices, used for determination of the direction of normal vectors on interelement boundaries. ICUBP is the number of current cubature point and DBAS contains the values of the basis functions and derivatives.

/CUB/ contains information about rules for numerical cubature or quadrature. The cartesian coordinates of the integration points on the reference elements $[-1, 1]$ and $[-1, 1]^2$ and the barycenric coordinates on the reference triangle are available in DXI. DOMEGA contains the corresponding weights and NCUBP denotes the total number of cubature points. All these values are determined in the subroutines of the group C and are needed for assembling finite element matrices and vectors. These latter routines (groups A and V) also set the final value, ICUBP, to the number of the current cubature point when communicating with the element library (group E). This information is also useful for the treatment of nonlinear problems.

/COAUX1/ or /COAUX2/ are defined during the assembly of finite element matrices of vectors (groups A and V) for the evaluation of the coefficient functions in the nonlinear case. The arrays KDFG (KDFG1) contain the global degrees of freedom and KDFL (KDFL1) the corresponding local d.o.f. on the current element. IDFL (IDFL1) denotes the total number of the d.o.f. per element. The block /COAUX1/ is used in the subroutines for the calculation of quadratic matrices and vectors where only one type of finite elements is needed. In the routine AB09 up to three finite elements are needed to evaluate the coefficient function in the nonlinear case. Here, we use the COMMON block /COAUX2/ which then also contains the values of the desired function values and derivatives for all basis functions in the array DBAS.

/TABLE/ is needed for the communication of those Z-routines that control allocation of arrays on the workspace vector. For each of the arrays (maximum number NNARR), KTYPE contains the data type (1, 2 or 3), KLEN the length, and KLEN8 the number of DOUBLE PRECISION locations needed on DWORK. IFLAG is used for internal communication between ZNEW and ZDISP. The COMMON block /TABLE/ should never be defined in another subprogram.

2.5. SAVE statement

At least the names of all COMMON blocks occurring in a program unit must be saved. The same holds true for local variables which shall keep their value until the next call of the routine.

Example

```
SAVE /TRIAA/,/TRIAD/,/OUTPUT/,/ERRCTL/,/CHAR/
```

3. Description of the Subprograms

In the following sections, we shall describe in detail the parameter lists and the tasks of most of the subprograms of FEAT2D. We shall begin with the most important groups, the routines with starting letters `X`, `Y`, and `Z`. These are the only subprograms which are used for the administration of the workspace vector. In most of the applications, only the subprograms of the groups `X` and `Z` are directly called by the user.

3.1. The groups X, Y, and Z

Group X – Communication between user programs and "working" routines Preparation of parameter lists for programs A-W

As already indicated in the section on the pseudodynamic memory management, most or all of the larger arrays are placed on the workspace vector. In the main program and/or in the higher user provided subprograms all of these arrays are known only by their number which is provided by the subroutine `ZNEW` (see Section 1.3). For example, the `DOUBLE PRECISION` array `DNAME` cannot be addressed directly but only by its number contained in the corresponding `L`-variable `LNAME`.

Dealing directly with the administration of the workspace on all levels makes user provided programs more or less unreadable. For instance, it is much more convenient to use the number `LNAME` in a `CALL` to a subroutine than the effective address `DWORK(L(LNAME))`, or even `DWORK(L(LNAME(1)))` in the case that `LNAME` itself is an array of tokens for several matrices. In FEAT2D, the working subroutines usually are not called directly but via an intermediate subprogram which handles the pseudodynamic memory management. The name of these intermediate programs starts with the letter `X`, followed by an indicative name characterizing the "working" routine called. Therefore, the second character in the subprogram name characterizes the group of tasks, see Section 1.1. For example, the subprogram `LSP1` forms the dot product of two vectors (group linear algebra). In the higher level programs the user will usually invoke this program by calling `XLSP1` which only requires the numbers of the two vectors as an argument.

```
SUBROUTINE XLSP1(LX,LY,NXY,SP)
...
CALL LSP1(DWORK(L(LX)),DWORK(L(LY)),NXY,SP)
END

SUBROUTINE LSP1(DX,DY,NXY,SP)
```

```
      END
```

Frequently, the structure of the X-routines is much more complex. Typical additional tasks of this group of routines are the following

- Determination of problem dependent array dimensions and allocation of arrays on the workspace,
- Check the size and the datatype of arrays, enlargement of arrays if required,
- Allocation and deletion of temporary auxiliary vectors (e.g. for CG-algorithms),
- Normalized I/O of arrays depending on the datatype.

For example, consider the subroutines XAP7 and XAA07 (XAB07) which are used to calculate finite element matrices for a given triangulation. First, one invokes XAP7 to determine the pointer vectors KCOL and KLD which are needed to describe the structure of the matrix in storage technique 7. XAP7 calculates the global number of degrees of freedom and allocates the vector KLD on DWORK in the correct length NEQ+1. Since the number of nonzero entries in the matrix is not known at that stage, XAP7 reserves the remaining free space of the workspace for the calculation of the pointer vector KCOL. Then, the two pointer vectors are filled by the subprogram AP7 which as a by-product calculates the total number of nonzero entries in the matrix.

Finally, XAP7 releases the space on DWORK that is not needed for KCOL for the subsequent programs. Only the numbers of the two pointer vectors, LCOL and LLD, and the dimensions NEQ and NA are returned to the calling routine.

In a second step one calculates the entries of the matrix by invoking XAA07, in the case of triangular elements, or XAB07, in the case of quadrilaterals. XAA07 (XAB07) allocates the matrix (which may consist of several blocks with the same pointer structure) on DWORK in the correct length NA each and then calls the routine AA07 (AB07). The latter routine calculates the element matrices and assembles the global matrix (for each block). Only the numbers LA(IBLOC) for each block matrix are returned to the routine that has invoked XAA07 (XAB07).

Since the function of the X-routines is closely related to the "working" routines, we shall describe the parameter lists in more detail together with the corresponding groups A-W.

Group Y – Handling of direct communicaton for lower level subroutines Preparation of parameter lists for routines A-W

All subprograms of FEAT2D use direct communication to exchange information. The routines of the Y-group are used to handle the difficulties usually occurring whenever low level subprograms need information on parameters which are not available in the calling routines.

As an example consider the implementation of the preconditioned conjugate gradient algorithm. The subprogram IE010 only knows the solution vector and the right hand side but not the system matrix. Subprograms forming the necessary matrix vector product and performing the preconditioning step are passed to `IE010` as `EXTERNAL` routines.

Assume that the system matrix is stored in technique 7. The linear algebra routine performing the matrix vector multiplication has the name `LAX17`. However, this function cannot be used as an `EXTERNAL` routine in `IE010`, since it needs as its arguments the system matrix and the corresponding pointer vectors which are not known to IE010. This difficulty is handled by passing the subprogram `YLAX17` to `IE010` instead of `LAX17` itself. `YLAX17` takes the `L`-numbers `LA`, for the matrix, and `LCOL`, `LLD`, for the pointer vectors, from a standard `COMMON` block. Then, it invokes `LAX17` with the correct arguments `DWORK(L(LA))`, `KWORK(L(LCOL))`, and `KWORK(L(LLD))`. The standard `COMMON` block for `YLAX17` is usually prepared by a user called `X`-routine.

In simple applications the user will not deal with `Y`-routines directly since they can completely be handled by the FEAT2D subprograms. In more complex algorithms, however, it will be necessary to modify routines to match the needs of the current application. For example, if one works with a block structured matrix one has to write his own routine for matrix vector multiplication, of course by combining the standard FEAT2D programs. We shall, therefore, sketch the interaction between `X`-routines, `Y`-routines, and the working programs again for the above mentioned example by the following abbreviated program segments.

Notice that the only subprogram which is directly called by the user is `XIE017`. (The parameter `OMEGA` in the following example is used in connection with SSOR-preconditioning only.)

```
      SUBROUTINE XIE017(LA,LCOL,LLD,LX,LB,NEQ,...)
      ...
      EXTERNAL YLAX17,...
      ...
      COMMON /XYPAR/  DXYPAR(NNARR),KXYPAR(NNARR)
      ...
      KXYPAR(1)=L(LA)
      KXYPAR(2)=L(LCOL)
      KXYPAR(3)=L(LLD)
      DXYPAR(1)=OMEGA
      ...
      CALL IE010(DWORK(L(LX)),DWORK(L(LB)),NEQ,...,YLAX17,...)
      ...
      END

      SUBROUTINE IE010(DX,DB,NEQ,...,DAX0,...)
      ...
      CALL DAX0(DX,DAX,NEQ,...)
      ...
      END

      SUBROUTINE YLAX17(DX,DAX,NEQ,...)
      ...
```

```
      COMMON /XYPAR/  DXYPAR(NNARR),KXYPAR(NNARR)
      ...
      CALL LAX17(DWORK(KXYPAR(1)),KWORK(KXYPAR(2)),
     *           KWORK(KXYPAR(3)),NEQ,DX,DAX,...)
      ...
      END
```

Because of the close relationship of Y-routines and the lower level routines we shall describe the parameter lists in more detail together with the groups A-W in Section 3.2.

Group Z **– Handling of the pseudodynamic memory management**
Machine dependent system routines

1. Pseudodynamic memory management

The first subgroup of subprograms has been described in detail in the section on the pseudodynamic memory management. For completeness, we list again the subprogram names and the parameter lists.

```
      SUBROUTINE ZCLEAR(LNR,ARR)
      SUBROUTINE ZCPY(LNR1,ARR1,LNR2,ARR2)
      SUBROUTINE ZCTYPE(ITYPE,LNR,ARR)
      SUBROUTINE ZDISP(ILONG,LNR,ARR)
      SUBROUTINE ZFREE(ITYPE,IFREE)
      SUBROUTINE ZLEN(LNR,LENGTH)
      SUBROUTINE ZLEN8(LNR,LENGTH)
      SUBROUTINE ZNEW(ILONG,ITYPE,LNR,ARR)
      SUBROUTINE ZTYPE(LNR,LTYPE)
```

2. Initialization of the BLANK COMMON **and of I/O devices**

```
      SUBROUTINE ZINIT(NNWORK,CMSG,CERR,CPRT,CSYS,CTRC)
      CHARACTER*(*)   CMSG,CERR,CPRT,CSYS,CTRC
      COMMON          NWORK,IWORK,IWMAX,L(NNARR),DWORK(1)
      COMMON /OUTPUT/ M,MT,MKEYB,MTERM,MERR,MPROT,MSYS,MTRC,IRECL8
```

3. BLOCK DATA **subprogram**

The following BLOCK DATA subprogram ZVALUE initializes the most important named COMMON blocks. For a complete list of all COMMON blocks see Appendix B.

```
      BLOCK DATA ZVALUE
C
      IMPLICIT DOUBLE PRECISION (A,C-H,O-U,W-Z),LOGICAL(B)
```

```
      CHARACTER SUB*6,FMT*15,CPARAM*1
      PARAMETER (NNARR=299,NNLEV=9)
      COMMON /OUTPUT/ M,MT,MKEYB,MTERM,MERR,MPROT,MSYS,MTRC,IRECL8
      COMMON /ERRCTL/ IER,ICHECK
      COMMON /CHAR/   SUB,FMT(3),CPARAM(120)
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /TRIAA/  LCORVG,LCORMG,LVERT,LMID,LADJ,LVEL,LMEL,LNPR,LMM,
     *                LVBD,LEBD,LBCT,LVBDP,LMBDP
      COMMON /TABLE/  KTYPE(NNARR),KLEN(NNARR),KLEN8(NNARR),IFLAG
      COMMON /MACROD/ NMAEL,NMAVT,NMAEDG,NMAVE,NMAVEL,NMABCT,NMAVBD
      COMMON /MACROA/ LMACVG,LMACMG,LMAVT,LMAMID,LMAADJ,LMAVEL,LMAMEL,
     *                LMANPR,LMAMM,LMAVBD,LMAEBD,LMABCT,LMAVBP,LMAMBP,
     *                LMAVE
      COMMON /MGTRD/  KNEL(NNLEV),KNVT(NNLEV),KNMT(NNLEV),
     *                KNVEL(NNLEV),KNVBD(NNLEV)
      COMMON /MGTRA/  KLCVG(NNLEV),KLCMG(NNLEV),KLVERT(NNLEV),
     *                KLMID(NNLEV),KLADJ(NNLEV),KLVEL(NNLEV),
     *                KLMEL(NNLEV),KLNPR(NNLEV),KLMM(NNLEV),
     *                KLVBD(NNLEV),KLEBD(NNLEV),KLBCT(NNLEV),
     *                KLVBDP(NNLEV),KLMBDP(NNLEV)
C
      DATA M/2/,MT/2/,MKEYB/5/,MTERM/6/,IER/0/,ICHECK/1/
      DATA MERR/11/,MPROT/12/,MSYS/13/,MTRC/14/,IRECL8/512/
      DATA SUB/'MAIN  '/
      DATA FMT/'(3D25.16)','(5E16.7)','(6I12)'/
      DATA CPARAM/120*' '/
      DATA NEL/0/,NVT/0/,NMT/0/,NVE/0/,NBCT/0/,NVEL/0/,NVBD/0/
      DATA LCORVG/0/,LCORMG/0/,LVERT/0/,LMID/0/,LADJ/0/,LVEL/0/,LMEL/0/,
     *     LNPR/0/,LMM/0/,LVBD/0/,LEBD/0/,LBCT/0/,LVBDP/0/,LMBDP/0/
      DATA KTYPE/NNARR*0/,KLEN/NNARR*0/,KLEN8/NNARR*0/,IFLAG/0/
      DATA NMAEL/0/,NMAVT/0/,NMAEDG/0/,NMAVE/0/,NMAVEL/0/,NMABCT/0/,
     *     NMAVBD/0/
      DATA LMACVG/0/,LMACMG/0/,LMAVT/0/,LMAMID/0/,LMAADJ/0/,LMAVEL/0/,
     *     LMAMEL/0/,LMANPR/0/,LMAMM/0/,LMAVBD/0/,LMAEBD/0/,LMABCT/0/,
     *     LMAVBP/0/,LMAMBP/0/,LMAVE/0/
      DATA KNEL/NNLEV*0/,KNVT/NNLEV*0/,KNMT/NNLEV*0/,
     *     KNVEL/NNLEV*0/,KNVBD/NNLEV*0/
      DATA KLCVG/NNLEV*0/,KLCMG/NNLEV*0/,KLVERT/NNLEV*0/,
     *     KLMID/NNLEV*0/,KLADJ/NNLEV*0/,KLVEL/NNLEV*0/,
     *     KLMEL/NNLEV*0/,KLNPR/NNLEV*0/,KLMM/NNLEV*0/,
     *     KLVBD/NNLEV*0/,KLEBD/NNLEV*0/,KLBCT/NNLEV*0/,
     *     KLVBDP/NNLEV*0/,KLMBDP/NNLEV*0/
C
      END
```

Explanation

`M` and `MT` are set to 2, producing relatively much output on `MPRT`, `MSYS`, and on the screen. `MKEYB` is set to the machine dependent value for standard input unit, e.g. `5` for IBM systems. MTERM is set to the machine dependent value for the standard output unit, e.g. 6 for IBM systems. The value ICHECK in the error control block is set to 1, i.e. as a default only elementary consistency checks are performed but no subprogram tracing.

The values 11-14 are used as default values for the I/O files for FEAT2D. IRECL8 denotes the maximum record length for format-free I/O (machine dependent, set to 512 by default).

4. Machine dependent system routines

The group Z of FEAT2D routines is also intended for system dependent subprograms. In the present version, only a routine for measuring CPU time is included.

```
SUBROUTINE ZTIME(T)
```

The parameter T returns the system time since the last call of ZTIME. At program start, time synchronization is performed by ZINIT.

3.2. The groups A-W

Group A – Bilinear forms and matrices

The subprograms of group A are used to calculate the pointer structure and the entries of global finite element matrices for a given triangulation and for a particular choice of elements. The entries a_{ij} of the matrices are of the form

$$a_{ij} = \int_\Omega \sum_{\alpha,\beta} c_{\alpha\beta}(x)\partial^\alpha \phi_i \partial^\beta \psi_j \, dx$$

or, for boundary integrals,

$$a_{ij} = \int_{\partial\Omega} \sum_{\alpha,\beta} c_{\alpha\beta}(s)\partial^\alpha \phi_i \partial^\beta \psi_j \, ds \, .$$

Here, ϕ_i and ψ_j denote the basis functions of the test space and the trial space (in this order!) which naturally may also coincide. The coefficients $c_{\alpha\beta}$ for multiindices α and β are given as `EXTERNAL` functions and, of course, are allowed to depend on the solution or its derivatives.

In the present version of FEAT2D matrices in the storage techniques `3` (or `4` in the symmetric case), `7` (or `8` in the symmetric case) and `9` (general rectangular matrices) are supported. The routines of this group, first, calculate the pointer vectors, see Section 1.6, and, then, assemble the global matrices, of course using a loop over all elements. In most applications the user will only need the `X`-routines which also allocate the necessary arrays on the workspace vector.

Names of the subprograms

The names of the programs which calculate the pointer vectors are of the form `APs`, where `P` stands for pointer and `s` for the storage technique of the matrix.

On the basis of this pointer structure, the routines `AAvs` and `ABvs` assemble the entries of the global matrices, where `AA..` is used for triangular elements and `AB..` for quadrilaterals. The character `v` denotes the version,

`v=0` for area integrals

`v=1` for boundary integrals.

The final character `s` is used to reference the storage technique for the matrix, see Section 1.6.

Subgroup AP – Pointer vectors

The programs AP3, AP7 and AP9 determine the pointer vectors for matrices in storage technique 3, 7 or 9. For storage techniques 3 and 7 it is assumed that the test and the trial space coincide and only one element subprogram is passed as an argument. Storage technique 9 is reserved for general rectangular matrices where the test and trial space may be different.

The program XAP3, first, determines the number of equations, NEQ, and then allocates three INTEGER vectors (KDIA, KDIAS and a help vector) of length 2*NEQ-1. After determination of the number of diagonal rows, NDIA, these arrays are compressed. The corresponding routines XAP7 and XAP9, first, determine the number of equations, NEQ, and allocate KLD on the workspace in the correct length, NEQ+1. Then, they determine the remaining free space of DWORK with the routine ZFREE to calculate the column pointer KCOL. After completion, the free space of DWORK is released.

Parameters Input

ELE	SUB	Name of the element subprogram (in AP3 and AP7)
ELEn	SUB	n=1,2, names of the element subprograms for the test and trial space (in this order) (in AP9)
ISYMM	I*4	=1 symmetry of matrix assumed, only pointers for upper triangular part generated (storage techniques 4 and 8)
KVERT	I*4	DIMENSION KVERT(NNVE,NEL) Vertices of elements
KMID	I*4	DIMENSION KMID(NNVE,NEL) Numbers of midpoints (edges) of elements, if necessary

Output

KDIA	I*4	DIMENSION KDIA(NDIA) Diagonal offset pointer (in AP3)
KDIAS	I*4	DIMENSION KDIA(NDIA+1) Pointer to start of new diagonal row (in AP3)
NDIA	I*4	Number of diagonal rows in matrix (in AP3)
KCOL	I*4	DIMENSION KCOL(NA) Column pointer (in AP7 and AP9)
KLD	I*4	DIMENSION KLD(NEQ+1) Pointer to start of new row (in AP7 and AP9)
NA	I*4	Number of entries in matrix
NEQ	I*4	Number of equations

<u>Parameters in COMMON blocks</u>

```
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /TRIAA/  LCORVG,LCORMG,LVERT,LMID,LADJ,LVEL,LMEL,LNPR,LMM,
     *                LVBD,LEBD,LBCT,LVBDP,LMBDP
```

Information about the subdivision. /TRIAA/ is only needed in XAP7 and XAP9.

List of available subprograms

```
SUBROUTINE XAP3(LDIA,LDIAS,NDIA,NA,NEQ,ELE,ISYMM)
SUBROUTINE  AP3(KDIA,KDIAS,NDIA,NA,NEQ,ELE,ISYMM,KVERT,KMID,KDIAH)

SUBROUTINE XAP7(LCOL,LLD,NA,NEQ,ELE,ISYMM)
SUBROUTINE  AP7(KCOL,KLD,NA,NEQ,ELE,ISYMM,KVERT,KMID)

SUBROUTINE XAP9(LCOL,LLD,NA,NEQ,ELE1,ELE2)
SUBROUTINE  AP9(KCOL,KLD,NA,NEQ,ELE1,ELE2,KVERT,KMID)
```

Subgroups AA and AB – Assembly of matrices

The subprograms AAvs (ABvs) assemble global matrices for meshes consisting of triangles (quadrilaterals). The matrices may consist of several blocks which are assumed to possess the same structure of pointer vectors, i.e. the vectors KCOL and KLD (in storage technique 7) are considered to be the same for all blocks. The number of blocks is NBLOC. The different blocks, each of length NA need not be stored sequentially, i.e. in a matrix DA(NA,NBLOC) or VA(NA,NBLOC). Only one starting address is passed as an argument in DA(1) or VA(1) and, additionally, a vector KOFF(NBLOC) containing the offset of the starting address of block matrix IBLOC relative to DA(1).

This is automatically handled by the provided X-routines which simply require the numbers LA(IBLOC) of each block matrix to be allocated on the workspace vector. If one or several block matrices do not exist, i.e. if LA(IBLOC)=0, the matrices are allocated by the X-routines. If the user prefers to directly invoke an A-routine like AB07 instead of the correponding X-routine XAB07, he should define a block matrix, DA(NA,NBLOC) and set KOFF(IBLOC)=(IBLOC-1)*NA.

The A-routines do not overwrite the elements of the matrices but add the new entries to the old ones. However, the corresponding X-routines have a parameter ICLEAR which may be set to 1 for deletion of the old entries.

Structure of the bilinear form

For each of the NBLOC bilinear forms to be evaluated for all basis functions there is passed the number of additive terms in the array KABN(NBLOC) and an abbreviation for the partial derivatives applied to the test (ϕ_i) and trial functions (ϕ_j) in the array KAB(2,NNAB,NBLOC). Consider, for example, the bilinear form

$$\int_\Omega (\partial_x\phi_i\partial_x\phi_j + \partial_y\phi_i\partial_y\phi_j + \beta_1\phi_i\partial_x\phi_j + \beta_2\phi_i\partial_y\phi_j)\; dx$$

Here, β_1 and β_2 are some coefficients. The correponding value for the array KABN is 4 since four additive terms form the integrand.

For the multiindices denoting the partial derivatives in the array `KAB(2,NNAB,NBLOC)` we choose the abbreviations

`KAB(.,.,.)`	Comment
1	Function value
2	First x-derivative
3	First y-derivative
4	Second x-derivative
5	Mixed xy-derivative
6	Second y-derivative

Therefore, the array `KAB(2,NNAB,NBLOC)` must contain the values

$$\begin{matrix} 2 & 2 \\ 3 & 3 \\ 1 & 2 \\ 1 & 3 \end{matrix}$$

for the four terms in the above example. Notice that the first number denotes the derivative applied to the test function (!).

Coefficients

For the evaluation of the coefficients $c_{\alpha\beta}$, an `EXTERNAL` function `COEFF` is passed as an argument

```
DOUBLE PRECISION FUNCTION COEFF(X,Y,IA,IB,IBLOC,BFIRST).
```

Parameters

`X,Y`	`R*8`	Coordinates of evaluation point
`IA,IB`	`I*4`	Abbreviation for partial derivatives applied to test and trial functions, see above
`IBLOC`	`I*4`	Number of block matrix
`BFIRST`	`L*4`	`.TRUE.` means that the coefficient function is evaluated for the first time at the particular cubature point. This can be used to save arithmetic operations for further calls of `COEFF`, in particular in nonlinear problems

`COEFF` returns the value of the coefficient at the given evaluation point `(X,Y)`. In case of nonlinear problems, `COEFF` may use information about the current element, the values of basis functions and its derivatives at the cubature point, etc. This information is available in the `COMMON` blocks `/ELEM/`, `/CUB/`, `/COAUX1/`, and `/COAUX2/`. Moreover, when called with `IA=IB=-1` the coefficient function has to set the values of `BDER(IDER)` if it needs information about the `IDER`-th derivative of the basis functions.

Parameters Input

LA	I*4	DIMENSION LA(NBLOC) Handles of block matrices (for X-routines only)
LDIA	I*4	DIMENSION KDIA(NDIA) Diagonal offset pointer, generated by AP3
LDIAS	I*4	DIMENSION KDIAS(NDIA+1) Pointer to start of new diagonal rows, generated by AP3
NDIA	I*4	Number of diagonal rows in each block matrix, generated by AP3
LCOL	I*4	DIMENSION KCOL(NA) Column pointer, generated by AP7 and AP9
LLD	I*4	DIMENSION KLD(NEQ+1) Pointer to start of new rows, generated by AP7 and AP9
NA	I*4	Number of nonzero entries in each block matrix
NEQ	I*4	Number of rows in each block matrix
NBLOC	I*4	Number of block matrices
ICLEAR	I*4	=1 Old entries are set to zero =0 New elements are added to old ones
KOFF	I*4	DIMENSION KOFF(NBLOC) Offsets of starting address of matrix block IBLOC relative to starting address of first block
KVERT	I*4	DIMENSION KVERT(NNVE,NEL) Vertices of elements
KMID	I*4	DIMENSION KMID(NNVE,NEL) Numbers of midpoints (edges) of elements, if needed
DCORVG	R*8	DIMENSION DCORVG(2,NVT) Coordinates of vertices
KNPR	I*4	DIMENSION KNPR(NVT+NMT) Nodal properties, see Section 1.5
IBCT	I*4	Boundary component (for routines AA1s and AB1s only)
KBCT	I*4	DIMENSION KBCT(NBCT+1) Additional arrays describing the triangulation, see Section 1.5 (for routines AA1s and AB1s only)
KVBD	I*4	DIMENSION KVBD(NVBD)
KEBD	I*4	DIMENSION KEBD(NVBD)
IVBDn	I*4	n=1,2 Minimum and maximum index on KVBD as calculated by routine SIVB (for routines AA1s and AB1s only)
TMAX	FUN	Maximum parameter of boundary curves (for routines XAA1s and AB1s only)
DPARn	R*8	n=1,2 Description of the boundary part (for routines XAA1s and XAB1s only)
ELE	SUB	Name of the element subprogram (storage techniques 3 and 7)
ELEn	SUB	n=1,2 Names of the element subprograms for the test and trial space (in this order, storage technique 9)
ELE3	SUB	Additional element eventually needed for nonlinear problems (storage technique 9)

COEFF	FUN	Coefficient function, as described above
BCON	L*4	DIMENSION BCON(NBLOC) BCON(IBLOC).EQ..TRUE. means that block IBLOC has constant coefficients
COECON	R*8	DIMENSION COECON(NNDER,NNDER,NBLOC) Auxiliary vector (for the constant coefficient case)
KAB	I*4	DIMENSION KAB(2,NNAB,NBLOC) Abbreviations of partial derivatives applied to basis functions
KABN	I*4	DIMENSION KABN(NBLOC) Numbers of additive terms in each bilinear form
ICUB	I*4	Number of cubature formula, see group C
ISYMM	I*4	= 0 No symmetry assumed, full matrix calculated = 1 Only upper triangular part calculated = 2 Symmetry assumed but full matrix calculated, lower triangular part obtained by reflection
ARR	C*6	DIMENSION ARR(NBLOC) Names of block matrices, for messages only
BSNGL	B*4	=.TRUE. Change matrix type to SINGLE PRECISION

Output

DA	R*8	DIMENSION DA(NA) Resulting block matrix, DOUBLE PRECISION
VA	R*4	DIMENSION VA(NA) Resulting block matrix, REAL

Parameters in COMMON blocks

```
      PARAMETER (NNCUBP=36)

      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /TRIAA/  LCORVG,LCORMG,LVERT,LMID,LADJ,LVEL,LMEL,LNPR,LMM,
     *                LVBD,LEBD,LBCT,LVBDP,LMBDP
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
```

Exchange of information with COEFF

```
      PARAMETER (NNBAS=21,NNDER=6)

      COMMON /COAUX1/ KDFG(NNBAS),KDFL(NNBAS),IDFL
      COMMON /COAUX2/ DBAS1(NNBAS,NNDER,3),KDFG1(NNBAS,3),
     *                KDFL1(NNBAS,3),IDFL1(3),BDER1(NNDER,3)
```

List of available subprograms – Triangular elements

```
      SUBROUTINE XAA03(LA,LDIA,LDIAS,NDIA,NA,NEQ,NBLOC,ICLEAR,
     *                 ELE,COEFF,BCON,KAB,KABN,ICUB,ISYMM,ARR,BSNGL)
      SUBROUTINE  AA03(DA,KDIA,KDIAS,NDIA,NA,NEQ,NBLOC,KOFF,KVERT,KMID,
     *                 DCORVG,KNPR,ELE,COEFF,BCON,COECON,KAB,KABN,ICUB,
     *                 ISYMM)

      SUBROUTINE XAA07(LA,LCOL,LLD,NA,NEQ,NBLOC,ICLEAR,ELE,COEFF,BCON,
     *                 KAB,KABN,ICUB,ISYMM,ARR,BSNGL)
      SUBROUTINE  AA07(DA,KCOL,KLD,NA,NEQ,NBLOC,KOFF,KVERT,KMID,DCORVG,
     *                 KNPR,ELE,COEFF,BCON,COECON,KAB,KABN,ICUB,ISYMM)

      SUBROUTINE XAA09(LA,LCOL,LLD,NA,NBLOC,ICLEAR,ELE1,ELE2,ELE3,COEFF,
     *                 BCON,KAB,KABN,ICUB,ARR,BSNGL)
      SUBROUTINE  AA09(DA,KCOL,KLD,NA,NBLOC,KOFF,KVERT,KMID,
     *                 DCORVG,KNPR,ELE1,ELE2,ELE3,COEFF,
     *                 BCON,COECON,KAB,KABN,ICUB)


      SUBROUTINE XAA13(LA,LDIA,LDIAS,NDIA,NA,NEQ,NBLOC,ICLEAR,TMAX,
     *                 DPAR1,DPAR2,IBCT,ELE,COEFF,BCON,KAB,KABN,
     *                 ICUB,ISYMM,ARR,BSNGL)
      SUBROUTINE  AA13(DA,KDIA,KDIAS,NDIA,NA,NEQ,NBLOC,KOFF,KVERT,KMID,
     *                 DCORVG,KBCT,IBCT,KVBD,KEBD,IVBD1,IVBD2,
     *                 ELE,COEFF,BCON,COECON,KAB,KABN,ICUB,ISYMM)

      SUBROUTINE XAA17(LA,LCOL,LLD,NA,NEQ,NBLOC,ICLEAR,TMAX,
     *                 DPAR1,DPAR2,IBCT,ELE,COEFF,BCON,KAB,KABN,
     *                 ICUB,ISYMM,ARR,BSNGL)
      SUBROUTINE  AA17(DA,KCOL,KLD,NA,NEQ,NBLOC,KOFF,KVERT,KMID,
     *                 DCORVG,KBCT,IBCT,KVBD,KEBD,IVBD1,IVBD2,
     *                 ELE,COEFF,BCON,COECON,KAB,KABN,ICUB,ISYMM)

      SUBROUTINE XAA19(LA,LCOL,LLD,NA,NEQ,NBLOC,ICLEAR,TMAX,
     *                 DPAR1,DPAR2,IBCT,ELE1,ELE2,ELE3,COEFF,BCON,
     *                 KAB,KABN,ICUB,ARR,BSNGL)
      SUBROUTINE  AA19(DA,KCOL,KLD,NA,NEQ,NBLOC,KOFF,KVERT,KMID,
     *                 DCORVG,KBCT,IBCT,KVBD,KEBD,IVBD1,IVBD2,
     *                 ELE1,ELE2,ELE3,COEFF,BCON,COECON,KAB,KABN,ICUB)
```

List of available subprograms – Quadrilateral elements

```
      SUBROUTINE XAB03(LA,LDIA,LDIAS,NDIA,NA,NEQ,NBLOC,ICLEAR,
     *                 ELE,COEFF,BCON,KAB,KABN,ICUB,ISYMM,ARR,BSNGL)
      SUBROUTINE  AB03(DA,KDIA,KDIAS,NDIA,NA,NEQ,NBLOC,KOFF,KVERT,KMID,
     *                 DCORVG,KNPR,ELE,COEFF,BCON,COECON,KAB,KABN,ICUB,
     *                 ISYMM)

      SUBROUTINE XAB07(LA,LCOL,LLD,NA,NEQ,NBLOC,ICLEAR,ELE,COEFF,BCON,
     *                 KAB,KABN,ICUB,ISYMM,ARR,BSNGL)
      SUBROUTINE  AB07(DA,KCOL,KLD,NA,NEQ,NBLOC,KOFF,KVERT,KMID,DCORVG,
     *                 KNPR,ELE,COEFF,BCON,COECON,KAB,KABN,ICUB,ISYMM)

      SUBROUTINE XAB09(LA,LCOL,LLD,NA,NBLOC,ICLEAR,ELE1,ELE2,ELE3,COEFF,
     *                 BCON,KAB,KABN,ICUB,ARR,BSNGL)
      SUBROUTINE  AB09(DA,KCOL,KLD,NA,NBLOC,KOFF,KVERT,KMID,
     *                 DCORVG,KNPR,ELE1,ELE2,ELE3,COEFF,
     *                 BCON,COECON,KAB,KABN,ICUB)


      SUBROUTINE XAB13(LA,LDIA,LDIAS,NDIA,NA,NEQ,NBLOC,ICLEAR,TMAX,
     *                 DPAR1,DPAR2,IBCT,ELE,COEFF,BCON,KAB,KABN,
     *                 ICUB,ISYMM,ARR,BSNGL)
      SUBROUTINE  AB13(DA,KDIA,KDIAS,NDIA,NA,NEQ,NBLOC,KOFF,KVERT,KMID,
     *                 DCORVG,KBCT,IBCT,KVBD,KEBD,IVBD1,IVBD2,
     *                 ELE,COEFF,BCON,COECON,KAB,KABN,ICUB,ISYMM)

      SUBROUTINE XAB17(LA,LCOL,LLD,NA,NEQ,NBLOC,ICLEAR,TMAX,
     *                 DPAR1,DPAR2,IBCT,ELE,COEFF,BCON,KAB,KABN,
     *                 ICUB,ISYMM,ARR,BSNGL)
      SUBROUTINE  AB17(DA,KCOL,KLD,NA,NEQ,NBLOC,KOFF,KVERT,KMID,
     *                 DCORVG,KBCT,IBCT,KVBD,KEBD,IVBD1,IVBD2,
     *                 ELE,COEFF,BCON,COECON,KAB,KABN,ICUB,ISYMM)

      SUBROUTINE XAB19(LA,LCOL,LLD,NA,NEQ,NBLOC,ICLEAR,TMAX,
     *                 DPAR1,DPAR2,IBCT,ELE1,ELE2,ELE3,COEFF,BCON,
     *                 KAB,KABN,ICUB,ARR,BSNGL)
      SUBROUTINE  AB19(DA,KCOL,KLD,NA,NEQ,NBLOC,KOFF,KVERT,KMID,
     *                 DCORVG,KBCT,IBCT,KVBD,KEBD,IVBD1,IVBD2,
     *                 ELE1,ELE2,ELE3,COEFF,BCON,COECON,KAB,KABN,ICUB)
```

Group C – Numerical integration

The subprograms of this group return information on numerical integration rules in one and two space dimensions. The only argument is `ICUB`, the number of the integration rule. The number and position of the cubature or quadrature points as well as the weights are returned in the corresponding arrays in `COMMON /CUB/`.

Parameters in `COMMON` blocks

```
PARAMETER (NNCUBP=36)

COMMON /CUB/      DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
```

List of available subprograms

```
SUBROUTINE CB1 (ICUB)
SUBROUTINE CB2Q(ICUB)
SUBROUTINE CB2T(ICUB)
```

1. Subroutine CB1

Integration formulas for the reference interval $[-1, 1]$
`DXI(ICUBP,1), ICUBP=1,NCUBP` returns the position of the integration points in the interval $[-1, 1]$.

List of available cubature formulas

ICUB	NCUBP	Degree	Comment
1	1	2	Gaussian formula
2	2	2	Trapezoidal rule
3	2	4	Gaussian formula
4	3	4	Simpson rule
5	3	6	Gaussian formula
6	4	8	Gaussian formula

2. Subroutine CB2Q

Integration formulas for the reference square $[-1, 1] \times [-1, 1]$
`DXI(ICUBP,I), I=1,2, ICUBP=1,NCUBP` returns the position of the cubature points in cartesian coordinates.

List of available cubature formulas

`ICUB`	`NCUBP`	Degree	Comment
1	1	2	1×1 Gaussian formula
2	3	2	Trapezoidal rule
3	4	2	Midpoints of edges
4	4	4	2×2 Gaussian formula
5	4	4	Not symmetric
6	6	5	Not symmetric
7	7	6	Not symmetric
8	9	6	3×3 Gaussian formula
9	12	7	Gaussian formula
10	16	8	4x4 Gaussian formula
11	25	10	5×5 Gaussian formula
12	4	2	Piecewise 1×1 Gauss formula
13	9	2	Piecewise trapezoidal rule
14	16	4	Piecewise 2×2 Gauss formula
15	36	6	Piecewise 3×3 Gauss formula

The piecewise defined formulas 12, 13, 14, and 15 are used in connection with the piecewise defined elements, e.g. `4Q1` and `4Q0`.

3. Subroutine `CB2T`

Integration formulas for the reference triangle
`DXI(ICUBP,I), I=1,2,3, ICUBP=1,NCUBP` returns the position of the cubature points in barycentric coordinates.

List of available cubature formulas

`ICUB`	`NCUBP`	Degree	Comment
1	1	2	1x1 Gauss formula
2	3	2	Trapezoidal rule
3	3	3	3-point Gauss formula
4	3	3	Collatz formula
5	6	4	As 4 plus midpoints of edge
6	7	4	vertices, midpoints, center
7			Not yet implemented
8	7	6	
9	12	7	
10			Not yet implemented
11	4	2	Piecewise 1×1 Gauss formula
12	6	2	Piecewise trapezoidal rule
13	9	3	Piecewise 3-point Gauss

The piecewise defined formulas 11, 12, and 13 are used in connection with the piecewise defined elements, e.g. `4P1`.

Group E – Element library

The routines of group E serve for evaluation of function values and/or derivatives of the basis functions on the current element at a given point. The name of the subprograms consists of the letter E followed by the number of the element (3 digits). The names of the subprograms in use must be declared EXTERNAL by the user in the main program.

All information needed for the evaluation except possibly the evaluation point is passed to the element subprograms in the COMMON blocks /ELEM/ and /CUB/. The calculated values are returned to the calling routines in the array DBAS in /ELEM/.

In the present version, FEAT2D supports triangular and quadrilateral elements. For triangular elements the transformation to the reference triangle is assumed to be affine linear, for quadrilaterals the transformation is bilinear. The parameter list depends on the shape of the element,

`SUBROUTINE Ennn(XI1,XI2,XI3,IPAR)`	for triangular elements,
`SUBROUTINE Ennn(XI1,XI2,IPAR)`	for quadrilaterals.

Parameters Input

XIn	R*8	n=1,2(,3) Coordinates of the evaluation point quadrilateral elements – XI1, XI2 are the cartesian coordinates in the reference element $[-1,1]\times[-1,1]$ triangular elements – XI1, XI2, XI3 are the barycentric coordiates
IPAR	I*4	Switch IPAR= 0: Evaluate at the given point IPAR=-1: Return number of element on IPAR IPAR=-2: The routine may save arithmetic operations for further evaluations. For example, the function values of the basis functions on the reference element may be calculated in all cubature points and saved on a local array. Then, in later calls one only performs the transformation to the actual element using the elements of the Jacobian in /ELEM/ and the number ICUBP of the current cubature point in /CUB/. A second application of this mechanism is used in piecewise defined elements. Here, the decision which cubature point is located in which of the subelements is stored. The calling routine must set ICUBP to the number of the cubature formula when calling with IPAR=-2! For IPAR=-2, no information is returned. IPAR=-3: Evaluate at given point assuming that the routine has been called using IPAR=-2, before.

Parameters in `COMMON` blocks

```
      PARAMETER (NNBAS=21,NNDER=6,NNVE=4,NNCUBP=36)

      COMMON /ELEM/ DX(NNVE),DY(NNVE),DJAC(2,2),DETJ,
     *              DBAS(NNBAS,NNDER),BDER(NNDER),KVE(NNVE),IEL
      COMMON /CUB/  DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
```

List of available elements

Name	IELTYP	# d.o.f.	Comment
E000	0	1	constant, triangle
E001	1	3	linear, continuous, triangle
E002	2	6	quadratic, continuous, triangle
E003	3	10	cubic, Hermite, continuous, triangle
E004	4	10	cubic, Lagrange, continuous, triangle
E010	10	1	constant, quadrilateral
E011	11	4	bilinear, continuous, quadrilateral
E012	12	8	reduced biquadratic (Serendipity), quadrilateral
E013	13	9	biquadratic, continuous, quadrilateral
E014	14	16	bicubic, continuous, quadrilateral
E020	20	3	linear nonconforming, discontinuous triangle nodal values in midpoints of edges
E021	21	4	augmented linear (MINI, P1+bulb), triangle
E022	22	6	piecewise linear (4P1), triangle triangle divided into 4 subtriangles
E023	23	7	augmented quadratic (P2+bulb), triangle
E030	30	4	rotated bilinear, discontinuous, quadrilateral mean values on edges as nodal values
E031	31	4	rotated bilinear, discontinuous, quadrilateral function values at midpoints of edges as nodal values
E032	32	5	five node (Han), discontinuous, quadrilateral Q1 plus two special cubic functions
E033	33	9	piecewise bilinear (4Q1), quadrilateral quadrilateral divided into 4 subelements
E034	34	4	piecewise constant (4Q0), quadrilateral quadrilateral divided into 4 subelements
E040	40	1	cubic bulb, single bulb function only, triangle
E050	50	6	quadratic Morley, triangle
E060	60	3	linear, discontinuous, triangle no continuity constraints
E061	61	3	linear, discontinuous, quadrilateral no continuity constraints

Group G – Normalized output for graphic packages

This group contains subroutines writing grid information or user supplied data in the special format that is needed by certain graphic packages. Up to now only the Movie.byu graphic package is supported. The subroutine XGOWSM writes FEAT2D grid data in Movie.byu format onto file CFILE and the subroutine XGOWFM writes a Movie.byu function file.

```
SUBROUTINE XGOWFM(LNR,MFILE,CFILE)
SUBROUTINE XGOWSM(MFILE,CFILE)
```

Parameters Input

LNR	I*4	Function array handle (XGOWFM only)
MFILE	I*4	Unit number
CFILE	C**	File name

Output

CFILE	C**	File containing grid or function data in Movie.byu format

Group I **– Iterative methods for linear equations**

The subprograms of group `I` deal with iterative methods for linear systems of equations. Several versions for each algorithm are provided differing in the data type of the matrix and the vector and in the assumed storage technique for the matrix. Further, we distinguish whether an algorithm is used for

0 Solution of a linear system up to a certain accuracy

1 Preconditioning (e.g., scaling, SSOR-preconditioning)

2 Smoothing (i.e. performing a fixed number of iteration steps independent of the accuracy as used in multigrid algorithms)

3 Approximate solution of a linear system (i.e. reducing the starting residual by a certain number of digits)

The numbers 0, 1, 2 and 3 again occur in the names of the subprograms, below.

Names of the subprograms

The names of the programs in this subgroup have the form `Iatns`. Here, `a` denotes the iterative algorithm, namely

`A` Jacobi method,

`B` Gauss-Seidel method,

`C` Successive overrelaxation (SOR),

`D` Symmetric successive overrelaxation (SSOR),

`E` (Preconditioned) Conjugate gradient algorithm (PCG),

`F` Incomplete Cholesky decomposition (ILU),

`G` (Preconditioned) Squared conjugate gradient algorithm (CGS),

`M` Multigrid algorithm.

`t` stands for the numbers `0`, `1`, `2` or `3` characterizing the specific task (solution, preconditioning, smoothing), as described above.
The number `n` refers to the data type of the arguments, namely

`1` `DOUBLE PRECISION` matrix, `DOUBLE PRECISION` vectors,

`2` `REAL` matrix, `REAL` vectors,

`3` `REAL` matrix, `DOUBLE PRECISION` vectors.

Finally, the character `s` stands for the storage technique (`0,...,A`), see Section 1.6.

If an algorithm is suited for the solution of a linear system the corresponding `X`-routines are provided. The name of the program is preceded by the letter `X` and only the numbers (handles) of the vectors are given as an argument. For algorithms that are only used as smoothers or preconditioners only the corresponding `Y`-routines exist in the present version of FEAT2D, see below.

Parameters Input

DA,VA	R*8	DIMENSION DA(NA),VA(NA) Double/single precision matrix
KDIA	I*4	DIMENSION KDIA(NDIA) Pointer vectors (see Section 1.6)
KDIAS	I*4	DIMENSION KDIA(NDIA+1)
NDIA	I*4	
KCOL	I*4	DIMENSION KCOL(NA)
KLD	I*4	DIMENSION KLD(NEQ+1)
KOP	I*4	DIMENSION KOP(NEQ)
NA	I*4	
DX,VX	R*8	DIMENSION DX(NEQ),VX(NEQ) Starting vector, double/single precision
DB,VB	R*8	DIMENSION DB(NEQ),VB(NEQ) Right hand side vector, double/single precision
NEQ	I*4	Number of equations
ITE	I*4	Minimum number of iterations, used for IREL=1
NIT	I*4	(Maximum) number of iterations
EPS	R*8	Desired accuracy for residual or iterative correction IREL=0: Stop if !!RES!! < EPS IREL=1: Stop if !!RES!!/!!RES0!! < EPS and a minimum of ITE iterations is performed
OMEGA	R*8	Relaxation or damping parameter
DD,VD	R*8	DIMENSION DD(NEQ),VD(NEQ)
DR,VR	R*8	Auxiliary vectors, double/single precision
DD1,VD1	R*8	
DG,VG	R*8	
IREL	I*4	relative or absolute accuracy desired

Output

DX,VX	R*8	Resulting vector, double/single precision
ITE	I*4	Number of iterations needed

```
      SUBROUTINE I000
```

This routine performs no operation and is used as `EXTERNAL` dummy routine only.

Subgroup IA **– Jacobi method**

Solution of linear system by Jacobi's method
A maximum of NIT iterations is performed. The procedure stops if the maximum of the correction between two steps is less than EPS. DD serves as auxiliary vector.

```
SUBROUTINE IA013(DA,KDIA,KDIAS,NDIA,DX,DB,DD,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE IA017(DA,KCOL,KLD,DX,DB,DD,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE IA01A(DA,KCOL,KLD,KOP,DX,DB,DD,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE IA023(VA,KDIA,KDIAS,NDIA,VX,VB,VD,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE IA027(VA,KCOL,KLD,VX,VB,VD,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE IA02A(VA,KCOL,KLD,KOP,VX,VB,VD,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE IA033(VA,KDIA,KDIAS,NDIA,DX,DB,DD,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE IA037(VA,KCOL,KLD,DX,DB,DD,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE IA03A(VA,KCOL,KLD,KOP,DX,DB,DD,NEQ,NIT,ITE,EPS,OMEGA)
```

Preconditioning by Jacobi's method, i.e. scaling by the diagonal of the matrix

```
SUBROUTINE IA113(DA,DX,NEQ)
SUBROUTINE IA117(DA,KLD,DX,NEQ)
SUBROUTINE IA11A(DA,KLD,KOP,DX,NEQ)
SUBROUTINE IA123(VA,VX,NEQ)
SUBROUTINE IA127(VA,KLD,VX,NEQ)
SUBROUTINE IA12A(VA,KLD,KOP,VX,NEQ)
SUBROUTINE IA133(VA,DX,NEQ)
SUBROUTINE IA137(VA,KLD,DX,NEQ)
SUBROUTINE IA13A(VA,KLD,KOP,DX,NEQ)
```

Smoothing by the damped Jacobi method
NIT damped Jacobi iterations with damping parameter OMEGA are performed, DD serves as auxiliary vector.

```
SUBROUTINE IA213(DA,KDIA,KDIAS,NDIA,DX,DB,DD,NEQ,NIT,OMEGA)
SUBROUTINE IA217(DA,KCOL,KLD,DX,DB,DD,NEQ,NIT,OMEGA)
SUBROUTINE IA21A(DA,KCOL,KLD,KOP,DX,DB,DD,NEQ,NIT,OMEGA)
SUBROUTINE IA223(VA,KDIA,KDIAS,NDIA,VX,VB,VD,NEQ,NIT,OMEGA)
SUBROUTINE IA227(VA,KCOL,KLD,VX,VB,VD,NEQ,NIT,OMEGA)
SUBROUTINE IA22A(VA,KCOL,KLD,KOP,VX,VB,VD,NEQ,NIT,OMEGA)
SUBROUTINE IA233(VA,KDIA,KDIAS,NDIA,DX,DB,DD,NEQ,NIT,OMEGA)
SUBROUTINE IA237(VA,KCOL,KLD,DX,DB,DD,NEQ,NIT,OMEGA)
SUBROUTINE IA23A(VA,KCOL,KLD,KOP,DX,DB,DD,NEQ,NIT,OMEGA)
```

Subgroup IB – Gauss-Seidel method

Solution of linear system by Gauss-Seidel method
A maximum of NIT iterations is performed. The procedure stops if the maximum of the correction between two steps is less than EPS.

```
SUBROUTINE IB017(DA,KCOL,KLD,DX,DB,NEQ,NIT,ITE,EPS)
SUBROUTINE IB01A(DA,KCOL,KLD,KOP,DX,DB,NEQ,NIT,ITE,EPS)
SUBROUTINE IB027(VA,KCOL,KLD,VX,VB,NEQ,NIT,ITE,EPS)
SUBROUTINE IB02A(VA,KCOL,KLD,KOP,VX,VB,NEQ,NIT,ITE,EPS)
SUBROUTINE IB037(VA,KCOL,KLD,DX,DB,NEQ,NIT,ITE,EPS)
SUBROUTINE IB03A(VA,KCOL,KLD,KOP,DX,DB,NEQ,NIT,ITE,EPS)
```

Smoothing by Gauss-Seidel method
NIT iterations are performed.

```
SUBROUTINE IB217(DA,KCOL,KLD,DX,DB,NEQ,NIT)
SUBROUTINE IB21A(DA,KCOL,KLD,KOP,DX,DB,NEQ,NIT)
SUBROUTINE IB227(VA,KCOL,KLD,VX,VB,NEQ,NIT)
SUBROUTINE IB22A(VA,KCOL,KLD,KOP,VX,VB,NEQ,NIT)
SUBROUTINE IB237(VA,KCOL,KLD,DX,DB,NEQ,NIT)
SUBROUTINE IB23A(VA,KCOL,KLD,KOP,DX,DB,NEQ,NIT)
```

Subgroup IC – SOR method

Solution of linear system by SOR-method
A maximum of NIT SOR iterations with relaxation parameter OMEGA is performed. The procedure stops if the maximum of the correction between two steps is less than EPS. The X-routines check the data type of array and vectors before calling the corresponding I-routines.

```
SUBROUTINE XIC017(LA,LCOL,LLD,LX,LB,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE  IC017(DA,KCOL,KLD,DX,DB,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE XIC01A(LA,LCOL,LLD,LOP,LX,LB,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE  IC01A(DA,KCOL,KLD,KOP,DX,DB,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE XIC027(LA,LCOL,LLD,LX,LB,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE  IC027(VA,KCOL,KLD,VX,VB,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE XIC02A(LA,LCOL,LLD,LOP,LX,LB,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE  IC02A(VA,KCOL,KLD,KOP,VX,VB,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE XIC037(LA,LCOL,LLD,LX,LB,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE  IC037(VA,KCOL,KLD,DX,DB,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE XIC03A(LA,LCOL,LLD,LOP,LX,LB,NEQ,NIT,ITE,EPS,OMEGA)
SUBROUTINE  IC03A(VA,KCOL,KLD,KOP,DX,DB,NEQ,NIT,ITE,EPS,OMEGA)
```

Smoothing by SOR method
`NIT` iterations are performed.

```
SUBROUTINE IC217(DA,KCOL,KLD,DX,DB,NEQ,NIT,OMEGA)
SUBROUTINE IC21A(DA,KCOL,KLD,KOP,DX,DB,NEQ,NIT,OMEGA)
SUBROUTINE IC227(VA,KCOL,KLD,VX,VB,NEQ,NIT,OMEGA)
SUBROUTINE IC22A(VA,KCOL,KLD,KOP,VX,VB,NEQ,NIT,OMEGA)
SUBROUTINE IC237(VA,KCOL,KLD,DX,DB,NEQ,NIT,OMEGA)
SUBROUTINE IC23A(VA,KCOL,KLD,KOP,DX,DB,NEQ,NIT,OMEGA)
```

Subgroup ID – SSOR method

Preconditioning by the S(ymmetric)SOR method with relaxation parameter `OMEGA`

```
SUBROUTINE ID117(DA,KCOL,KLD,DX,NEQ,OMEGA)
SUBROUTINE ID118(DA,KCOL,KLD,DX,NEQ,OMEGA)
SUBROUTINE ID11A(DA,KCOL,KLD,KOP,DX,NEQ,OMEGA)
SUBROUTINE ID127(VA,KCOL,KLD,VX,NEQ,OMEGA)
SUBROUTINE ID128(VA,KCOL,KLD,VX,NEQ,OMEGA)
SUBROUTINE ID12A(VA,KCOL,KLD,KOP,VX,NEQ,OMEGA)
SUBROUTINE ID137(VA,KCOL,KLD,DX,NEQ,OMEGA)
SUBROUTINE ID138(VA,KCOL,KLD,DX,NEQ,OMEGA)
SUBROUTINE ID13A(VA,KCOL,KLD,KOP,DX,NEQ,OMEGA)
```

Smoothing by SSOR method
`NIT` iterations are performed.

```
SUBROUTINE ID217(DA,KCOL,KLD,DX,DB,NEQ,NIT,OMEGA)
SUBROUTINE ID218(DA,KCOL,KLD,DX,DB,NEQ,NIT,OMEGA)
SUBROUTINE ID21A(DA,KCOL,KLD,KOP,DX,DB,NEQ,NIT,OMEGA)
SUBROUTINE ID227(VA,KCOL,KLD,VX,VB,NEQ,NIT,OMEGA)
SUBROUTINE ID228(VA,KCOL,KLD,VX,VB,NEQ,NIT,OMEGA)
SUBROUTINE ID22A(VA,KCOL,KLD,KOP,VX,VB,NEQ,NIT,OMEGA)
SUBROUTINE ID237(VA,KCOL,KLD,DX,DB,NEQ,NIT,OMEGA)
SUBROUTINE ID238(VA,KCOL,KLD,DX,DB,NEQ,NIT,OMEGA)
SUBROUTINE ID23A(VA,KCOL,KLD,KOP,DX,DB,NEQ,NIT,OMEGA)
```

Subgroup IE – Conjugate gradient method

Solution of linear system by the Conjugate gradient method
A maximum of NIT preconditioned CG-iterations are performed. The procedure stops if the residual is less than EPS(routines XIE0ns, IE0ns and IE0n0) or if the relative residual is less than EPS (routines XIE3ns, IE3ns and IE3n0).

The CG-algorithm is implemented using an EXTERNAL subroutine for the matrix-vector product (DAX0, VAX0) and a second one for the preconditioning step (DCG0C, CG0C). No information about the matrix is passed directly to the I-routines. This is indicated by the character 0 for the storage technique (IE010, IE020, IE310, IE320). The X-routines check the data type of array and vectors before calling the corresponding I-routines. They allocate the auxiliary vectors DR, DD, DD1, DG or VR, VD, VD1, VG on the workspace vector. For the matrix-vector multiplication they pass the routines YLAXns as arguments, where n denotes the data type and s stands for the storage technique. They also prepare the COMMON block /MAT1/ for this purpose (see Sections 3.1 and 3.2). The preconditioning is controlled by the parameter OMEGA, namely

OMEGA<0	No preconditioning It is assumed that the vectors DR (VR) and DG (VG) coincide.
OMEGA=0	Scaling by the diagonal of the matrix The X-routines choose YIA1ns for preconditioning, see below.
0<OMEGA<2	SSOR preconditioning with relaxation parameter OMEGA The X-routines choose YID1ns for preconditioning, see below.
OMEGA>2	User defined preconditioning The X-routines choose their arguments DCG0C or CG0C for preconditioning.

In the case OMEGA<0 one may choose the dummy routine I000 as EXTERNAL argument when calling XIE0ns or XIE3ns. In the following lists, the letters v and n stand for v=0,3 and n=1,3. Triple dots, ..., indicate the set of auxiliary vectors DR,DD,DD1,DG, and VR,VD,VD1,VG, respectively.

```
SUBROUTINE XIEvn3(LA,LDIA,LDIAS,NDIA,LX,LB,NEQ,NIT,ITE,EPS,OMEGA,DCG0C)
SUBROUTINE XIEvn4(LA,LDIA,LDIAS,NDIA,LX,LB,NEQ,NIT,ITE,EPS,OMEGA,DCG0C)
SUBROUTINE XIEvn7(LA,LCOL,LLD,LX,LB,NEQ,NIT,ITE,EPS,OMEGA,DCG0C)
SUBROUTINE XIEvn8(LA,LCOL,LLD,LX,LB,NEQ,NIT,ITE,EPS,OMEGA,DCG0C)
SUBROUTINE XIEvnA(LA,LCOL,LLD,LOP,LX,LB,NEQ,NIT,ITE,EPS,OMEGA,DCG0C)
SUBROUTINE  IE010(DX,DB,NEQ,NIT,ITE,EPS,DAX0,DCG0C,BNOCON,
                  DR,DD,DD1,DG,IREL)

SUBROUTINE XIEv23(LA,LDIA,LDIAS,NDIA,LX,LB,NEQ,NIT,ITE,EPS,OMEGA,DCG0C)
SUBROUTINE XIEv24(LA,LDIA,LDIAS,NDIA,LX,LB,NEQ,NIT,ITE,EPS,OMEGA,DCG0C)
SUBROUTINE XIEv27(LA,LCOL,LLD,LX,LB,NEQ,NIT,ITE,EPS,OMEGA,CG0C)
SUBROUTINE XIEv28(LA,LCOL,LLD,LX,LB,NEQ,NIT,ITE,EPS,OMEGA,CG0C)
SUBROUTINE XIEv2A(LA,LCOL,LLD,LOP,LX,LB,NEQ,NIT,ITE,EPS,OMEGA,CG0C)
```

```
SUBROUTINE  IE020(VX,VB,NEQ,NIT,ITE,EPS,VAX0,CGOC,BNOCON,
                  VR,VD,VD1,VG,IREL)

SUBROUTINE  IEv13(DA,KDIA,KDIAS,NDIA,DX,DB,NEQ,NIT,ITE,EPS,OMEGA,...)
SUBROUTINE  IEv14(DA,KDIA,KDIAS,NDIA,DX,DB,NEQ,NIT,ITE,EPS,OMEGA,...)
SUBROUTINE  IEv17(DA,KCOL,KLD,DX,DB,NEQ,NIT,ITE,EPS,OMEGA,...)
SUBROUTINE  IEv18(DA,KCOL,KLD,DX,DB,NEQ,NIT,ITE,EPS,OMEGA,...)
SUBROUTINE  IEv1A(DA,KCOL,KLD,KOP,DX,DB,NEQ,NIT,ITE,EPS,OMEGA,...)

SUBROUTINE  IEv23(VA,KDIA,KDIAS,NDIA,VX,VB,NEQ,NIT,ITE,EPS,OMEGA,...)
SUBROUTINE  IEv24(VA,KDIA,KDIAS,NDIA,VX,VB,NEQ,NIT,ITE,EPS,OMEGA,...)
SUBROUTINE  IEv27(VA,KCOL,KLD,VX,VB,NEQ,NIT,ITE,EPS,OMEGA,...)
SUBROUTINE  IEv28(VA,KCOL,KLD,VX,VB,NEQ,NIT,ITE,EPS,OMEGA,...)
SUBROUTINE  IEv2A(VA,KCOL,KLD,KOP,VX,VB,NEQ,NIT,ITE,EPS,OMEGA,...)

SUBROUTINE  IEv33(VA,KDIA,KDIAS,NDIA,DX,DB,NEQ,NIT,ITE,EPS,OMEGA,...)
SUBROUTINE  IEv34(VA,KDIA,KDIAS,NDIA,DX,DB,NEQ,NIT,ITE,EPS,OMEGA,...)
SUBROUTINE  IEv37(VA,KCOL,KLD,DX,DB,NEQ,NIT,ITE,EPS,OMEGA,...)
SUBROUTINE  IEv38(VA,KCOL,KLD,DX,DB,NEQ,NIT,ITE,EPS,OMEGA,...)
SUBROUTINE  IEv3A(VA,KCOL,KLD,KOP,DX,DB,NEQ,NIT,ITE,EPS,OMEGA,...)
```

Smoothing by (preconditioned) conjugate gradient method, `NIT` iterations are performed

```
SUBROUTINE IE213(DA,KDIA,KDIAS,NDIA,DX,DB,NEQ,NIT,OMEGA,DR,DD,DD1,DG)
SUBROUTINE IE214(DA,KDIA,KDIAS,NDIA,DX,DB,NEQ,NIT,OMEGA,DR,DD,DD1,DG)
SUBROUTINE IE217(DA,KCOL,KLD,DX,DB,NEQ,NIT,OMEGA,DR,DD,DD1,DG)
SUBROUTINE IE218(DA,KCOL,KLD,DX,DB,NEQ,NIT,OMEGA,DR,DD,DD1,DG)
SUBROUTINE IE21A(DA,KCOL,KLD,KOP,DX,DB,NEQ,NIT,OMEGA,DR,DD,DD1,DG)

SUBROUTINE IE223(VA,KDIA,KDIAS,NDIA,DX,DB,NEQ,NIT,OMEGA,VR,VD,VD1,VG)
SUBROUTINE IE224(VA,KDIA,KDIAS,NDIA,DX,DB,NEQ,NIT,OMEGA,VR,VD,VD1,VG)
SUBROUTINE IE227(VA,KCOL,KLD,VX,VB,NEQ,NIT,OMEGA,VR,VD,VD1,VG)
SUBROUTINE IE228(VA,KCOL,KLD,VX,VB,NEQ,NIT,OMEGA,VR,VD,VD1,VG)
SUBROUTINE IE22A(VA,KCOL,KLD,KOP,VX,VB,NEQ,NIT,OMEGA,VR,VD,VD1,VG)

SUBROUTINE IE233(VA,KDIA,KDIAS,NDIA,DX,DB,NEQ,NIT,OMEGA,DR,DD,DD1,DG)
SUBROUTINE IE234(VA,KDIA,KDIAS,NDIA,DX,DB,NEQ,NIT,OMEGA,DR,DD,DD1,DG)
SUBROUTINE IE237(VA,KCOL,KLD,DX,DB,NEQ,NIT,OMEGA,DR,DD,DD1,DG)
SUBROUTINE IE238(VA,KCOL,KLD,DX,DB,NEQ,NIT,OMEGA,DR,DD,DD1,DG)
SUBROUTINE IE23A(VA,KCOL,KLD,KOP,DX,DB,NEQ,NIT,OMEGA,DR,DD,DD1,DG)
```

Subgroup `IF` – ILU Decomposition

Calculation of (shifted, modified) ILU-decomposition, resulting triangular factors stored on input matrix. `ILU` controls whether standard or modified ILU-decompositions are calculated. `ALPHA` serves as a shift parameter. If pivot elements become smaller that `TOL` they are increased to modulus `TOL`.

```
SUBROUTINE IFD17(DA,KCOL,KLD,NEQ,ILU,ALPHA,TOL)
SUBROUTINE IFD27(VA,KCOL,KLD,NEQ,ILU,ALPHA,TOL)
```

Preconditioning by ILU
Input matrix contains incomplete factorization

```
SUBROUTINE IF117(DA,KCOL,KLD,DX,NEQ)
SUBROUTINE IF127(VA,KCOL,KLD,VX,NEQ)
SUBROUTINE IF137(VA,KCOL,KLD,DX,NEQ)
```

Y-routines

The subroutines for preconditioning and smoothing are frequently used in subprograms which have no information about the matrix and its pointer vectors. Therefore, the corresponding I-routines cannot be called directly but an intermediate program preparing the correct parameter list must be used. They are denoted by the letter `Y` followed by the name of the correponding subprogram. The information about the matrix and its pointer vectors is passed to these Y-routines in a `COMMON` block

```
PARAMETER (NNARR=299)

COMMON /XYPAR/  DXYPAR(NNARR),KXYPAR(NNARR)
```

It is assumed that `KXYPAR` contains the numbers of the array and of the pointer vectors which are allocated on the workspace vector, namely

KXYPAR(1)=L(LA) KXYPAR(2)=L(LDIA) KXYPAR(3)=L(LDIAS) KXYPAR(4)=NDIA	or	KXYPAR(1)=L(LA) KXYPAR(2)=L(LCOL) KXYPAR(3)=L(LLD)	or	KXYPAR(1)=L(LA) KXYPAR(2)=L(LCOL) KXYPAR(3)=L(LLD) KXYPAR(4)=L(LOP) .

In multigrid routines this information is stored for all levels at the positions starting from 100, see `XM017`. Moreover, double precision information is stored on `DXYPAR`. For example `DXYPAR(1)=OMEGA` is used for iterative solvers to pass the value of `OMEGA` to the preconditioner. The positions starting from 200 are left for free use.

Further, the BLANK `COMMON` is known to the `Y`-routines. This information then is used to create the full parameter in the call of the correponding `L`-routines as described above.

Names and parameter lists of the `Y`-routines available in the present version of FEAT2D

```
SUBROUTINE  YIA1n3(DX,NEQ)
SUBROUTINE  YIA1n7(DX,NEQ)
SUBROUTINE  YIA1nA(DX,NEQ)
SUBROUTINE  YIA123(VX,NEQ)
```

```
SUBROUTINE  YIA127(VX,NEQ)
SUBROUTINE  YIA12A(VX,NEQ)

SUBROUTINE  YID1n3(DX,NEQ)
SUBROUTINE  YID1n7(DX,NEQ)
SUBROUTINE  YID1nA(DX,NEQ)
SUBROUTINE  YID123(VX,NEQ)
SUBROUTINE  YID127(VX,NEQ)
SUBROUTINE  YID12A(VX,NEQ)
```

Group L – Elementary linear algebra

This group is devoted to the basic linear algebra tasks. We distinguish between two subgroups, namely vector operations and matrix-vector operations. The routines of the first subgroup internally use BLAS routines (see Section 1.8) or, at least, loop unrolling, adapted to the particular machine. The second subgroup contains routines for forming matrix-vector products and, similarly, products using the transposed matrix. Routines corresponding to storage techniques 3 and 4 make use of the BLAS routines `DAXPY/SAXPY` which allow for vectorization. The `X`- or `Y`-routines are discussed together with the corresponding L-programs.

1. Vector operations

Names of the subprograms

The names of the programs in this subgroup have the form `Lttn`. Here, the two characters `tt` characterize the task. For example, `LC` stands for linear combination. `n` stands for the data type of the arguments, namely

1 `DOUBLE PRECISION`,

2 `REAL`,

3 `INTEGER`.

The name of the program is preceded by the letter `X` if only the numbers (handles) of the vectors are given as an argument.

Parameters Input

`DX,DY`	`R*8`	`DIMENSION DX(NX),DY(NX)` Double precision input vectors
`VX,VY`	`R*4`	`DIMENSION VX(NX),VY(NX)` Single precision input vectors
`KX,KY`	`I*4`	`DIMENSION KX(NX),KY(NX)` Integer input vectors (routines `LCLn`)
`NX`	`I*4`	Length of the vectors
`A,An`	`R*8`	Scalars

Output

`XNORM`	`R*8`	As described below
`IND`	`I*4`	

Clearing a vector (fill with zeroes)

```
SUBROUTINE XLCL1(LX,NX)
SUBROUTINE  LCL1(DX,NX)
SUBROUTINE XLCL2(LX,NX)
SUBROUTINE  LCL2(VX,NX)
SUBROUTINE XLCL3(LX,NX)
SUBROUTINE  LCL3(KX,NX)
```

Explanation

DX:=0, VX:=0, or KX:=0.

Copy of a vector

```
SUBROUTINE XLCP1(LX,LY,NX)
SUBROUTINE  LCP1(DX,DY,NX)
SUBROUTINE XLCP2(LX,LY,NX)
SUBROUTINE  LCP2(VX,VY,NX)
SUBROUTINE XLCP3(LX,LY,NX)
SUBROUTINE  LCP3(KX,KY,NX)
```

Explanation

DY:=DX, VY:=VX, or KY:=KX. Implemented using the BLAS routines DCOPY and SCOPY for the DOUBLE PRECISION and REAL version.

l^2-norm of a vector

```
SUBROUTINE XLL21(LX,NX,XNORM)
SUBROUTINE  LL21(DX,NX,XNORM)
SUBROUTINE XLL22(LX,NX,XNORM)
SUBROUTINE  LL22(VX,NX,XNORM)
```

Explanation

XNORM :=||DX||, or XNORM:=||VX||, where ||.|| denotes the l^2-norm. Implemented using the routines LSP1 and LSP2, respectively.

Linear combination of two vectors

```
SUBROUTINE XLLC1(LX,LY,NX,A1,A2)
SUBROUTINE  LLC1(DX,DY,NX,A1,A2)
SUBROUTINE XLLC2(LX,LY,NX,A1,A2)
SUBROUTINE  LLC2(VX,VY,NX,A1,A2)
```

Explanation

DY:=A1*DX+A2*DY, or VY:=A1*VX+A2*VY. Implemented using the BLAS routines DAXPY and DSCAL for the DOUBLE PRECISION version and SAXPY and SSCAL for the REAL version.

Maximum-norm of a vector

```
      SUBROUTINE XLLI1(LX,NX,XNORM,IND)
      SUBROUTINE  LLI1(DX,NX,XNORM,IND)
      SUBROUTINE XLLI2(LX,NX,XNORM,IND)
      SUBROUTINE  LLI2(VX,NX,XNORM,IND)
```

Explanation

`XNORM:=`‖`DX`‖, or `XNORM:=`‖`VX`‖, where ‖.‖ denotes the maximum-norm. `IND` returns the (first) index where the (absolute) maximum occurs. Implemented using the BLAS routines `IDAMAX` and `ISAMAX`, respectively.

Scaling of a vector

```
      SUBROUTINE XLSC1(LX,NX,A)
      SUBROUTINE  LSC1(DX,NX,A)
      SUBROUTINE XLSC2(LX,NX,A)
      SUBROUTINE  LSC2(VX,NX,A)
```

Explanation

`DX:=A*DX`, or `VX:=A*VX`. Implemented using the BLAS routines `DSCAL` and `SSCAL`, respectively.

Scalar product of two vectors

```
      SUBROUTINE XLSP1(LX,LY,NX,SP)
      SUBROUTINE  LSP1(DX,DY,NX,SP)
      SUBROUTINE XLSP2(LX,LY,NX,SP)
      SUBROUTINE  LSP2(VX,VY,NX,SP)
```

Explanation

`SP:=(DX,DY)`, or `SP:=(VX,VY)`, where (.,.) denotes the scalar product. Implemented using the BLAS routines `DDOT` and `SDOT`, respectively.

Vector multiply and add

```
      SUBROUTINE XLVM1(LX1,LX2,LX,NX,A1,A2)
      SUBROUTINE  LVM1(DX1,DX2,DX,NX,A1,A2)
      SUBROUTINE XLVM2(LX1,LX2,LX,NX,A1,A2)
      SUBROUTINE  LVM2(VX1,VX2,VX,NX,A1,A2)
      SUBROUTINE XLVM3(LX1,LX2,LX,NX,A1,A2)
      SUBROUTINE  LVM3(DX1,VX2,DX,NX,A1,A2)
```

Explanation

`DX(IX):=A2*DX(IX)+A1*DX1(IX)*DX2(IX)`, `VX(IX):=A2*VX(IX)+A1*VX1(IX)*VX2(IX)`, or `DX(IX):=A2*DX(IX)+A1*DX1(IX)*VX2(IX)`, respectively. Implemented using the BLAS routines `DAXPY` and `SAXPY`, respectively.

2. Matrix-vector operations

The subroutines of this subgroup form matrix-vector products of a given matrix or its transpose and a given vector. A linear combination of the result of this operation and the preceding contents of the output vector is returned to the calling routine. Versions for `DOUBLE` and `SINGLE` precision are provided and also the mixed type, `DOUBLE PRECISION` vectors, `REAL` matrix, is supported.

Names of the subprograms

The names of the programs in this subgroup have the form `LcXns` or `LWSns`. Here, the character `c` stands for `A` in subprograms forming the matrix-vector product `A*X`, and for `T` if the transpose of the matrix `A` is used.
`n` characterizes the data type of the arguments, namely

1 `DOUBLE PRECISION` matrix, `DOUBLE PRECISION` vectors,

2 `REAL` matrix, `REAL` vectors,

3 `REAL` matrix, `DOUBLE PRECISION` vectors.

The final character `s` is used to reference the storage technique for the matrix, see Section 1.6. For all subprograms in this subgroup there exist corresponding `Y`-routines, described below.

Parameters Input

DA	R*8	DIMENSION DA(NA) Double precision matrix
VA	R*4	DIMENSION VA(NA) Single precision matrix
KCOL	I*4	DIMENSION KCOL(NA) Pointer vectors (see Section 1.6)
KLD	I*4	DIMENSION KLD(NEQ+1)
KOP	I*4	DIMENSION KOP(NEQ)
DX	R*8	DIMENSION DX(NX) Double precision input vector
VX	R*4	DIMENSION VX(NX) Single precision input vector
NEQ	I*4	Number of equations = number of rows in the matrix A
NEQn	I*4	n=1,2 Length of `DX` (`VX`) and `DTX` (`VTX`) resp. (for routines `LcXn9` only)
An	R*8	n=1,2 Scalars for linear combination, see below

Output

DAX	R*8	`DAX:=A1*DA*DX+A2*DAX` (Double precision version) `DAX:=A1*VA*DX+A2*DAX` (Mixed precision version)

VAX	R*4	VAX:=A1*VA*VX+A2*VAX (Single precision version)
DTX	R*8	DTX:=A1*DA(T)*DX+A2*DTX (Double precision version)
		DTX:=A1*VA(T)*DX+A2*DTX (Mixed precision version)
VTX	R*4	VTX:=A1*VA(T)*VX+A2*VTX (Single precision version)

Here, DA(T) and VA(T) denote the transpose of the matrix.

Subroutines forming the matrix vector product $A \cdot x$

```
      SUBROUTINE LAX13(DA,KDIA,KDIAS,NDIA,NEQ,DX,DAX,A1,A2)
      SUBROUTINE LAX14(DA,KDIA,KDIAS,NDIA,NEQ,DX,DAX,A1,A2)
      SUBROUTINE LAX17(DA,KCOL,KLD,NEQ,DX,DAX,A1,A2)
      SUBROUTINE LAX18(DA,KCOL,KLD,NEQ,DX,DAX,A1,A2)
      SUBROUTINE LAX19(DA,KCOL,KLD,NEQ,DX,DAX,A1,A2)
      SUBROUTINE LAX1A(DA,KCOL,KLD,KOP,NEQ,DX,DAX,A1,A2)

      SUBROUTINE LAX23(VA,KDIA,KDIAS,NDIA,NEQ,VX,VAX,A1,A2)
      SUBROUTINE LAX24(VA,KDIA,KDIAS,NDIA,NEQ,VX,VAX,A1,A2)
      SUBROUTINE LAX27(VA,KCOL,KLD,NEQ,VX,VAX,A1,A2)
      SUBROUTINE LAX28(VA,KCOL,KLD,NEQ,VX,VAX,A1,A2)
      SUBROUTINE LAX29(VA,KCOL,KLD,NEQ,VX,VAX,A1,A2)
      SUBROUTINE LAX2A(VA,KCOL,KLD,KOP,NEQ,VX,VAX,A1,A2)

      SUBROUTINE LAX33(VA,KDIA,KDIAS,NDIA,NEQ,DX,DAX,A1,A2)
      SUBROUTINE LAX34(VA,KDIA,KDIAS,NDIA,NEQ,DX,DAX,A1,A2)
      SUBROUTINE LAX37(VA,KCOL,KLD,NEQ,DX,DAX,A1,A2)
      SUBROUTINE LAX38(VA,KCOL,KLD,NEQ,DX,DAX,A1,A2)
      SUBROUTINE LAX39(VA,KCOL,KLD,NEQ,DX,DAX,A1,A2)
      SUBROUTINE LAX3A(VA,KCOL,KLD,KOP,NEQ,DX,DAX,A1,A2)
```

Subroutines forming the matrix vector product $A^T \cdot x$

```
      SUBROUTINE LTX13(DA,KDIA,KDIAS,NDIA,NEQ,DX,DTX,A1,A2)
      SUBROUTINE LTX17(DA,KCOL,KLD,NEQ,DX,DTX,A1,A2)
      SUBROUTINE LTX19(DA,KCOL,KLD,NEQ1,NEQ2,DX,DTX,A1,A2)
      SUBROUTINE LTX1A(DA,KCOL,KLD,KOP,NEQ,DX,DTX,A1,A2)

      SUBROUTINE LTX23(VA,KDIA,KDIAS,NDIA,NEQ,VX,VTX,A1,A2)
      SUBROUTINE LTX27(VA,KCOL,KLD,NEQ,VX,VTX,A1,A2)
      SUBROUTINE LTX29(VA,KCOL,KLD,NEQ1,NEQ2,VX,VTX,A1,A2)
      SUBROUTINE LTX2A(VA,KCOL,KLD,KOP,NEQ,VX,VTX,A1,A2)

      SUBROUTINE LTX33(VA,KDIA,KDIAS,NDIA,NEQ,DX,DTX,A1,A2)
      SUBROUTINE LTX37(VA,KCOL,KLD,NEQ,DX,DTX,A1,A2)
      SUBROUTINE LTX39(VA,KCOL,KLD,NEQ1,NEQ2,DX,DTX,A1,A2)
      SUBROUTINE LTX3A(VA,KCOL,KLD,KOP,NEQ,DX,DTX,A1,A2)
```

Subroutines forming the weighted scalar product of two vectors $(A \cdot x, y)$

```
SUBROUTINE LWS13(DA,KDIA,KDIAS,NDIA,DX,DY,NX,SP)
SUBROUTINE LWS14(DA,KDIA,KDIAS,NDIA,DX,DY,NX,SP)
SUBROUTINE LWS17(DA,KCOL,KLD,DX,DY,NX,SP)
SUBROUTINE LWS18(DA,KCOL,KLD,DX,DY,NX,SP)
SUBROUTINE LWS1A(DA,KCOL,KLD,KOP,DX,DY,NX,SP)

SUBROUTINE LWS23(VA,KDIA,KDIAS,NDIA,VX,VY,NX,SP)
SUBROUTINE LWS24(VA,KDIA,KDIAS,NDIA,VX,VY,NX,SP)
SUBROUTINE LWS27(VA,KCOL,KLD,VX,VY,NX,SP)
SUBROUTINE LWS28(VA,KCOL,KLD,VX,VY,NX,SP)
SUBROUTINE LWS2A(VA,KCOL,KLD,KOP,VX,VY,NX,SP)

SUBROUTINE LWS33(VA,KDIA,KDIAS,NDIA,DX,DY,NX,SP)
SUBROUTINE LWS34(VA,KDIA,KDIAS,NDIA,DX,DY,NX,SP)
SUBROUTINE LWS37(VA,KCOL,KLD,DX,DY,NX,SP)
SUBROUTINE LWS38(VA,KCOL,KLD,DX,DY,NX,SP)
SUBROUTINE LWS3A(VA,KCOL,KLD,KOP,DX,DY,NX,SP)
```

Y-routines

Y-routines

The subprograms of this subgroup are mainly used during the iterative solution of linear systems of equations in eigenvalue problems. Frequently one writes these algorithms in an abstract form which requires a routine that returns the result of the matrix vector product whereas the matrix itself is not explicitly needed, e.g. in the conjugate gradient algorithm and in the Lanczos procedure.

For this purpose, subprograms with a normalized parameter list are provided which result in the vector `DAX` or `VAX` in the form described above (see Section 3.2 for calling conventions and the use of `COMMON /XYPAR/`). The letter `n`, again, stands for `n=1,3`.

Names and parameter lists of the Y-routines

```
SUBROUTINE YLAXn3(DX,DAX,NEQ,A1,A2)
SUBROUTINE YLAXn4(DX,DAX,NEQ,A1,A2)
SUBROUTINE YLAXn7(DX,DAX,NEQ,A1,A2)
SUBROUTINE YLAXn8(DX,DAX,NEQ,A1,A2)
SUBROUTINE YLAXn9(DX,DAX,NEQ,A1,A2)
SUBROUTINE YLAXnA(DX,DAX,NEQ,A1,A2)

SUBROUTINE YLAX23(VX,VAX,NEQ,A1,A2)
SUBROUTINE YLAX24(VX,VAX,NEQ,A1,A2)
SUBROUTINE YLAX27(VX,VAX,NEQ,A1,A2)
SUBROUTINE YLAX28(VX,VAX,NEQ,A1,A2)
SUBROUTINE YLAX29(VX,VAX,NEQ,A1,A2)
SUBROUTINE YLAX2A(VX,VAX,NEQ,A1,A2)
```

```
SUBROUTINE YLTXn3(DX,DAX,NEQ,A1,A2)
SUBROUTINE YLTXn7(DX,DAX,NEQ,A1,A2)
SUBROUTINE YLTXn9(DX,DAX,NEQ,A1,A2)
SUBROUTINE YLTXnA(DX,DAX,NEQ,A1,A2)

SUBROUTINE YLTX23(VX,VAX,NEQ,A1,A2)
SUBROUTINE YLTX27(VX,VAX,NEQ,A1,A2)
SUBROUTINE YLTX29(VX,VAX,NEQ,A1,A2)
SUBROUTINE YLTX2A(VX,VAX,NEQ,A1,A2)
```

Group `M` – Multigrid components

In this subsection we describe subprograms intended to speed up the solution procedure of the linear or nonlinear systems resulting from the discretization by multigrid techniques. The reader is assumed to be familiar with the basic multigrid notation such as *smoothing iterations*, *prolongation*, *restriction*, etc. In the present release only subprograms to handle standard coarsening are contained. Also only a driver for standard V-, W-, and F-cycles is provided. Drivers for nested iteration, additional step length control, nonlinear multigrid methods, etc. will be provided in the forthcoming release.

We not only give a description of the subprograms in group `M` but also the related`X`- and `Y`- routines, as well as the multigrid related `COMMON`-blocks. Several additional `Y`-routines have to be provided by the user for implementing a multigrid algorithm which are not part of the FEAT2D-package.

First, we recall the list of the multigrid related `COMMON`-blocks, see also section (1.5). The first two of them contain the mesh information for all levels in full correspondence with the blocks `/TRIAD/` and `/TRIAA/`. The parameters in the remaining blocks are described below.

Parameters in `COMMON` blocks (see Section 1.5)

```
      PARAMETER (NNLEV=9)
      COMMON /MGTRD/  KNEL(NNLEV),KNVT(NNLEV),KNMT(NNLEV),
     *                KNVEL(NNLEV),KNVBD(NNLEV)
      COMMON /MGTRA/  KLCVG(NNLEV),KLCMG(NNLEV),KLVERT(NNLEV),
     *                KLMID(NNLEV),KLADJ(NNLEV),KLVEL(NNLEV),
     *                KLMEL(NNLEV),KLNPR(NNLEV),KLMM(NNLEV),
     *                KLVBD(NNLEV),KLEBD(NNLEV),KLBCT(NNLEV),
     *                KLVBDP(NNLEV),KLMBDP(NNLEV)
      COMMON /MGPAR/  ILEV,NLEV,NLMIN,NLMAX,
     *                ICYCLE,KPRSM(NNLEV),KPOSM(NNLEV)
      COMMON /MGTIME/ TTMG,TTS,TTE,TTD,TTP,TTR,IMTIME
```

Input

`ILEV`	Number of currently active level
`NLEV`	Total number of mesh levels
`NLMIN`	Minimum and maximum level used for the
`NLMAX`	multigrid iteration, NLMIN need not be 1.
`NLMAX`	Cycle type `0` F-Cycle `1` V-Cycle `2` W-Cycle `3` higher order (rarely used)

`KPRSM` Number of presmoothing steps for all levels

`KPOSM` Number of postsmoothing steps for all levels

`IMTIME` `>0` CPU-time measured (separately for all multigrid components)
`=1` CPU-time reset at start of multigrid iteration

Output

`TTMG` Total time for multigrid iterations

`TTS` Time for smoothing iterations

`TTE` Time for "exact" coarse grid solver

`TTD` Time for defect evaluation

`TTP` Time for prolongation

`TTR` Time for restriction

Multilevel mesh generation

The standard hierarchy of meshes (standard refinement $h' = h/2$) is generated by successive calls of the mesh refinement routines `SA0`, `SB0`, etc.

```
      SUBROUTINE XMSA0(IMID,IADJ,IVEL,IDISP,IBDP,
     *                 S2DI,S2DB,PARX,PARY,TMAX)
      SUBROUTINE XMSB0(IMID,IADJ,IVEL,IDISP,IBDP,
     *                 S2DI,S2DB,PARX,PARY,TMAX)
      SUBROUTINE XMSB1(NFINE,IMID,IADJ,IVEL,IDISP,IBDP,
     *                 MFILE,CFILE,IFMT,PARX,PARY,TMAX)
      SUBROUTINE XMSCL
```

Explanation
The routines `XMSA0`,`XMSB0`,`XMSB1` generate a sequence of `NLEV` meshes by successive standard refinement. The arguments are as described in section `S`.

The routine `XMSCL` (without arguments) resets the `COMMON`-blocks `/MGTRA/` and `/MGTRD/` (analogue of `XSCL`).

Multilevel problem generation

The following routines are used to generate the finite element matrices and right hand sides used during the multigrid iteration for all levels `NLMIN` to `NLMAX`, i.e. for the levels employed during the iteration. Usually, the matrix and right side in the finest level correspond to the original discretization scheme. The names of the subprograms are self-explaining as `XMxxxx` means multiple call of `Xxxxx`. The parameters `KLA`, `KLCOL`, etc., contain the numbers of the arrays and the corresponding pointer vectors for all levels. For all remaining parameters see section `A` and section `V`, respectively. Notice that the routines `XMVxx` usually are needed for nonlinear multigrid iterations and are provided in the current release for compatibility with future versions only, see also the multigrid example in the Appendix.

```
      SUBROUTINE XMAP3 (KLDIA,KLDIAS,KNDIA,KNA,KNEQ,ELE,ISYMM)
      SUBROUTINE XMAP7 (KLCOL,KLLD,KNA,KNEQ,ELE,ISYMM)
      SUBROUTINE XMAA03(KLA,KLDIA,KLDIAS,KNDIA,KNA,KNEQ,NBLOC,ICLEAR,
     *                  ELE,COEFF,BCON,KAB,KABN,ICUB,ISYMM,CARR,BSNGL)
      SUBROUTINE XMAA07(KLA,KLCOL,KLLD,KNA,KNEQ,NBLOC,ICLEAR,ELE,
     *                  COEFF,BCON,KAB,KABN,ICUB,ISYMM,CARR,BSNGL)
      SUBROUTINE XMAB03(KLA,KLDIA,KLDIAS,KNDIA,KNA,KNEQ,NBLOC,ICLEAR,
     *                  ELE,COEFF,BCON,KAB,KABN,ICUB,ISYMM,CARR,BSNGL)
      SUBROUTINE XMAB07(KLA,KLCOL,KLLD,KNA,KNEQ,NBLOC,ICLEAR,ELE,
     *                  COEFF,BCON,KAB,KABN,ICUB,ISYMM,CARR,BSNGL)
      SUBROUTINE XMVA0 (KLB,KNEQ,NBLOC,ICLEAR,ELE,
     *                  COEFF,BCON,KB,KBN,ICUB,CARR,BSNGL)
      SUBROUTINE XMVA1 (KLB,KNEQ,NBLOC,ICLEAR,TMAX,DPAR1,DPAR2,IBCT,
     *                  ELE,COEFF,BCON,KB,KBN,ICUB,CARR,BSNGL)
      SUBROUTINE XMVB0 (KLB,KNEQ,NBLOC,ICLEAR,ELE,
     *                  COEFF,BCON,KB,KBN,ICUB,CARR,BSNGL)
      SUBROUTINE XMVB1 (KLB,KNEQ,NBLOC,ICLEAR,TMAX,DPAR1,DPAR2,IBCT,
     *                  ELE,COEFF,BCON,KB,KBN,ICUB,CARR,BSNGL)
```

Multilevel drivers

The standard multigrid driver for the `F`-, `V`-, and `W`-cycle is provided by the subprogram `M010`, and is invoked by the `X`- routines `XM010` or `XM017`. The functionality of the control parameters (e.g. the stopping criterion) is kept compatible with that for the iterative solvers described in section `I`. For the description of the parameters let `NEQM` denote the sum of all numbers of unknowns for the levels `NLMIN` through `NLMAX`.

```
      SUBROUTINE M010 (DX,DB,DD,KOFFX,KOFFB,KOFFD,KNEQ,NIT,ITE,EPS,
     *                 DAX,DPROL,DREST,DPRSM,DPOSM,DEX,DBC,KIT0,KIT,IREL)
```

Parameters Input

DX	R*8	DIMENSION DX(*)
DB	R*8	DIMENSION DB(*)
DD	R*8	DIMENSION DD(*) Starting addresses of vectors containing the solution and the right hand side, DD is used as auxiliary vector only
KOFFX	I*4	DIMENSION KOFFX(NLEV)
KOFFB	I*4	DIMENSION KOFFB(NLEV)
KOFFD	I*4	DIMENSION KOFFD(NLEV)

KOFFD	I*4	The actual starting address of DX on level ILEV is DX(1+KOFFX(ILEV)) (analogously for DB and DD) Total space required for all vectors is NEQM DX(1+KOFFX(NLMAX)) contains initial solution, DB(1+KOFFB(NLMAX)) contains right hand side
KNEQ	I*4	Number of equations for all levels
NLMAX	I*4	Iteration uses levels NLMIN through NLMAX
NLMIN	I*4	NLMAX is the finest level
NIT	I*4	Maximum number of iterations One iteration is considered as completed after reaching the finest level again
EPS	R*8	Desired precision IREL=0: Stop if !!RES!! < EPS IREL=1: Stop if !!RES!!/!!RES0!! < EPS and a minimum of ITE iterations is performed
KPRSM	I*4	Number of pre-smoothing steps for all levels
KPOSM	I*4	Number of post-smoothing steps for all levels
ICYCLE	I*4	<0: special cycle types (not yet implemented) =0: F-Cycle =1: V-Cycle =2: W-Cycle >2: Cycle of higher order
DAX	SUBR	DAX(DX,DAX,NEQ,A1,A2) Returns DAX := A1*A*DX+A2*DAX
DPROL	SUBR	DPROL(DX1,DX2) Returns DX2 := Prolongation(DX1) to higher level
DREST	SUBR	DREST(DD1,DD2) Returns DD2 := Restriction(DD1) to lower level
DPRSM	SUBR	DPRSM(DX,DB,DD,NEQ,NPRSM) Returns DX after NPRSM:=KPRSM(ILEV) pre-smoothing steps DD can be used as auxiliary vector
DPOSM	SUBR	Same as above, used for post-smoothing
DEX	SUBR	DEX(DX,DB,DD,NEQ) Returns "exact" solution
DBC	SUBR	DBC(DX,NEQ) Copies boundary data onto components of DX
KIT0	I*4	auxiliary vectors of length NLMAX
KIT	I*4	

Output

DX	R*8	Solution vector on DX(1+KOFFX(NLMAX))
ITE	I*4	Number of iterations
IER	I*4	Error indicator

```
      SUBROUTINE XM017(KLA,KLCOL,KLLD,LX,LB,KNEQ,NIT,ITE,
     *                 EPS,YAX,YPROL,YREST,YPRSM,YPOSM,YEX,YBC,IDISP)
      SUBROUTINE XM010(LX,LB,KNEQ,NIT,ITE,EPS,
     *                 YAX,YPROL,YREST,YPRSM,YPOSM,YEX,YBC,IDISP,IREL)
```

Parameters Input

KLA	I*4	Numbers of arrays for all levels
KLCOL	I*4	Numbers pointer vectors for all levels
KLLD	I*4	see description of storage technique 7
LX	I*4	Number of solution vector (finest level)
LB	I*4	Number of right hand side (finest level)

Output

IER	I*4	Error indicator

The Y-routines in the parameter lists are dummy names for the EXTERNAL-routines realizing DAX, DPROL, DREST, DPRSM, DPOSM, DEX, and DBC, described above.

Explanation

XM017 stores the numbers for the arrays and pointer vectors in KLA and the other array descriptors onto KXYPAR(100+I), I>=0) in order to cooperate with the standard Y-routines, described below. Then XM010 is called. XM010 provides the workspace vectors for all lower levels, and stores the offsets with respect to the starting address on DWORK(1) onto the offset vectors KOFFX, etc. Then the standard driver M010 is invoked.

Prolongations - restrictions

In the present version only standard prolongations and restrictions for the simplest finite elements are provided, namely piecewise linear and piecewise bilinear elements (E001 and E011). The names of the routines in group M are MPsnn and MRsnn, respectively. Here, P stands for prologation and R stands for restriction. The number s refers to the refinement procedure, e.g. SA0 or SA1, and, thus, to the numbering scheme for vertices and midpoints of edges. Finally, nn is the number of the element.

```
SUBROUTINE MP001(DU1,DU2,KVERT1,KVERT2,KADJ1,NVT1,NEL1)
SUBROUTINE MP011(DU1,DU2,KVERT1,KVERT2,KADJ1,KADJ2,NVT1,NEL1)
SUBROUTINE MP111(DU1,DU2,NEL1,KVERT1,KVERT2,KMAVT,KMAADJ,KMAVE)
SUBROUTINE MR001(DU2,DU1,KVERT2,KADJ2,NVT1,NEL2,NEL1)
SUBROUTINE MR011(DU2,DU1,KVERT2,KADJ2,NVT1,NEL2)
SUBROUTINE MR111(DU2,DU1,NEL1,KVERT2,KVERT1,KMAVT,KMAADJ,KMAVE)
```

Explanation

The routines assume standard coarsening or standard refinement, respectively. The prolongation programs return a fine grid vector DU2 from a coarse grid vector DU1. Analogously, the restriction routines calculate a coarse grid vector DU1. Also in the names of the remaining parameters, 1 stands for coarse grid information and 2 stands for fine grid information. All other characters in the remaining parameter names correspond to the mesh information as described in section S.

Y-routines for prolongations and restrictions

The subprograms MPsnn and MRsnn are invoked via the Y-routines listed below.

```
SUBROUTINE YMP001(DX1,DX2)
SUBROUTINE YMP011(DX1,DX2)
SUBROUTINE YMP111(DX1,DX2)
SUBROUTINE YMR001(DX2,DX1)
SUBROUTINE YMR011(DX2,DX1)
SUBROUTINE YMR111(DX2,DX1)
```

Explanation

These subprogram names are passed as EXTERNAL arguments to the driver routines XM017 or XM010. Again, DX2 denotes a fine grid vector and DX1 is a coarse grid vector. The Y-routines obtain the information about the current level ILEV on COMMON /MGPAR/ and the mesh information on /MGTRD/ and MGTRA.

Multilevel I/O

The routines listed in this subsection are the multilevel analogues of the I/O-subprograms XOWS, XORS, and XGOWSM.

```
SUBROUTINE XMOWS (MFILE,CCFILE,IFMT)
SUBROUTINE XMORS (MFILE,CCFILE,IFMT)
SUBROUTINE XGMOWS(MFILE,CCFILE)
```

Parameters Input

MFILE	I*4	Output unit
CCFILE	C**	DIMENSION CCFILE(NLEV) Output file names for all levels
IFMT	I*4	= 1 Formatted I/O
		= 0 Unformatted I/O

Explanation

The routines XMOWS, XMORS write or read mesh information for all levels to/ from files in FEAT2D-format. XGMOWS writes the (restricted) mesh information needed for graphical representation in MOVIE.BYU-format.

Group `N` – Auxiliary routines for global and local numbers of d.o.f.

The functions and subroutines are used as auxiliary routines, e.g., during the assembly of element and global stiffness matrices. They return information about the local and global size of the problem depending on the data of the triangulation and on the type of element in use. The dimensions of the arrays describing the subdivision are passed in `COMMON /TRIAD/`.

Parameters in `COMMON` blocks

```
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
```

List of available subprograms

```
      INTEGER FUNCTION NDFL(IELTYP)
      INTEGER FUNCTION NDFG(IELTYP)
      SUBROUTINE NDFGL(IEL,IPAR,IELTYP,KVERT,KMID,KDFG,KDFL)
```

The first two routines return the number of d.o.f. on each element (Local) and on the whole domain (Global), depending on the element number `IELTYP`. The correspondence of local and global d.o.f.s on element `IEL` of the triangulation is calculated by `NDFGL`.

Parameters Input

`IEL`	`I*4`	Number of current element of the triangulation
`IPAR`	`I*4`	Switch - controls output on `KDFG` and `KDFL` (see below)
`IELTYP`	`I*4`	Number of element
`KVERT`	`I*4`	`DIMENSION KVERT(NNVE,NEL)` Numbers of vertices of each element
`KMID`	`I*4`	`DIMENSION KMID(NNVE,NEL)` Numbers of midpoints of edges, if necessary

Output

`KDFG`	`I*4`	`DIMENSION KDFG(NNBAS)`
		Global degrees of freedom on element `IEL`
		`KDFG` is sorted if `IPAR >= 0`

`KDFL`	`I*4`	`DIMENSION KDFL(NNBAS)` Local degrees of freedom corresponding to `KDFG`, `KDFL` is determined only if `IPAR = 1`

Group 0 – Input/Output

The programs of group 0 are used for Input/Output. The first subgroup contains routines that are used for protocol and error messages and for subprogram tracing. These subprograms are only active if the I/O files MERR, MPRT, MSYS, and MTRC, as well as the message file FEAT.MSG have been successively opened by the program ZINIT. The second subgroup deals with normalized I/O for single arrays. The subroutines in subgroup 3 use the I/O routines in subgroup 2 to read and write the information of a whole subdivision of the domain in normalized form.

Subgroup 1 – Messages

Parameters Input

IER	I*4	Number of error message (routine OERR)
IMSG	I*4	Number of protocol message (routine OMSG)
SUB	C*6	Number of calling routine
VER	C*8	Date of version of calling routine mm/dd/yy (routine OTRC)

```
SUBROUTINE OERR(IER,SUB)
```

Invoked by WERR to write error messages to unit MERR. Formats for messages are contained in the message file FEAT.MSG , parameters are passed in CPARAM in COMMON block /CHAR/ .

```
SUBROUTINE OMSG(IMSG,SUB)
```

Used to display protocol or system messages on unit MPRT and MSYS, respectively. Formats for messages are contained in the message file FEAT.MSG, parameters are passed in CPARAM in COMMON block /CHAR/ .

```
SUBROUTINE OTRC(SUB,VER)
```

Writes the name and the date of the version of subprograms to unit MTRC. Tracing occurs only if ICHECK in COMMON block /ERRCTL/ is set to the values 997, 998, or 999. For ICHECK=999 even elementary auxiliary subroutines are traced, for ICHECK=997 only higher level subprograms.

Subgroup 2 – Input/Output of arrays

This subgroup is used to save or read single arrays of different data types.

Names of the subprograms

The subprogram names are of the form `XOaA` and `XOaS`, for routines performing I/O for arrays on the workspace and `OaAt` for programs that directly obtain the address of the array as input.
Here, `t` denotes the data type

- 1 `DOUBLE PRECISION`,
- 2 `REAL`,
- 3 `INTEGER`,

and `a` stands for the desired action,

- `W` for writing,
- `R` for reading.

Further, the program `OFO` is used to open an I/O file, either directly by the user or implicitly by the programs `XORA` and `XOWA`. The `O`-routines assume that the I/O file is already open. The files may either be written `FORMATTED` or `UNFORMATTED`, depending on the parameter `IFMT`. For formatted writing the current `FORMAT` in `FMT(t)` is used, for input the `FORMAT` stored on the input file is used. The array `FMT` of type `CHARACTER` is contained in `COMMON /CHAR/`. An array is written to or read from a file in the following form.

1. Formatted I/O

Record 1

`ARR`, `CFORM`, `ITYPE`, `ILEN`

written in `FORMAT (2A10,2I10)`, where `ARR` is the array name (see above), `CFORM` is the `FORMAT` for the subsequent data, `ITYPE=1`, `2`, or `3` is the data type, and `ILEN` is the number of elements of array `ARR`. The following records contain the elements of the array in `FORMAT CFORM`.

2. Unformatted I/O

Record 1

`ARR`, `ITYPE`, `ILEN`, `ILEN8`, `IRECL8`

written `FORMAT` free, where `ARR` is the array name (see above), `ITYPE=1,2`, or `3` is the data type, `ILEN` is the number of elements of array `ARR`, and `ILEN8` is the number of `DOUBLE PRECISION` storage locations needed for array `ARR`.

The maximum record length is determined by the value IRECL8 in COMMON /OUTPUT/ which is set during initialization to a machine dependent value (IRECL8=512 by default). With this information arrays can be read in exactly as they were written. The following records contain the elements of the array FORMAT free.

Parameters Input

MFILE	I*4	Number of I/O unit
CFILE	C**	Name of I/O file
		For CFILE.EQ.'SCRATCH' an unnamed scratch file is used
IFMT	I*4	=1 Formatted I/O
		=0 Unformatted I/O
ARR	C*6	Name of array read from or written to unit MFILE

Output

LNR	I*4	Number of array (for X-routines)
ARR	C*6	Name of array, for messages only
DX	R*8	
VX	R*4	Arrays to be read from or written to unit MFILE
KX	I*4	
CFILE	C**	Name of I/O file
IFMT	I*4	=1 Formatted I/O

List of available subprograms

Open I/O file

```
      SUBROUTINE OFO(MFILE,CFILE,IFMT)
```

Read array from file

```
      SUBROUTINE XORA(LNR,ARR,MFILE,CFILE,IFMT)
      SUBROUTINE ORA1(DX,ARR,MFILE,IFMT)
      SUBROUTINE ORA2(VX,ARR,MFILE,IFMT)
      SUBROUTINE ORA3(KX,ARR,MFILE,IFMT)
```

Write array onto file

```
      SUBROUTINE XOWA(LNR,ARR,MFILE,CFILE,IFMT)
      SUBROUTINE OWA1(DX,ARR,MFILE,IFMT)
      SUBROUTINE OWA2(VX,ARR,MFILE,IFMT)
      SUBROUTINE OWA3(KX,ARR,MFILE,IFMT)
```

Notice that no routines of this subgroup rewind the I/O file.

3. Input/Output of subdivisions

This third subgroup serves for storing and reading of whole subdivisions of the domain. The information about dimensions of the arrays describing the triangulation and the numbers for the arrays on `DWORK` are contained on `COMMON /TRIAD/` and `/TRIAA/`, see Section 1.5. The parameters `MFILE`, `CFILE`, and `IFMT` are used as above.

Triangulations are usually generated automatically, frequently starting from a coarse initial subdivision. To read coarse mesh information form a file, the subprogram `XORSC` is used. The input file only contains information necessary for the application of the subprograms of group `S`. For the description of the arrays and parameters see Section 1.5.

```
      SUBROUTINE  XORSC(MFILE,CFILE)
```

Contents of the input file for `XORSC`:

* - FORMAT used for input
`NMT` usually is `0`.

Comment line
Comment line
`NEL NVT NMT NVE NBCT`
Comment line
`((DCORVG(IDIM,IVT),IDIM=1,2),IVT=1,NVT)`
Comment line
`((KVERT(IVE,IEL),IVE=1,NVE),IEL=1,NEL)`
Comment line
`(KNPR(IVMT),IVMT=1,NVT+NMT)`
Comment line
`((KMM(I,IBCT),I=1,2),IBCT=1,NBCT)`

Example

Unit cube with a hole – $\Omega = (0,1)^2 \setminus [0.3, 0.7]^2$

The following programs are used to read and write complete subvisions generated by one or more of the subprograms of group `S`. `XOWS` checks which of the arrays describing a triangulation are really generated and, of course, only these files are stored. Similarly, `XORS` reads the information about dimensions, allocates all necessary arrays on the workspace and reads the contents of the arrays from the I/O file. Again, formatted or format-free I/O may be used.

Notice that `XORS` and `XOWS` do not rewind the file.

```
      SUBROUTINE XORS(MFILE,CFILE,IFMT)
      SUBROUTINE XOWS(MFILE,CFILE,IFMT)
```

Group R – Reorganization

The subprograms of group R are intended to rearrange a given data structure. For example, the storage technique may be changed or the vertices of a subvision may be ordered with respect to certain criteria. A particular application is the compression (=deletion of zero entries) of matrices after the implementation of boundary conditions.

Subgroup RC – compression of matrices

Names of the subprograms

The subprogram names are of the form RCns, where n stands for the data type,

1 DOUBLE PRECISION,

2 REAL,

and s refers to the storage technique.

All of the subprograms deal with block matrices. In the X-routines the numbers of all NBLOC matrices are passed in LA(NBLOC), in the R-routines only one starting address is given, DA(1) or VA(1), and the offsets relative to the starting address in KOFF(NBLOC). If at a certain position the entries of all block matrices possess a modulus below a given tolerance TOL, the entries are removed and the common pointer vectors are updated.

Parameters Input

LA	I*4	DIMENSION LA(NBLOC) Numbers of block matrices
KLD	I*4	DIMENSION KLD(NEQ+1) Pointer to start of rows
KDIAS	I*4	DIMENSION KDIAS(NDIA+1) Pointer to start of diagonal rows
NEQ	I*4	Number of equations
NBLOC	I*4	Number of block matrices
TOL	R*8	Entries are deleted if modulus is less than TOL in all blocks
IDISP	I*4	IDISP=1: Free space on DWORK is released after compression (used in X-routines only)
ARR1	C*6	DIMENSION ARR1(NBLOC) Names for block matrices in DA (VA)
ARR2	C*6	Name for column pointer matrix (KCOL) ARR1 and ARR2 used for messages only

Output

DA	R*8	DIMENSION DA(NA) Double precision matrix
VA	R*4	DIMENSION VA(NA) Single precision matrix

KCOL	I*4	DIMENSION KCOL(NA) Column pointer
KDIA	I*4	DIMENSION KDIA(NDIA) Diagonal offset pointer
KDIAS	I*4	DIMENSION KDIAS(NDIA+1) Pointer to start of diagonal rows
NDIA	I*4	Number of diagonal rows
NA	I*4	Number of entries in matrix

List of available subprograms

```
SUBROUTINE XRC13(LA,LDIA,LDIAS,NDIA,NA,NEQ,NBLOC,TOL,IDISP,ARR)
SUBROUTINE  RC13(DA,KDIA,KDIAS,NDIA,NA,NEQ,NBLOC,KOFF,TOL)
SUBROUTINE XRC23(LA,LDIA,LDIAS,NDIA,NA,NEQ,NBLOC,TOL,IDISP,ARR)
SUBROUTINE  RC23(VA,KDIA,KDIAS,NDIA,NA,NEQ,NBLOC,KOFF,TOL)

SUBROUTINE XRC17(LA,LCOL,LLD,NA,NEQ,NBLOC,TOL,IDISP,ARR1,ARR2)
SUBROUTINE  RC17(DA,KCOL,KLD,NA,NEQ,NBLOC,KOFF,TOL)
SUBROUTINE XRC27(LA,LCOL,LLD,NA,NEQ,NBLOC,TOL,IDISP,ARR1,ARR2)
SUBROUTINE  RC27(VA,KCOL,KLD,NA,NEQ,NBLOC,KOFF,TOL)

SUBROUTINE XRC19(LA,LCOL,LLD,NA,NEQ,NBLOC,TOL,IDISP,ARR1,ARR2)
SUBROUTINE  RC19(DA,KCOL,KLD,NA,NEQ,NBLOC,KOFF,TOL)
SUBROUTINE XRC29(LA,LCOL,LLD,NA,NEQ,NBLOC,TOL,IDISP,ARR1,ARR2)
SUBROUTINE  RC29(VA,KCOL,KLD,NA,NEQ,NBLOC,KOFF,TOL)
```

Group S – Generation of subdivisions

The programs of this group are devoted to the generation and modification of subdivisions. Meshes usually are constructed from coarse grids through regular or adaptive mesh refinements. Information about coarse meshes can be read from a data file using XORSC.

In many applications only the coordinates of the vertices, the numbers of vertices forming each element and the information whether or not a vertex is situated on the boundary is needed. For regular refinement, this information is generated by the central routines SA0 (XSA0), for triangular meshes, and SB0 (XSB0), for quadrilateral meshes. Further information on numbers of midpoints of edges or on the numbers of elements meeting at a vertex is generated (for already refined meshes) by the programs S2M (XS2M) and S2V (XS2V). Finally, the routines SVEB (XSVEB) extract information on the boundary vertices and elements needed for easy handling of boundary conditions.

For a detailed description of the following parameters see Sections 1.4 and 1.5.

Parameters Input

PARX	FUNC	
PARY	FUNC	Parametrization of the domain
TMAX	FUNC	

Input/Output

DCORVG	R*8	DIMENSION DCORVG(2,NVT) Coordinates of vertices
DCORMG	R*8	DIMENSION DCORMG(2,NVT) Coordinates of midpoints of edges
KVERT	I*4	DIMENSION KVERT(NNVE,NEL) Numbers of vertices
KADJ	I*4	DIMENSION KADJ(NNVE,NEL) Numbers of adjacent elements
KMID	I*4	DIMENSION KMID(NNVE,NEL) Numbers of edges (midponts)
KNPR	I*4	DIMENSION KNPR(NVT+NMT) Nodal properties
KMM	I*4	DIMENSION KMM(2,NBCT) Boundary points with maximum and minimum parameter

Output

KVEL	I*4	DIMENSION KVEL(NVEL,NVT) Numbers of elements meeting at a vertex
KMEL	I*4	DIMENSION KMEL(2,NMT) Numbers of elements meeting at an edge
KVBD	I*4	DIMENSION KVBD(NVBD) Vertices on the boundary
KEBD	I*4	DIMENSION KEBD(NVBD) Elements at the boundary

KBCT	I*4	DIMENSION KBCT(NBCT+1) Pointer vector for KVBD and KEBD

Parameters in COMMON blocks (see Section 1.5)

Input/Output

```
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /TRIAA/  LCORVG,LCORMG,LVERT,LMID,LADJ,LVEL,LMEL,LNPR,LMM,
     *                LVBD,LEBD,LBCT,LVBDP,LMBDP
```

Further parameters will be explained together with the specific routines.

<u>The central routines XSAOX and XSBOX</u>

For creating a regularly refined mesh and allocating all arrays on the workspace vector the user may invoke the subprograms XSAOX, for triangular meshes and XSBOX for quadrilaterals. The second letter X denotes that these routines again only invoke the X-routines XSA0, XSB0, XS2M, and XS2V.

All elements are subdivided into four subelements. For triangular elements, for each edge an interior point is determined by the subdivision programs S2DI, for interior edges, and S2DB, on the boundary. This new point, frequently the midpoint of the edge becomes a new vertex in the refined mesh. The three new points are connnected by lines what divides the triangle into four subelements. The interior element is given the former number of the element, the other three elements obtain the new numbers NEL+1, NEL+2, and NEL+3. Then NEL is increased by 3. New vertices obtain new numbers, starting at NVT+1, old vertices keep their numbers.

Similarly, for quadrilaterals, all four edges are subdivided and connected. The subelement containing the "first" vertex of the original element keeps the old number, the other three subelements are enumerated as described above. The old vertices keep their number, then the new vertices on edges of the old subdivision and, finally, the new interior vertices are enumerated. In order to modify the mesh the user may provide subprograms S2DI and S2DB by himself.

S2DI	SUB	S2DI(X1,Y1,X2,Y2,XT,YT) gives the coordinates of the new vertex on the line (X1,Y1)-(X2,Y2)
S2DB	SUB	S2DB(X1,X2,XT,BMM,IBCT,PARX,PARY,TMAX(IBCT)) gives the parameter of the new boundary vertex on the boundary segment joining the vertices with parameter X1 and X2 BMM=.TRUE. means that X1,X2 are the minimum and maximum parameter on boundary component IBCT

If one chooses the standard EXTERNAL programs

```
      SUBROUTINE S2DB0(X1,X2,XT,BMM,IBCT,PARX,PARY,TMAX(IBCT))
      SUBROUTINE S2DI0(X1,Y1,X2,Y2,XT,YT)
```

a uniform subdivision using midpoints of edges is constructed.

Parameters Input

NFINE	I*4	Desired number of regular subdivisions of the given mesh
IMID	I*4	> 0: Determine numbers of midpoints (array KMID)
		> 1: Determine coordinates of midpoints (array DCORMG)
IADJ	I*4	= 1: Determine adjacent elements (KADJ) on finest mesh
IVEL	I*4	= 1: Determine numbers of elements meeting at each vertex (array KVEL)
IDISP	I*4	= 1: Release all unused arrays on DWORK on return
IBDP	I*4	=-1: Adjust only NVBD and KBCT after completion
		>=0: Store numbers of boundary vertices on KVBD
		>=1: Calculate sorted arrays KVBD and KEBD
		>=2: Store parameter values for boundary vertices (and midpoints) on DVBDP and DMBDP
S2DI	SUB	Determination of intermediate points on edges (see above)
S2DB	SUB	

```
      SUBROUTINE XSAOX(NFINE,IMID,IADJ,IVEL,IDISP,IBDP,S2DI,S2DB,
     *                 PARX,PARY,TMAX)
      SUBROUTINE XSBOX(NFINE,IMID,IADJ,IVEL,IDISP,IBDP,S2DI,S2DB,
     *                 PARX,PARY,TMAX)
```

The following programs are invoked by XSAOX and XSBOX and are not discussed in detail here.

Generation of uniform subdivision

```
      SUBROUTINE XSAO(NFINE,IDISP,S2DI,S2DB,PARX,PARY,TMAX)
      SUBROUTINE SAO (DCORVG,KVERT,KADJ,KNPR,KMM,
     *                NFINE,NNEL,NNVT,S2DI,S2DB,PARX,PARY,TMAX)
      SUBROUTINE XSBO(NFINE,IDISP,S2DI,S2DB,PARX,PARY,TMAX)
      SUBROUTINE SBO (DCORVG,KVERT,KADJ,KNPR,KMM,
     *                NFINE,NNEL,NNVT,S2DI,S2DB,PARX,PARY,TMAX)
```

Generation of midpoint information (KMID, DCORMG)

```
      SUBROUTINE XS2M(IMID,IADJ,IDISP,S2DI,S2DB,PARX,PARY,TMAX)
      SUBROUTINE S2M (DCORVG,DCORMG,KVERT,KMID,KADJ,KNPR,KMM,
     *               IMID,S2DI,S2DB,PARX,PARY,TMAX)
```

Generation of vertex element connectivity (KVEL)

```
      SUBROUTINE XS2V
      SUBROUTINE S2V (KVERT,KADJ,KVEL,ICHK)
```

Generation of adjacent element information (KADJ)

```
SUBROUTINE XS2A
SUBROUTINE S2A (KVERT,KADJ,KAUX1,KAUX2)
```

Check of a subdivision

The programs `XS2C` and `S2C` perform elementary consistency checks for a given subdivision. For a list of possible errors see error messages 150 to 163.

```
SUBROUTINE XS2C(PARX,PARY,TMAX)
SUBROUTINE S2C (DCORVG,KVERT,KADJ,KNPR,BADJ,PARX,PARY,TMAX)
```

Parameters Input

`BADJ`	`L*4`	`.FALSE.` means that array `KADJ` is not available, skip all consistency checks concerning adjacent elements

Information on boundary vertices and elements

The program `XSVEB` and the invoked subprograms `SVEB` and `SIVB` extract information about the vertices on the boundary and, additionally, about the elements adjacent to the boundary of the domain. The information is returned in the arrays `KVBD`, `KEBD`, and `KBCT` (`COMMON /TRIAA/`), the number of boundary vertices (and elements) in `NVBD` (`COMMON /TRIAD/`). For details see Section 1.5. This information can be used to implement boundary conditions very easily.

Parameters Input

`IPAR`	`I*4`	`-1`: Determine number `NVBD` only `0`: Determine vertices at the boundary (`KVBD`) `1`: Sort vertices in `KVBD` with respect to the parameter in increasing order (within each boundary component) and determine `KEBD`
`TMAX`	`FUNC`	as above

```
SUBROUTINE  XSVEB(IPAR,TMAX)
```

Group V – Linear forms

The programs of group V deal with the calculation of linear forms corresponding to integrals of the form

$$b_i = \int_\Omega \sum_\alpha c_\alpha(x)\partial^\alpha \phi_i \, dx \, ,$$

or

$$b_i = \int_{\partial\Omega} \sum_\alpha c_\alpha(s)\partial^\alpha \phi_i \, ds \, .$$

The notation ϕ_i stands for the basis functions of the space of test functions. The conventions in the notation and parameter lists are similar to those of group A for bilinear forms. The structure of the linear form, number of additive terms and abbreviations for the partial derivatives applied to the basis functions is contained in the arrays KB(.,IBLOC) and KBN(IBLOC), for each of the NBLOC block vector separately. The coefficient function is of the form

```
      DOUBLE PRECISION FUNCTION COEFF(X,Y,IA,IBLOC,BFIRST)
```

cf. group A. Clearly, for linear forms, only one multiindex of derivatives abbreviated by the number IA is passed.

Names of the subprograms

The names of the programs are of the form VAv and VBv, where VA. is used for triangular elements and VB. for quadrilaterals. The character v denotes the version,

v=0 for area integrals,

v=1 for boundary integrals.

Parameters Input

LB	I*4	DIMENSION LB(NBLOC) Handles of block vectors (for X-routines only)
NEQ	I*4	Dimension of each block vector
NBLOC	I*4	Number of block vectors
ICLEAR	I*4	=1 Old entries are set to zero =0 New elements are added to old ones
KOFF	I*4	DIMENSION KOFF(NBLOC) Offsets of starting address of vector block IBLOC relative to starting address of first block
KVERT	I*4	DIMENSION KVERT(NNVE,NEL) Vertices of elements
KMID	I*4	DIMENSION KMID(NNVE,NEL) Numbers of midpoints (edges) of elements, if necessary
DCORVG	R*8	DIMENSION DCORVG(2,NVT) Coordinates of vertices

KNPR	I*4	DIMENSION KNPR(NVT+NMT) Nodal properties, see Section 1.5
IBCT	I*4	Boundary component (for routines VA1 and VB1 only)
KBCT	I*4	DIMENSION KBCT(NBCT+1) Additional arrays describing the triangulation, see Section 1.5 (for routines VA1 and VB1 only)
KVBD	I*4	DIMENSION KVBD(NVBD)
KEBD	I*4	DIMENSION KEBD(NVBD)
IVBDn	I*4	n=1,2 Minimum and maximum index on KVBD as calculated by routine SIVB (for routines VA1 and VB1 only)
TMAX	FUN	Maximum parameter of boundary curves (for routines VA1 and VB1 only)
DPARn	R*8	n=1,2 Description of the boundary part (for routines XVA1 and XVB1 only)
ELE	SUB	Name of the element subprogram
COEFF	FUN	Coefficient function, as described above
BCON	L*4	DIMENSION BCON(NBLOC) BCON(IBLOC).EQ..TRUE. means that block IBLOC has constant coefficients
COECON	I*4	DIMENSION COECON(NNDER,NBLOC) Auxiliary array (for constant coefficients)
KB	I*4	DIMENSION KB(NNAB,NBLOC) Abbreviations of partial derivatives applied to basis functions
KBN	I*4	DIMENSION KBN(NBLOC) Numbers of additive terms in each linear form
ICUB	I*4	Number of cubature formula, see group C
ARR	C*6	DIMENSION ARR(NBLOC) Names of block vectors, for messages only
BSNGL	B*4	=.TRUE. Change vector type to SINGLE PRECISION

Output

DB	R*8	DIMENSION DB(NEQ) Resulting block vector, DOUBLE PRECISION
VB	R*4	DIMENSION VB(NEQ) Resulting block vector, REAL

Parameters in COMMON blocks

```
      PARAMETER (NNCUBP=36)

      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /TRIAA/  LCORVG,LCORMG,LVERT,LMID,LADJ,LVEL,LMEL,LNPR,LMM,
     *                LVBD,LEBD,LBCT,LVBDP,LMBDP
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
```

Exchange of information with COEFF

```
      COMMON /COAUX1/ KDFG(NNBAS),KDFL(NNBAS),IDFL
```

List of available subprograms – Triangular elements

```
      SUBROUTINE XVA0(LB,NEQ,NBLOC,ICLEAR,ELE,
     *                COEFF,BCON,KB,KBN,ICUB,ARR,BSNGL)
      SUBROUTINE  VA0(DB,NBLOC,KOFF,KVERT,KMID,DCORVG,KNPR,
     *                ELE,COEFF,BCON,COECON,KB,KBN,ICUB)

      SUBROUTINE XVA1(LF,NEQ,NBLOC,ICLEAR,TMAX,DPAR1,DPAR2,IBCT,
     *                ELE,COEFF,BCON,KB,KBN,ICUB,ARR,BSNGL)
      SUBROUTINE  VA1(DB,NBLOC,KOFF,KVERT,KMID,DCORVG,KNPR,
     *                KBCT,IBCT,KVBD,KEBD,IVBD1,IVBD2,
     *                ELE,COEFF,BCON,COECON,KB,KBN,ICUB)
```

List of available subprograms – Quadrilateral elements

```
      SUBROUTINE XVB0(LB,NEQ,NBLOC,ICLEAR,ELE,
     *                COEFF,BCON,KB,KBN,ICUB,ARR,BSNGL)
      SUBROUTINE  VB0(DB,NBLOC,KOFF,KVERT,KMID,DCORVG,KNPR,
     *                ELE,COEFF,BCON,COECON,KB,KBN,ICUB)

      SUBROUTINE XVB1(LF,NEQ,NBLOC,ICLEAR,TMAX,DPAR1,DPAR2,IBCT,
     *                ELE,COEFF,BCON,KB,KBN,ICUB,ARR,BSNGL)
      SUBROUTINE  VB1(DB,NBLOC,KOFF,KVERT,KMID,DCORVG,KNPR,
     *                KBCT,IBCT,KVBD,KEBD,IVBD1,IVBD2,
     *                ELE,COEFF,BCON,COECON,KB,KBN,ICUB)
```

Group W – Error handling

The programs of group W are intended for handling of errors occurring in FEAT2D subprograms. In particular, they are used to display error messages, to dump the contents of the COMMON block /TABLE/ (see group Z) containing information on the arrays currently allocated on the workspace, or to selectively list the contents of variables and arrays in COMMON blocks. In the present version of FEAT2D only an elementary standard routine is provided.

```
      SUBROUTINE WERR(IER0,SUB0)
```

Parameters Input

IER0	I*4	Number of error (see file FEAT.MSG)
SUB0	C*6	Name of calling routine

Explanation

The error indicator IER0 is copied to IER in COMMON /ERRCTL/ and the error message routine OERR is invoked to display the corresponding error message on unit MERR and/or MTERM. Arguments for the message are passed via CPARAM in COMMON /CHAR/.

Bibliography

[1] Axelsson, O., Barker, V.A.: *Finite Element Solution of Boundary Value Problems*, Academic Press, 1984

[2] Blum, H., Harig, J., Müller, S.: *FEAT, Finite Element Analysis Tools, User Manual*, SFB 123 Report **554**, Univ. Heidelberg (1990)

[3] Schwarz, H.R.: *Methode der finiten Elemente*, Teubner, Stuttgart 1984

[4] Schwarz, H.R.: *FORTRAN-Programme zur Methode der finiten Elemente*, Teubner, Stuttgart 1984

A. List of FEAT2D subprograms

Routine	Filename	Short description	Page
AA03	AA03.F	Bilinear form, dbl./sgl. precision, triangle	38
AA07	AA07.F	Bilinear form, dbl./sgl. precision, triangle	38
AA09	AA09.F	Bilinear form, dbl./sgl. precision, triangle	38
AA13	AA13.F	Bilinear form (bound. int.), dbl./sgl. prec., triangle	38
AA17	AA17.F	Bilinear form (bound. int.), dbl./sgl. prec., triangle	38
AA19	AA19.F	Bilinear form (bound. int.), dbl./sgl. prec., triangle	38
AB03	AB03.F	Bilinear form, double and single precision, quadrilateral	38
AB07	AB07.F	Bilinear form, double and single prec., quadrilateral	38
AB09	AB09.F	Bilinear form, double and single prec.,quadrilateral	38
AB13	AB13.F	Bilinear form (bound. int.), dbl./sgl. prec., quadrilat.	38
AB17	AB17.F	Bilinear form (bound. int.), dbl./sgl. prec., quadrilat.	38
AB19	AB19.F	Bilinear form (bound. int.), dbl./sgl. prec., quadrilat.	38
AP3	AP3.F	Pointer vectors for bilinear forms, stor. techn. 3	39
AP7	AP7.F	Pointer vectors for bilinear forms, stor. techn. 7	39
AP9	AP9.F	Pointer vectors for bilinear forms, stor. techn. 9	39
CB1	CB1.F	1-dimensional quadrature formulas	46
CB2Q	CB2Q.F	2-dimensional cubature formulas, quadrilaterals	46
CB2T	CB2T.F	2-dimensional cubature formulas, triangles	46
E000	E000.F	Element, constant, triangle	48
E001	E001.F	Element, linear, triangle	48
E002	E002.F	Element, quadratic, triangle	48
E002A	E002.F	Auxiliary routine for E002	48
E003	E003.F	Element, cubic, Hermite, triangle	48
E003A	E003.F	Auxiliary routine for E003	48
E004	E004.F	Element, cubic, Lagrange, triangle	48
E004A	E004.F	Auxiliary routine for E004	48
E010	E010.F	Element, constant, quadrilateral	48
E011	E011.F	Element, bilinear, quadrilateral	48
E012	E012.F	Element, biquadratic, quadrilateral	48
E012A	E012.F	Auxiliary routine for E012	48
E013	E013.F	Element, reduced biquadratic (Serendipity), quadrilateral	48
E013A	E013.F	Auxiliary routine for E013	48
E014	E014.F	Element, bicubic, quadrilateral	48
E014A	E014.F	Auxiliary routine for E014	48
E020	E020.F	Element, linear nonconforming, triangle	48

Routine	Filename	Short description	Page
IA223	IA22.F	Jacobi-method, smooth., prec. (S/S), stor. techn. 3	53
IA227	IA22.F	Jacobi-method, smooth., prec. (S/S), stor. techn. 7	53
IA22A	IA22.F	Jacobi-method, smooth., prec. (S/S), stor. techn. A	53
IA233	IA23.F	Jacobi-method, smooth., prec. (S/D), stor. techn. 3	53
IA237	IA23.F	Jacobi-method, smooth., prec. (S/D), stor. techn. 7	53
IA23A	IA23.F	Jacobi-method, smooth., prec. (S/D), stor. techn. A	53
IB017	IB01.F	G-S-method, solver, prec. (D/D), stor. techn. 7	54
IB01A	IB01.F	G-S-method, solver, prec. (D/D), stor. techn. A	54
IB027	IB02.F	G-S-method, solver, prec. (S/S), stor. techn. 7	54
IB02A	IB02.F	G-S-method, solver, prec. (S/S), stor. techn. A	54
IB037	IB03.F	G-S-method, solver, prec. (S/D), stor. techn. 7	54
IB03A	IB03.F	G-S-method, solver, prec. (S/D), stor. techn. A	54
IB217	IB21.F	G-S-method, smooth., prec. (D/D), stor. techn. 7	54
IB21A	IB21.F	G-S-method, smooth., prec. (D/D), stor. techn. A	54
IB227	IB22.F	G-S-method, smooth., prec. (S/S), stor. techn. 7	54
IB22A	IB22.F	G-S-method, smooth., prec. (S/S), stor. techn. A	54
IB237	IB23.F	G-S-method, smooth., prec. (S/D), stor. techn. 7	54
IB23A	IB23.F	G-S-method, smooth., prec. (S/D), stor. techn. A	54
IC017	IC01.F	SOR-method, solver, prec. (D/D), stor. techn. 7	54
IC01A	IC01.F	SOR-method, solver, prec. (D/D), stor. techn. A	54
IC027	IC02.F	SOR-method, solver, prec. (S/S), stor. techn. 7	54
IC02A	IC02.F	SOR-method, solver, prec. (S/S), stor. techn. A	54
IC037	IC03.F	SOR-method, solver, prec. (S/D), stor. techn. 7	54
IC03A	IC03.F	SOR-method, solver, prec. (S/D), stor. techn. A	54
IC217	IC21.F	SOR-method, smooth., prec. (D/D), stor. techn. 7	54
IC21A	IC21.F	SOR-method, smooth., prec. (D/D), stor. techn. A	54
IC227	IC22.F	SOR-method, smooth., prec. (S/S), stor. techn. 7	54
IC22A	IC22.F	SOR-method, smooth., prec. (S/S), stor. techn. A	54
IC237	IC23.F	SOR-method, smooth., prec. (S/D), stor. techn. 7	54
IC23A	IC23.F	SOR-method, smooth., prec. (S/D), stor. techn. A	54
ID117	ID11.F	SSOR-method, precond., prec. (D/D), stor. techn. 7	55
ID118	ID11.F	SSOR-method, precond., prec. (D/D), stor. techn. 8	55
ID11A	ID11.F	SSOR-method, precond., prec. (D/D), stor. techn. A	55
ID127	ID12.F	SSOR-method, precond., prec. (S/S), stor. techn. 7	55
ID128	ID12.F	SSOR-method, precond., prec. (S/S), stor. techn. 8	55
ID12A	ID12.F	SSOR-method, precond., prec. (S/S), stor. techn. A	55
ID137	ID13.F	SSOR-method, precond., prec. (S/D), stor. techn. 7	55
ID138	ID13.F	SSOR-method, precond., prec. (S/D), stor. techn. 8	55
ID13A	ID13.F	SSOR-method, precond., prec. (S/D), stor. techn. A	55
ID217	ID21.F	SSOR-method, smooth., prec. (D/D), stor. techn. 7	55
ID218	ID21.F	SSOR-method, smooth., prec. (D/D), stor. techn. 8	55
ID21A	ID21.F	SSOR-method, smooth., prec. (D/D), stor. techn. A	55
ID227	ID22.F	SSOR-method, smooth., prec. (S/S), stor. techn. 7	55

Routine	Filename	Short description	Page
LSC2	LSC2.F	Scaling of vector, single precision	60
LSP1	LSP1.F	Scalar product of two vectors, double precision	60
LSP2	LSP2.F	Scalar product of two vectors, single precision	60
LTX13	LTX1.F	Transp. matrix-vector-mult., prec. (D/D), stor. techn. 3	60
LTX17	LTX1.F	Transp. matrix-vector-mult., prec. (D/D), stor. techn. 7	60
LTX19	LTX1.F	Transp. matrix-vector-mult., prec. (D/D), stor. techn. 9	60
LTX1A	LTX1.F	Transp. matrix-vector-mult., prec. (D/D), stor. techn. A	60
LTX23	LTX2.F	Transp. matrix-vector-mult., prec. (S/S), stor. techn. 3	60
LTX27	LTX2.F	Transp. matrix-vector-mult., prec. (S/S), stor. techn. 7	60
LTX29	LTX2.F	Transp. matrix-vector-mult., prec. (S/S), stor. techn. 9	60
LTX2A	LTX2.F	Transp. matrix-vector-mult., prec. (S/S), stor. techn. A	60
LTX33	LTX3.F	Transp. matrix-vector-mult., prec. (S/D), stor. techn. 3	60
LTX37	LTX3.F	Transp. matrix-vector-mult., prec. (S/D), stor. techn. 7	60
LTX39	LTX3.F	Transp. matrix-vector-mult., prec. (S/D), stor. techn. 9	60
LTX3A	LTX3.F	Transp. matrix-vector-mult., prec. (S/D), stor. techn. A	60
LVM1	LVM.F	Vector multiply and add, double precision	60
LVM2	LVM.F	Vector multiply and add, single precision	60
LVM3	LVM.F	Vector multiply and add, mixed precision	60
LWS13	LWS1.F	Weighted scalar product, prec. (D/D), stor. techn. 3	60
LWS14	LWS1.F	Weighted scalar product, prec. (D/D), stor. techn. 4	60
LWS17	LWS1.F	Weighted scalar product, prec. (D/D), stor. techn. 7	60
LWS18	LWS1.F	Weighted scalar product, prec. (D/D), stor. techn. 8	60
LWS19	LWS1.F	Weighted scalar product, prec. (D/D), stor. techn. 9	60
LWS1A	LWS1.F	Weighted scalar product, prec. (D/D), stor. techn. A	60
LWS23	LWS2.F	Weighted scalar product, prec. (S/S), stor. techn. 3	60
LWS24	LWS2.F	Weighted scalar product, prec. (S/S), stor. techn. 4	60
LWS27	LWS2.F	Weighted scalar product, prec. (S/S), stor. techn. 7	60
LWS28	LWS2.F	Weighted scalar product, prec. (S/S), stor. techn. 8	60
LWS29	LWS2.F	Weighted scalar product, prec. (S/S), stor. techn. 9	60
LWS2A	LWS2.F	Weighted scalar product, prec. (S/S), stor. techn. A	60
LWS33	LWS3.F	Weighted scalar product, prec. (S/D), stor. techn. 3	60
LWS34	LWS3.F	Weighted scalar product, prec. (S/D), stor. techn. 4	60
LWS37	LWS3.F	Weighted scalar product, prec. (S/D), stor. techn. 7	60
LWS38	LWS3.F	Weighted scalar product, prec. (S/D), stor. techn. 8	60
LWS39	LWS3.F	Weighted scalar product, prec. (S/D), stor. techn. 9	60
LWS3A	LWS3.F	Weighted scalar product, prec. (S/D), stor. techn. A	60
M010	MO1O.F	Multigrid solver, double precision	67
MPO01	MPO.F	Multigrid prolongation (version 0)	67
MPO11	MPO.F	Multigrid prolongation (version 0)	67
MP111	MP1.F	Multigrid prolongation (version 1)	67
MRO01	MRO.F	Multigrid restriction (version 0)	67
MRO11	MRO.F	Multigrid restriction (version 0)	67
MR111	MR1.F	Multigrid restriction (version 1)	67

Routine	Filename	Short description	Page
VA0	VA0.F	Linear form, dbl./sgl. prec., triangle	84
VA1	VA1.F	Linear form (bound. int.), dbl./sgl. prec., triangle	84
VB0	VB0.F	Linear form, dbl./sgl. prec., quadrilat.	84
VB1	VB1.F	Linear form (bound. int.), dbl./sgl. prec., quadrilat.	84
WERR	WERR.F	Basic error handling	87
XGOWSM	GOWSM.F	Write multigrid triang. in MOVIE.BYU format	50
XIC017	XIC01.F	Call IC017	54
XIC01A	XIC01.F	Call IC01A	54
XIC027	XIC02.F	Call IC027	54
XIC02A	XIC02.F	Call IC02A	54
XIE013	XIE01.F	CALL IE010, stor. techn. 3, double prec.	56
XIE014	XIE01.F	CALL IE010, stor. techn. 4, double prec.	56
XIE017	XIE01.F	CALL IE010, stor. techn. 7, double prec.	56
XIE018	XIE01.F	CALL IE010, stor. techn. 8, double prec.	56
XIE01A	XIE01.F	CALL IE010, stor. techn. A, double prec.	56
XIE023	XIE02.F	CALL IE020, stor. techn. 3, single prec.	56
XIE024	XIE02.F	CALL IE020, stor. techn. 4, double prec.	56
XIE027	XIE02.F	CALL IE020, stor. techn. 7, double prec.	56
XIE028	XIE02.F	CALL IE020, stor. techn. 8, double prec.	56
XIE02A	XIE02.F	CALL IE020, stor. techn. A, double prec.	56
XIE033	XIE03.F	CALL IE010, stor. techn. 3, sgl./dbl. prec.	56
XIE034	XIE03.F	CALL IE010, stor. techn. 4, sgl./dbl. prec.	56
XIE037	XIE03.F	CALL IE010, stor. techn. 7, sgl./dbl. prec.	56
XIE038	XIE03.F	CALL IE010, stor. techn. 8, sgl./dbl. prec.	56
XIE03A	XIE03.F	CALL IE010, stor. techn. A, sgl./dbl. prec.	56
XIE313	XIE31.F	CALL IE010, stor. techn. 3, double prec.	56
XIE314	XIE31.F	CALL IE010, stor. techn. 4, double prec.	56
XIE317	XIE31.F	CALL IE010, stor. techn. 7, double prec.	56
XIE318	XIE31.F	CALL IE010, stor. techn. 8, double prec.	56
XIE31A	XIE31.F	CALL IE010, stor. techn. A, double prec.	56
XIE323	XIE32.F	CALL IE020, stor. techn. 3, single prec.	56
XIE324	XIE32.F	CALL IE020, stor. techn. 4, double prec.	56
XIE327	XIE32.F	CALL IE020, stor. techn. 7, double prec.	56
XIE328	XIE32.F	CALL IE020, stor. techn. 8, double prec.	56
XIE32A	XIE32.F	CALL IE020, stor. techn. A, double prec.	56
XIE333	XIE33.F	CALL IE010, stor. techn. 3, sgl./dbl. prec.	56
XIE334	XIE33.F	CALL IE010, stor. techn. 4, sgl./dbl. prec.	56
XIE337	XIE33.F	CALL IE010, stor. techn. 7, sgl./dbl. prec.	56
XIE338	XIE33.F	CALL IE010, stor. techn. 8, sgl./dbl. prec.	56
XIE33A	XIE33.F	CALL IE010, stor. techn. A, sgl./dbl. prec.	56
XM010	XM010.F	Call of M010	67
XM017	XM017.F	Call of M017	67
XMAA03	XMAA03.F	Successive call of XAA03	67

B. List of COMMON blocks

```
      IMPLICIT DOUBLE PRECISION (A,C-H,O-U,W-Z),LOGICAL(B)
      CHARACTER SUB*6,FMT*15,CPARAM*120
C
      PARAMETER (NNARR=299,NNLEV=9)
      PARAMETER (NNVE=4,NNBAS=21,NNCUBP=36,NNAB=21,NNDER=6)
C
      COMMON          NWORK,IWORK,IWMAX,L(NNARR),DWORK(1)
      COMMON /OUTPUT/ M,MT,MKEYB,MTERM,MERR,MPROT,MSYS,MTRC,IRECL8
      COMMON /ERRCTL/ IER,ICHECK
      COMMON /CHAR/   SUB,FMT(3),CPARAM
      COMMON /TRIAA/  LCORVG,LCORMG,LVERT,LMID,LADJ,LVEL,LMEL,LNPR,LMM,
     *                LVBD,LEBD,LBCT,LVBDP,LMBDP
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /MACROD/ NMAEL,NMAVT,NMAEDG,NMAVE,NMAVEL,NMABCT,NMAVBD
      COMMON /MACROA/ LMACVG,LMACMG,LMAVT,LMAMID,LMAADJ,LMAVEL,LMAMEL,
     *                LMANPR,LMAMM,LMAVBD,LMAEBD,LMABCT,LMAVBP,LMAMBP,
     *                LMAVE
      COMMON /MGTRD/  KNEL(NNLEV),KNVT(NNLEV),KNMT(NNLEV),
     *                KNVEL(NNLEV),KNVBD(NNLEV)
      COMMON /MGTRA/  KLCVG(NNLEV),KLCMG(NNLEV),KLVERT(NNLEV),
     *                KLMID(NNLEV),KLADJ(NNLEV),KLVEL(NNLEV),
     *                KLMEL(NNLEV),KLNPR(NNLEV),KLMM(NNLEV),
     *                KLVBD(NNLEV),KLEBD(NNLEV),KLBCT(NNLEV),
     *                KLVBDP(NNLEV),KLMBDP(NNLEV)
      COMMON /ELEM/   DX(NNVE),DY(NNVE),DJAC(2,2),DETJ,
     *                DBAS(NNBAS,NNDER),BDER(NNDER),KVE(NNVE),IEL
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
      COMMON /COAUX1/ KDFG(NNBAS),KDFL(NNBAS),IDFL
      COMMON /COAUX2/ DBAS1(NNBAS,NNDER,3),KDFG1(NNBAS,3),
     *                KDFL1(NNBAS,3),IDFL1(3)
      COMMON /TABLE/  KTYPE(NNARR),KLEN(NNARR),KLEN8(NNARR),IFLAG
```

C. List of error message file FEAT.MSG

```
  1 1 3  2'ARRAY',A7,' ALLOCATED AS #',I3,' , LENGTH IS',I8
  2 1 3  2'ARRAY',A7,' ALLOCATED AS #',I3,' , LENGTH IS',I8/16X,'TOTAL
          FREE PART OF DWORK IS USED'
  3 1 3  3'ARRAY',A7,' DELETED ,    #',I3,' RELEASED'
  4 1 3  2'ARRAY',A7,' (#',I3,') COMPRESSED, LENGTH IS',I8
  5 1 3  6'ARRAY',A7,' (#',I3,') SUCCESSFULLY COPIED ONTO ARRAY',A7,
          ' (#',I3,')'
  6 1 3  2'TYPE OF ARRAY',A7,' (#',I3,') CHANGED TO',I3
  7 1 2  2'ARRAY',A7,' (#',I3,') SUCCESSFULLY READ FROM UNIT',I3
  8 1 2  2'ARRAY',A7,' (#',I3,') SUCCESSFULLY SAVED ONTO UNIT',I3
  9 1 0  1'PLEASE ENTER UNIT NUMBER AND FILENAME'
 20 2 2  8'LENGTH OF KCOL INCREASED, NEW LENGTH IS',I8
 21 1 2 11'ARRAYS COMPRESSED, OLD LENGTH',I8/35X,'NEW LENGTH',I8
 30 1 0  5'UNIT NUMBER',I3,' INVALID'/       16X,'ERROR OCCURED WHILE
          PROCESSING ARRAY',A7
 31 1 0  9'FILE ',A??,' COULD NOT BE OPENED AS UNIT',I3
 32 1 0 10'UNIT',I3,' ALREADY OPENED, FILENAME IS ',A??
 33 1 0  8'UNIT',I3,' ALREADY OPENED AS SCRATCH FILE'
 34 1 0  1'CORRECTION OF UNIT NUMBER AND FILENAME ?  (0/1)'
 35 1 0  1'CLOSE PREVIOUSLY USED FILE ?  (0/1)'
 36 2 3  1'MORE CHARACTERS USED FOR FILENAME THAN PROVIDED BY CFILE'
 51 2 2  2'WARNING DURING REVISION OF',A7,' , WRONG INPUT PARAMETER
          LNR=',I3,' IFLAG=',I3,' IS USED'
 52 2 2  4'WARNING DURING REVISION OF',A7,' , WRONG INPUT PARAMETER
          LNR=  0'
 53 2 2  3'WARNING DURING REVISION OF',A7,' (#',I3,') , SAME TYPE OF
          SOURCE AND TARGET ARRAY'
 54 2 3  8'WARNING WHILE COMPRESSING ARRAYS'/16X,'ROW',I8,' HAS ONLY
          ZERO ENTRIES'
 70 2 2  1'WARNING  ZERO RIGHT HAND SIDE'
 71 1 0  1'CONVERGENCE FAILED'
 72 1 1 13/5X,'ITERATIONS',13X,I10/5X,'NORM OF RESIDUAL',7X,D12.3/5X,
              '!!RES!!/!!INITIAL RES!!',D12.3
 73 2 2 12'  ITERATION',I6,5X,'  !!RES!! =',D12.3
 74 2 2 12'  ITERATION',I6,5X,'  !!CORR!! =',D12.3
 75 1 1 12/5X,'ITERATIONS',13X,I10/5X,'MAXIMUM OF CORRECTION',D12.3
 76 1 1 14/5X,'RATE OF CONVERGENCE',4X,D12.3
100 0 1  4'IWORK=NWORK'/                 16X,'ERROR OCCURED WHILE
          PROCESSING ARRAY',A7
```

```
101 0 1  5'ITYPE=',I3,' INVALID'/        16X,'ERROR OCCURED WHILE
          PROCESSING ARRAY',A7
102 0 1  4'NNARR EXCEEDED'/              16X,'ERROR OCCURED WHILE
          PROCESSING ARRAY',A7
103 0 1  5'NWORK MUST BE INCREASED BY',I8/16X,'ERROR OCCURED WHILE
          PROCESSING ARRAY',A7
104 0 1  5'LNR=',I3,' INVALID'/          16X,'ERROR OCCURED WHILE
          PROCESSING ARRAY',A7
105 0 1  4'WRONG VALUE OF ILONG'/        16X,'ERROR OCCURED WHILE
          PROCESSING ARRAY',A7
106 0 1  8'ITYPE=',I3,' INVALID'
107 0 1  6'DATA TYPES OF SOURCE ARRAY',A7,' (#',I3,') AND'/36X,'TARGET
          ARRAY',A7,' (#',I3,') DO NOT MATCH'
108 0 1  6'SOURCE ARRAY',A7,' (#',I3,') LARGER THAN TARGET ARRAY',A7,
          ' (#',I3,')'
109 0 1  1'FILENAME CFILE CONTAINS MORE THAN 60 CHARACTERS'
110 0 1  8'WHILE READING FROM UNIT',I3
111 0 1  8'WHILE WRITING ONTO UNIT',I3
112 0 1  9'FILE ',A??,' COULD NOT BE OPENED AS UNIT',I3
113 0 1  3'WRONG DATA TYPE OF ARRAY',A7/      16X,'ERROR OCCURED WHILE
          READING FROM UNIT',I3
114 0 1  3'WRONG DATA TYPE OF ARRAY',A7/      16X,'ERROR OCCURED WHILE
          PROCESSING BLOCK',I3
115 0 1  3'ARRAY',A7,' TOO SHORT'/            16X,'ERROR OCCURED WHILE
          PROCESSING BLOCK',I3
116 0 1  8'WRONG VALUE IN ARRAY KAB SPECIFIED'/16X,'ERROR OCCURED WHILE
          PROCESSING BLOCK',I3
117 0 1  8'WRONG VALUE IN ARRAY KB SPECIFIED'/ 16X,'ERROR OCCURED WHILE
          PROCESSING BLOCK',I3
118 0 1  8'NOT ENOUGH SPACE FOR KCOL'/         16X,'ERROR OCCURED WHILE
          PROCESSING ELEMENT',I6
119 0 1  8'ICUB=',I3,' INVALID'
120 0 1  8'IELTYP=',I3,' INVALID'
121 0 1  1'AT LEAST ONE OF THE VECTORS TOO SMALL OR INVALID NUMBER'
130 0 1  1'VERTICES NOT ORDERED IN COUNTERCLOCKWISE SENSE'
131 0 1  1'DESIRED DERIVATIVE NOT AVAILABLE'
132 0 1  1'ELEMENT HAS VANISHING AREA'
140 0 1  1'FIRST PARAMETER LARGER THAN SECOND'
141 0 1  1'AT LEAST ONE PARAMETER OUTSIDE OF VALID RANGE'
150 0 1  1'VALUE OF NVE INVALID'
151 0 1  1'ONE OF THE VALUES NEL, NVT, NBCT LESS OR EQUAL 0'
152 0 1 11'KNPR(',I6,') CONTAINS INVALID VALUE',I3
153 0 1 12'DCORVG(1,',I6,') CONTAINS INVALID PARAMETER',D12.5
154 0 1  8'TWO VERTICES WITH THE SAME NUMBER'/     16X,'ERROR OCCURED
          IN ELEMENT',I6
155 0 1  8'INVALID ZERO ENTRY IN KVERT'/           16X,'ERROR OCCURED
          IN ELEMENT',I6
156 0 1  8'ENTRY IN KVERT LARGER THAN NVT'/        16X,'ERROR OCCURED
          IN ELEMENT',I6
```

```
157 0 1  8'LAST ENTRY IN KVERT NOT ZERO BUT NVE=3'/16X,'ERROR OCCURED
          IN ELEMENT',I6
158 0 1  8'VERTICES NOT ORDERED IN COUNTERCLOCKWISE SENSE'/16X,'ERROR
          OCCURED IN ELEMENT',I6
159 0 1  8'ELEMENT HAS VANISHING AREA'/           16X,'ERROR OCCURED
          IN ELEMENT',I6
160 0 1  8'LAST ENTRY IN KADJ NOT ZERO BUT NVE=3'/ 16X,'ERROR OCCURED
          IN ELEMENT',I6
161 0 1  8'ENTRY IN KADJ LARGER THAN NEL'/        16X,'ERROR OCCURED
          IN ELEMENT',I6
162 0 1  8'NO JOINING EDGE BETWEEN TWO NEIGHBORED ELEMENTS'/16X,'ERROR
          OCCURED IN ELEMENT',I6
163 0 1  8'BOUNDARY EDGE FORMED BY VERTICES ON DIFFERENT BOUNDARY
          COMPONENTS'/16X,'ERROR OCCURED IN ELEMENT',I6
164 0 1  1'CANNOT PROCEED - PARAMETERS OF BOUNDARY VERTICES NOT AVAILABLE'
170 0 1  1'WRONG DATA TYPE OF AT LEAST ONE ARRAY'
```

D. Sample Programs

```
      PROGRAM SAMPLE1
C
C     Sample program
C     Demonstration of FEAT
C
C     Problem  -2u   - u   + u = f
C                 xx    yy
C
C     Exact solution  u = 16*x*(1-x)*y*(1-y)
C     Homogeneneous Dirichlet boundary conditions implemented
C     according to exact solution
C
C     linear triangle element
C
C     Domain: Unit square
C
C     REMARK: exact solution equals zero on unit square boundary which is
C     used to implement above mentioned boundary conditions in
C     subroutine BDRY
C
C
C *** Standard declarations and parameter settings
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      PARAMETER (NNARR=299,NNAB=21,NNWORK=300000)
C
C *** NBLOCA=1 means that the stiffness matrix made up of only one block,
C     analogously NBLOCF stands for the right hand side.
      PARAMETER (NBLOCA=1,NBLOCF=1)
C
C *** Standard declaration for CHARACTER quantities in /CHAR/
      CHARACTER SUB*6,FMT*15,CPARAM*120

C *** CFILE needed as filename for input in XORSC
C     ARRDA and ARRDF are used only for messages
      CHARACTER CFILE*15,ARRDA*6,ARRDF*6
C
C *** Standard dimensioning for workspace concept
      DIMENSION VWORK(1),KWORK(1)
C
C *** KABA  - structure of the stiffness matrix
C *** KF    -  structure of the right hand side
C *** KABAN - number of terms in the integrand for the stiffness matrix
C *** KFN   - number of terms in the integrand for the right hand side
      DIMENSION KABA(2,NNAB,NBLOCA),KF(NNAB,NBLOCF)
      DIMENSION KABAN(NBLOCA),KFN(NBLOCF)
C
C *** BCONA, BCONF - .TRUE. if the corresponding coefficients are constant
      DIMENSION BCONA(NBLOCA),BCONF(NBLOCF)
C
```

```
C *** LA, LF - numbers for matrix hand right hand side
      DIMENSION LA(NBLOCA),LF(NBLOCF)
C
C *** ARRDA, ARRDF - Names of matrices and right hand sides (for messages only)
      DIMENSION ARRDA(NBLOCA),ARRDF(NBLOCF)
      DIMENSION BSNGLA(NBLOCA),BSNGLF(NBLOCF)
C
C *** Standard COMMON blocks
      COMMON          NWORK,IWORK,IWMAX,L(NNARR),DWORK(NNWORK)
      COMMON /ERRCTL/ IER,ICHECK
      COMMON /CHAR/   SUB,FMT(3),CPARAM
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /TRIAA/  LCORVG,LCORMG,LVERT,LMID,LADJ,LVEL,LMEL,LNPR,LMM,
     *                LVBD,LEBD,LBCT,LVBDP,LMBDP
      COMMON /OUTPUT/ M,MT,MKEYB,MTERM,MERR,MPROT,MSYS,MTRC,IRECL8
C
C *** EQUIVALENCE statement needed for DWORK concept
      EQUIVALENCE (DWORK(1),VWORK(1),KWORK(1))
C
C *** Parametrization of the domain
      EXTERNAL PARXQ,PARYQ,TMAXQ
C *** Coefficient of stiffness matrix, right hand side, exact solution
      EXTERNAL COEFFA,RHS,UE,UEX,UEY
C *** Control of refinement - here, regular refinement is used
      EXTERNAL S2DI0,S2DB0
C *** E001 - linear element on triangular mesh ; dummy program I000
C *** needed in call of XIE017
      EXTERNAL E001,I000
C
C *** Number of terms in stiffness matrix and right hand side
      DATA KABAN/3/,KFN/1/
C *** Coefficients are constant for the matrix, but not for the right hand side
      DATA BCONA/.TRUE./,BCONF/.FALSE./
C *** Names of matrix and vector (for messages only)
      DATA ARRDA/'DA    '/,ARRDF/'DF    '/
      DATA BSNGLA/.FALSE./,BSNGLF/.FALSE./
C
C
C
C *** Initialization
C *** Default names are used for output devices
C
      CALL ZINIT(NNWORK,'feat.msg',' ',' ',' ',' ')
C
C
C *** Structure of bilinear and linear forms
C
C
      KABA(1,1,1)=2
      KABA(2,1,1)=2
      KABA(1,2,1)=3
      KABA(2,2,1)=3
      KABA(1,3,1)=1
      KABA(2,3,1)=1
C
      KF(1,1)=1
C
C *** Perform elementary checks only
      ICHECK=1
```

```
      CALL ZTIME(TIME1)
C
C *** Open data file for input
      OPEN (88,FILE='CTEST1.DAT')
C
C *** Read in message levels for files and for the screen
      READ(88,*) M
      WRITE(MTERM,*) 'M = ',M
      READ(88,*) MT
      WRITE(MTERM,*) 'MT = ',MT
C
C *** Read in coarse grid from file 'triat1', opened as unit 55
      CFILE='triat1'
      MUNITT=55
      CALL XORSC(MUNITT,CFILE)
      IF (IER.NE.0) GOTO 99998
C
C *** Read in desired number of refinements
      READ(88,*) NFINE
      WRITE(MTERM,*) 'NFINE = ',NFINE
C
C *** Construct a regular subdivision of the given coarse grid
C *** IMID=0  - Numbers of midpoints are  not needed for linear elements
C *** IADJ=0  - Numbers of neighbouring elements can be neglected
C *** IDISP=1 - Release all free space
C *** IBDP=2  - Keep information on parameter values for boundary vertices
C ***           Store numbers of boundary vertices on KVBD
C *** Routines for controlling the refinement and parametrization are passed as
C *** Externals
      IMID=0
      IADJ=0
      IVEL=0
      IDISP=1
      IBDP=2
      CALL XSAOX(NFINE,IMID,IADJ,IVEL,IDISP,IBDP,
     *           S2DI0,S2DB0,PARXQ,PARYQ,TMAXQ)
      IF (IER.NE.0) GOTO 99998
C
C *** Calculate the pointer vectors for the stiffness matrix
C *** Storage technique 7 - symmetry of the matrix is neglected
C *** which facilitates the implementation boundary conditions
      ISYMM=0
      CALL XAP7(LCOLA,LLDA,NA,NEQ,E001,ISYMM)
      IF (IER.NE.0) GOTO 99998
C
C *** Allocate solution vector DU in the correct length NEQ, determined by XAP7
      CALL ZNEW(NEQ,1,LU,'DU    ')
      IF (IER.NE.0) GOTO 99998
C
C
C *** Calculation of the stiffness matrix
C *** Number is returned in LA(1)
C *** Symmetric elements are not calculated twice (ISYMM=2)
C *** Number of cubature formula read in as a parameter
      LA(1)=0
      READ(88,*) ICUB
      WRITE(MTERM,*) 'ICUB FOR MATRIX = ',ICUB
      ICLR=1
      CALL XAA07(LA,LCOLA,LLDA,NA,NEQ,NBLOCA,ICLR,E001,
     *           COEFFA,BCONT,KABA,KABAN,ICUB,ISYMM,ARRDA,BSNGLA)
```

```
      IF (IER.NE.0) GOTO 99998
C
C *** Calculation of the right hand side vector
C *** Number is returned in LF(1)
C *** Number of cubature formula read in as a parameter
      LF(1)=0
      READ(88,*) ICUB
      WRITE(MTERM,*) 'ICUB RIGHT HAND SIDE = ',ICUB

      CALL XVA0(LF,NEQ,NBLOCF,ICLR,E001,
     *          RHS,BCONF,KF,KFN,ICUB,ARRDF,BSNGLF)
      IF (IER.NE.0) GOTO 99998
C
C *** User provided program for implementation of boundary conditions
      CALL BDRY(DWORK(L(LA(1))),KWORK(L(LCOLA)),KWORK(L(LLDA)),
     *          KWORK(L(LVBD)),KWORK(L(LBCT)),DWORK(L(LU)),
     *          DWORK(L(LF(1))),NEQ,DWORK(L(LCORVG)))
C
C *** Elements with modulus smaller than 1D-15 are considered as zero and are
C *** deleted from matrix LA(1), i.e. DA. The pointer vector LCOLA
C *** i.e. the vector KCOLA is changed correspondingly.
      CALL XRC17 (LA,LCOLA,LLDA,NA,NEQ,NBLOCA,1D-15,1,
     *           'DA    ','KCOLA ')
      IF (IER.NE.0) GOTO 99998
C
C *** Parameters for cg solution algorithm are read in from data file
C *** NIT maximum number of iterations, EPS desired accuracy,
C *** OMEGA preconditioning parameter
      READ(88,*) NIT
      WRITE(MTERM,*) 'NIT CG ',NIT
      READ(88,*) EPS
      WRITE(MTERM,*) 'EPS CG ',EPS
      READ(88,*) OMEGA
      WRITE(MTERM,*) 'OMEGA CG ',OMEGA
C
C *** The total CPU time for the preparation of the linear system is displayed.
      CALL ZTIME(TIME2)
      WRITE (MPROT,*) 'TIME PREPARATION ',TIME2-TIME1
C
C *** RBNORM is set to the L2-norm of the right hand side
      CALL XLL21(LF(1),NEQ,RBNORM)
C *** EPS is normalized by the norm of the right hand side (relative accuracy)
      EPS=EPS*RBNORM
C *** the preconditioned conjugate gradient method is used for solution of the
C *** linear system
C *** OMEGA<0 - no prec. , OMEGA=0 - scaling by diagonal
C *** 0 < OMEGA < 2 - SSOR preconditioning
C *** For values OMEGA>2, I000 must be replaced by the prec. subprogram
C
      CALL ZTIME(TIME1)
      CALL XIE017(LA(1),LCOLA,LLDA,LU,LF(1),NEQ,NIT,ITE,EPS,OMEGA,I000)
C
C *** The cpu time for the solution is displayed
      CALL ZTIME(TIME2)
      WRITE (MPROT,*) 'TIME SOLVER ',TIME2-TIME1
C
C *** A user provided subprogram is called to calculate the maximum error
C *** The maximum is calculated over all cubature points of formula ICUB
      CALL ZTIME(TIME1)
      READ(88,*) ICUBM
```

```
      WRITE(MTERM,*) 'ICUB ELIT ',ICUBM
      CALL ELIT(DWORK(L(LU)),KWORK(L(LVERT)),
     *          KWORK(L(LMID)),DWORK(L(LCORVG)),E001,ICUB,UE)
      IF (IER.NE.0) GOTO 99998
C
C *** A user provided subprogram is called to calculate the
C *** L2- and H1-error
      READ(88,*) ICUB2
      WRITE(MTERM,*) 'ICUB ELPT ',ICUB2
      CALL ELPT(DWORK(L(LU)),KWORK(L(LVERT)),
     *          KWORK(L(LMID)),DWORK(L(LCORVG)),E001,ICUB,UE,UEX,UEY)
      IF (IER.NE.0) GOTO 99998
C
C *** The solution vector is written to file CFILE (unit IFILE)
C *** XOWA uses the standard FORMAT contained in FMT(1), for DOUBLE PRECISION
C *** vectors
      WRITE(MTERM,*) 'OUTPUT DATA ONTO FILE ?'
      READ(88,*) IFILE
      WRITE(MTERM,*) IFILE
      WRITE(MTERM,*) 'OUTPUT FILENAME ?'
      READ(88,*) CFILE
      WRITE(MTERM,*) CFILE
      IF (IFILE.NE.0) CALL XOWA(LU,'DU    ',IFILE,CFILE,1)
C
C *** The CPU time for the calculation of the error is displayed
      CALL ZTIME(TIME2)
      WRITE(MTERM,*) 'TIME POSTPROCESSING ',TIME2-TIME1
C
C *** Write triangulation and solution vector (vertices only) to the disk
C *** in MOVIE.BYU format
      CALL ZDISP(NVT,LU,'DU    ')
      IF (IER.NE.0) GOTO 99998
      CALL XGOWFM(LU,69,'DU.TEST1.OUT')
      IF (IER.NE.0) GOTO 99998
      CALL XGOWSM(70,'TRIA.TEST1.OUT')
      IF (IER.NE.0) GOTO 99998
C *** NWORK contains the total number of DOUBLE PRECISION elements of DWORK
C *** IWMAX is the maximum number of elements of DWORK used in the current
C *** program
      WRITE(MTERM,*) 'NWORK ',NWORK
      WRITE(MTERM,*) 'IWMAX ',IWMAX
C
      GOTO 99999
99998 WRITE(MTERM,*) 'IER', IER
      WRITE(MTERM,*) 'IN SUBROUTINE ',SUB
99999 END
C
C
C
C *** Implementation of Dirichlet boundary conditions
C *** for elements with degree of freedom (d.o.f.) at vertices
      SUBROUTINE BDRY(DA,KCOL,KLD,KVBD,KBCT,DX,DB,NEQ,DCORVG)
C
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      DIMENSION DA(*),KCOL(*),KLD(*),KVBD(*),DX(*),DB(*),KBCT(*)
      DIMENSION DCORVG(2,*)
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      SAVE /TRIAD/
C
C *** UE is the exact solution
```

```
      XBD(X,Y)=UE(X,Y)
C
C *** First loop over all boundary vertices
C *** The numbers of the vertices are contained in KVBD (set by XSVEB)
      DO 1 IVT=KBCT(1),KBCT(2)-1
      IEQ=KVBD(IVT)
C *** The diagonal element is set to 1
      DA(KLD(IEQ))=1D0
C *** DCORVG(1,IEQ) contains the parameter of the boundary point
      X=DCORVG(1,IEQ)
      Y=DCORVG(2,IEQ)
      DX(IEQ)=XBD(X,Y)
      DB(IEQ)=XBD(X,Y)
C *** The nondiagonal elements of the matrix are set to zero
      DO 2 ICOL=KLD(IEQ)+1,KLD(IEQ+1)-1
2     DA(ICOL)=0D0
C
1     CONTINUE
      END
C
C
C
C *** The following three subprograms describe the parametrization of the
C *** boundary of the unit square
C
      DOUBLE PRECISION FUNCTION PARXQ(T,IBCT)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      IF (T.GE.3.D0) GOTO 50
      IF (T.GE.2.D0) GOTO 60
      IF (T.GE.1.D0) GOTO 70
      PARXQ=T
      GOTO 99999
70    PARXQ=1D0
      GOTO 99999
60    PARXQ=3D0-T
      GOTO 99999
50    PARXQ=0D0
99999 END
C
C
C
      DOUBLE PRECISION FUNCTION PARYQ(T,IBCT)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      IF (T.GE.3.D0) GOTO 80
      IF (T.GE.2.D0) GOTO 90
      IF (T.GE.1.D0) GOTO 100
      PARYQ=0D0
      GOTO 99999
100   PARYQ=T-1D0
      GOTO 99999
90    PARYQ=1D0
      GOTO 99999
80    PARYQ=4D0-T
99999 END
C
C
C
      DOUBLE PRECISION FUNCTION TMAXQ(IBCT)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      TMAXQ=4D0
```

```
      END
C
C
C *** Coefficient function for the stiffness matrix
C
      DOUBLE PRECISION FUNCTION COEFFA(X,Y,IA,IB,IDA,BFIRST)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      IF (IA.EQ.2) THEN
C *** Coefficient for the second x-derivative
       COEFFA=2D0
      ELSE
C *** Coefficient for the second y-derivative and the absolute term
       COEFFA=1D0
      ENDIF
C
      END
C
C
C *** Right hand side yielding the exact solution UE
C
      DOUBLE PRECISION FUNCTION RHS(X,Y,IA,IDA,BFIRST)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      RHS=64D0*Y*(1D0-Y)+32D0*X*(1D0-X)+16D0*X*(1D0-X)*Y*(1D0-Y)
      END
C
C
C *** Exact solution
C
      DOUBLE PRECISION FUNCTION UE(X,Y)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      UE=16D0*X*(1D0-X)*Y*(1D0-Y)
      END
C
C
C *** Exact UEX
C
      DOUBLE PRECISION FUNCTION UEX(X,Y)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      UEX=16D0*(1D0-2D0*X)*Y*(1D0-Y)
      END
C
C
C *** Exact UEY
C
      DOUBLE PRECISION FUNCTION UEY(X,Y)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      UEY=16D0*X*(1D0-X)*(1D0-2D0*Y)
      END
C
C
C *** Subprogram to determine the maximum error at the cubature points of
C *** cubature formula ICUB
C *** The procedure for the evaluation of the discrete function is copied from
C *** the FEAT subprogram AA07
C
      SUBROUTINE ELIT(DU,KVERT,KMID,DCORVG,ELE,ICUB,U)
C
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      CHARACTER FMT*15,SUB*6,CPARAM*120
      PARAMETER (NNBAS=21,NNDER=6,NNCUBP=36,NNVE=4)
```

```
      DIMENSION DU(*)
      DIMENSION KVERT(NNVE,1),KMID(NNVE,1),DCORVG(2,1)
      DIMENSION KDFG(NNBAS),KDFL(NNBAS)
      COMMON /ERRCTL/ IER,ICHECK
      COMMON /ELEM/   DX(NNVE),DY(NNVE),DJAC(2,2),DETJ,
     1                DBAS(NNBAS,NNDER),BDER(NNDER),KVE(NNVE),IEL
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /TRIAA/  LCORVG,LCORMG,LVERT,LMID,LADJ,LVEL,LMEL,LNPR,LMM,
     *                LVBD,LEBD,LBCT,LVBDP,LMBDP
      COMMON /CHAR/   SUB,FMT(3),CPARAM
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
      COMMON /OUTPUT/ M,MT,MKEYB,MTERM,MERR,MPROT,MSYS,MTRC,IRECL8
      COMMON /COAUX1/ KDFG,KDFL,IDFL
      SAVE
C
      IER=0
C
      IELTYP=-1
      CALL ELE(0D0,0D0,0D0,IELTYP)
      IF (IER.NE.0) GOTO 99999
      IDFL=NDFL(IELTYP)
      IF (IER.LT.0) GOTO 99999
      CALL CB2T(ICUB)
      IF (IER.NE.0) GOTO 99999
      DO 1 IDER=1,NNDER
1     BDER(IDER)=.FALSE.
      BDER(1)=.TRUE.
C
      ERRM=0D0
      DNM=0D0
      CALL ELE(0D0,0D0,0D0,-2)
      IF (IER.LT.0) GOTO 99999
C
      DO 100 IEL=1,NEL
C
      CALL NDFGL(IEL,1,IELTYP,KVERT,KMID,KDFG,KDFL)
      IF (IER.LT.0) GOTO 99999
C
      DO 110 IVE=1,NVE
      JP=KVERT(IVE,IEL)
      KVE(IVE)=JP
      DX(IVE)=DCORVG(1,JP)
      DY(IVE)=DCORVG(2,JP)
110   CONTINUE
C
      DJAC(1,1)=DX(2)-DX(1)
      DJAC(1,2)=DX(3)-DX(1)
      DJAC(2,1)=DY(2)-DY(1)
      DJAC(2,2)=DY(3)-DY(1)
      DETJ=DJAC(1,1)*DJAC(2,2)-DJAC(1,2)*DJAC(2,1)
C
      DO 200 ICUBP=1,NCUBP
C
      XI1=DXI(ICUBP,1)
      XI2=DXI(ICUBP,2)
      XI3=DXI(ICUBP,3)
      OM=DOMEGA(ICUBP)*DETJ
C
      CALL ELE(XI1,XI2,XI3,-3)
      IF (IER.LT.0) GOTO 99999
```

```
C
      XX=DX(1)+XI2*DJAC(1,1)+XI3*DJAC(1,2)
      YY=DY(1)+XI2*DJAC(2,1)+XI3*DJAC(2,2)
C
      UH =0D0
C
      DO 210 JDFL=1,IDFL
      IEQ=KDFG(JDFL)
      ILO=KDFL(JDFL)
      UH=UH+DU(IEQ)*DBAS(ILO,1)
210   CONTINUE
C
C
      ELML=ABS(U(XX,YY)-UH)
C
      ERRM=MAX(ERRM,ELML)
      DNM= MAX(DNM,ABS(U(XX,YY)))
C
200   CONTINUE
100   CONTINUE
C
      WRITE(MTERM,*)
      WRITE(MTERM,*) '* ELIT * DISCRETE SOLUTION   ',IELTYP
      WRITE(MTERM,*) '* ELIT * EVALUATION PTS AS FORMULA ',ICUB
      WRITE(MTERM,*) '* ELIT * MAX ERR FUNCTION-VALUES ',ERRM
      IF (M.GE.1)
     * WRITE(MTERM,*) '* ELIT * MAXIMUM FUNCTION-VALUES ',DNM
      IF (MTERM.NE.MPROT) THEN
       WRITE(MPROT,*)
       WRITE(MPROT,*) '* ELIT * DISCRETE SOLUTION   ',IELTYP
       WRITE(MPROT,*) '* ELIT * EVALUATION PTS AS FORMULA ',ICUB
       WRITE(MPROT,*) '* ELIT * MAX ERR FUNCTION-VALUES ',ERRM
       IF (M.GE.1)
     *  WRITE(MPROT,*) '* ELIT * MAXIMUM FUNCTION-VALUES ',DNM
      ENDIF
C
99999 END
C
C
      SUBROUTINE ELPT(DU,KVERT,KMID,DCORVG,ELE,ICUB,U,UX,UY)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      PARAMETER (NNBAS=21,NNDER=6,NNCUBP=36,NNVE=4)
      CHARACTER FMT*15,SUB*6,CPARAM*120
      DIMENSION DU(*),KVERT(NNVE,*),KMID(NNVE,*),DCORVG(2,*)
      DIMENSION KDFG(NNBAS),KDFL(NNBAS)
      COMMON /OUTPUT/ M,MT,MKEYB,MTERM,MERR,MPROT,MSYS,MTRC,IRECL8
      COMMON /ERRCTL/ IER,ICHECK
      COMMON /CHAR/   SUB,FMT(3),CPARAM
      COMMON /ELEM/   DX(NNVE),DY(NNVE),DJAC(2,2),DETJ,
     *                DBAS(NNBAS,NNDER),BDER(NNDER),KVE(NNVE),IEL
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
      COMMON /COAUX1/ KDFG,KDFL,IDFL
      SAVE
C
      SUB='ELPT'
      IER=0
C
      IELTYP=-1
      CALL ELE(0D0,0D0,0D0,IELTYP)
```

```
      IF (IER.NE.0) GOTO 99999
      IDFL=NDFL(IELTYP)
      IF (IER.LT.0) GOTO 99999
      CALL CB2T(ICUB)
      IF (IER.NE.0) GOTO 99999
      DO 1 IDER=1,NNDER
1     BDER(IDER)=.FALSE.
      BDER(1)=.TRUE.
      BDER(2)=.TRUE.
      BDER(3)=.TRUE.
C
      ERRL2=0D0
      ERRH1=0D0
      DNL2=0D0
      DNH1=0D0
      CALL ELE(0D0,0D0,0D0,-2)
C
      DO 100 IEL=1,NEL
C
      CALL NDFGL(IEL,1,IELTYP,KVERT,KMID,KDFG,KDFL)
      IF (IER.LT.0) GOTO 99999
C
      DO 110 IVE=1,NVE
      JP=KVERT(IVE,IEL)
      KVE(IVE)=JP
      DX(IVE)=DCORVG(1,JP)
      DY(IVE)=DCORVG(2,JP)
110   CONTINUE
C
      DJAC(1,1)=DX(2)-DX(1)
      DJAC(1,2)=DX(3)-DX(1)
      DJAC(2,1)=DY(2)-DY(1)
      DJAC(2,2)=DY(3)-DY(1)
      DETJ=DJAC(1,1)*DJAC(2,2)-DJAC(1,2)*DJAC(2,1)
C
      DO 200 ICUBP=1,NCUBP
C
      XI1=DXI(ICUBP,1)
      XI2=DXI(ICUBP,2)
      XI3=DXI(ICUBP,3)
      OM=DOMEGA(ICUBP)*DETJ
C
      CALL ELE(XI1,XI2,XI3,-3)
      IF (IER.LT.0) GOTO 99999
C
      XX=DX(1)+XI2*DJAC(1,1)+XI3*DJAC(1,2)
      YY=DY(1)+XI2*DJAC(2,1)+XI3*DJAC(2,2)
C
      UH =0D0
      UHX=0D0
      UHY=0D0
      DO 210 JDOFE=1,IDFL
      IEQ=KDFG(JDOFE)
      ILO=KDFL(JDOFE)
      UH =UH +DU(IEQ)*DBAS(ILO,1)
      UHX=UHX+DU(IEQ)*DBAS(ILO,2)
      UHY=UHY+DU(IEQ)*DBAS(ILO,3)
210   CONTINUE
C
      ERRL2=ERRL2+OM*(U(XX,YY)-UH)**2
```

```
      ERRH1=ERRH1+OM*((UX(XX,YY)-UHX)**2+(UY(XX,YY)-UHY)**2)
      DNL2=DNL2+OM*U(XX,YY)**2
      DNH1=DNH1+OM*(UX(XX,YY)**2+UY(XX,YY)**2)
C
200   CONTINUE
100   CONTINUE
C
      IF (DNL2.LT.1D-15) THEN
       WRITE(MTERM,*) '* ELPT * EXACT SOLUTION ZERO !!!'
       WRITE(MPROT,*) '* ELPT * EXACT SOLUTION ZERO !!!'
       DNL2=1D0
       DNH1=1D0
      ENDIF
C
      IF (MT.GT.0) THEN
       WRITE(MTERM,*)
       WRITE(MTERM,*) '* ELPT * SOLUTION ELE      ',IELTYP
       WRITE(MTERM,*) '* ELPT * CUBATURE FORMULA ',ICUB
       WRITE(MTERM,*) '* ELPT * REL. L2-ERROR    ',SQRT(ERRL2/DNL2)
       WRITE(MTERM,*) '* ELPT * REL. H1-ERROR    ',SQRT(ERRH1/DNH1)
       IF (MT.GT.1) THEN
        WRITE(MTERM,*) '* ELPT * FUNCTION-VALUES L2-NORM ',SQRT(DNL2)
        WRITE(MTERM,*) '* ELPT * FUNCTION-VALUES H1-NORM ',SQRT(DNH1)
       ENDIF
      ENDIF
      IF (MTERM.NE.MPROT) THEN
       WRITE(MPROT,*)
       WRITE(MPROT,*) '* ELPT * SOLUTION ELE      ',IELTYP
       WRITE(MPROT,*) '* ELPT * CUBATURE FORMULA ',ICUB
       WRITE(MPROT,*) '* ELPT * REL. L2-ERROR    ',SQRT(ERRL2/DNL2)
       WRITE(MPROT,*) '* ELPT * REL. H1-ERROR    ',SQRT(ERRH1/DNH1)
       IF (M.GT.1) THEN
        WRITE(MPROT,*) '* ELPT * FUNCTION-VALUES L2-NORM ',SQRT(DNL2)
        WRITE(MPROT,*) '* ELPT * FUNCTION-VALUES H1-NORM ',SQRT(DNH1)
       ENDIF
      ENDIF
C
99999 END
C
C
C
C *** Input data file CTEST1.DAT
C
1 M
1 MT
5 NFINE
3 ICUB
4 ICUB RHS
400 NIT CG
1D-6 EPS CG
1D0   OMEGA CG
2 ICUBM ELIT
8 ICUB2 ELPT
66 OUTPUT UNIT
'TEST1.OUT' OUTPUT FILENAME
C
C
C
C *** Coarse grid file triat1
C
```

```
Grobgitter triat1 - Einheitsquadrat - Dreiecksgitter
Parametrisierung PARXQ, PARYQ, TMAXQ
      2  4  0  3  1           NEL NVT NMT NVE NBCT
DCORVG
      0D0                      0D0
      1D0                      0D0
      2D0                      0D0
      3D0                      0D0
KVERT
      1  2  3
      1  3  4
KNPR
      1  1  1  1
KMM
      1  4
```

```
      PROGRAM SAMPLE2
C
C     Sample program
C     Demonstration of FEAT
C
C     Problem  -u   - u    = f
C                xx    yy
C
C     Exact solution  u = .25D0*(1-x)*(2-x)*(1-y)*(2-y)
C     Homogeneous Dirichlet boundary conditions on bdry ( (1,2)x(1,2) )
C     Inhomogeneous Neumann boundary conditions on bdry ( (0,3)x(0,3) )
C
C     Bilinear quadrilateral element
C
C     Domain: (0,3)x(0,3) \ [1,2]x[1,2]
C
C
C *** Standard declarations and parameter settings
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      PARAMETER (NNARR=299,NNAB=21,NNWORK=300000)
C
C *** NBLOCA=1 means that the stiffness matrix only consists of one block,
C     analogously NBLOCF stands for the right hand side.
      PARAMETER (NBLOCA=1,NBLOCF=1)
C
C *** Standard declaration for CHARACTER quantities in /CHAR/
      CHARACTER SUB*6,FMT*15,CPARAM*120

C *** CFILE needed as filename for input in XORSC
C     ARRDA and ARRDF are used only for messages
      CHARACTER CFILE*15,ARRDA*6,ARRDF*6
C
C *** Standard dimensioning for workspace concept
      DIMENSION VWORK(1),KWORK(1)
C
C *** KABA  - structure of the stiffness matrix
C *** KF    -  structure of the right hand side
C *** KABAN - number of terms in the integrand for the stiffness matrix
C *** KFN   - number of terms in the integrand for the right hand side
      DIMENSION KABA(2,NNAB,NBLOCA),KF(NNAB,NBLOCF)
      DIMENSION KABAN(NBLOCA),KFN(NBLOCF)
C
C *** BCONA, BCONF - .TRUE. if the corresponding coefficients are constant
      DIMENSION BCONA(NBLOCA),BCONF(NBLOCF)
      DIMENSION BSNGLA(NBLOCA),BSNGLF(NBLOCF)
C

C *** LA, LF - numbers for matrix hand right hand side
      DIMENSION LA(NBLOCA),LF(NBLOCF)
C
C *** ARRDA, ARRDF - Names of matrices and right hand sides (for messages only)
      DIMENSION ARRDA(NBLOCA),ARRDF(NBLOCF)
      DIMENSION BSNGLA(NBLOCA),BSNGLF(NBLOCF)
C
C *** Standard COMMON blocks
      COMMON          NWORK,IWORK,IWMAX,L(NNARR),DWORK(NNWORK)
      COMMON /ERRCTL/ IER,ICHECK
      COMMON /CHAR/   SUB,FMT(3),CPARAM
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /TRIAA/  LCORVG,LCORMG,LVERT,LMID,LADJ,LVEL,LMEL,LNPR,LMM,
```

```
     *               LVBD,LEBD,LBCT,LVBDP,LMBDP
      COMMON /OUTPUT/ M,MT,MKEYB,MTERM,MERR,MPROT,MSYS,MTRC,IRECL8
C
C *** EQUIVALENCE statement needed for DWORK concept
      EQUIVALENCE (DWORK(1),VWORK(1),KWORK(1))
C
C *** Parametrization of the domain
      EXTERNAL PARXH,PARYH,TMAXH
C *** Coefficient of stiffness matrix, right hand side, exact solution
      EXTERNAL COEFFA,COEFF1,RHS,UE,UEX,UEY
C *** Control of refinement - here, regular refinement is used
      EXTERNAL S2DI0,S2DB0
C *** E011 - bilinear element ; dummy program I000 needed in call of XIE017
      EXTERNAL E011,I000
C
C *** Numbers of terms in stiffness matrix and right hand side
      DATA KABAN/2/,KFN/1/
C *** Coefficients constant for the matrix, but not for the right hand side
      DATA BCONA/.TRUE./,BCONF/.FALSE./
C *** Names of matrix and vector (for messages only)
      DATA ARRDA/'DA    '/,ARRDF/'DF    '/
      DATA BSNGLA/.FALSE./,BSNGLF/.FALSE./
C
C
C
C *** Initialization
C *** Default names are used for output devices
C
      CALL ZINIT(NNWORK,'feat.msg',' ',' ',' ',' ')
C
C
C *** Structure of bilinear and linear forms
C
      KABA(1,1,1)=2
      KABA(2,1,1)=2
      KABA(1,2,1)=3
      KABA(2,2,1)=3
C
      KF(1,1)=1
C
C *** Perform elementary checks only
      ICHECK=1
      CALL ZTIME(TIME1)
C
C *** Open data file for input
      OPEN (88,FILE='CTEST2.DAT')
C
C *** Read in message levels for files and for the screen
      READ(88,*) M
      WRITE(MTERM,*) 'M = ',M
      READ(88,*) MT
      WRITE(MTERM,*) 'MT = ',MT
C
C *** Read in coarse grid from file 'TRIAQ2', opened as unit 55
      CFILE='triaq2'
      MUNITT=55
      CALL XORSC(MUNITT,CFILE)
      IF (IER.NE.0) GOTO 99998
C
C *** Read in desired number of refinements
```

```
      READ(88,*) NFINE
      WRITE(MTERM,*) 'NFINE = ',NFINE
C
C *** Construct a regular subvision of the given coarse grid
C *** IMID=0  - Numbers of midpoints are not needed for bilinear elements
C *** IADJ=0  - Numbers of neighbouring elements can be neglected
C *** IDISP=1 - Release all free space
C *** IBDP=2  - Keep information on parameter values for boundary vertices
C ***           Store numbers of boundary vertices on KVBD
C *** Routines for controlling the refinement and parametrization are passed as
C *** EXTERNALs
      IMID=0
      IADJ=0
      IVEL=0
      IDISP=1
      IBDP=2
      CALL XSB0X(NFINE,IMID,IADJ,IVEL,IDISP,IBDP,
     *           S2DI0,S2DB0,PARXH,PARYH,TMAXH)
      IF (IER.NE.0) GOTO 99998
C
C *** Calculate the pointer vectors for the stiffness matrix
C *** Storage technique 7 - symmetry of the matrix is neglected
C *** This makes it easier to implement boundary conditions
      ISYMM=0
      CALL XAP7(LCOLA,LLDA,NA,NEQ,E011,ISYMM)
      IF (IER.NE.0) GOTO 99998
C
C *** Allocate solution vector DU in the correct length NEQ, determined by XAP7
      CALL ZNEW(NEQ,1,LU,'DU    ')
      IF (IER.NE.0) GOTO 99998
C
C
C *** Calculation of the stiffness matrix
C *** Number is returned in LA(1)
C *** Symmetric elements are not calculated twice (ISYMM=2)
C *** Number of cubature formula read in as a parameter
      LA(1)=0
      READ(88,*) ICUB
      WRITE(MTERM,*) 'ICUB FOR MATRIX = ',ICUB
      ICLR=1
      CALL XAB07(LA,LCOLA,LLDA,NA,NEQ,NBLOCA,ICLR,E011,
     *           COEFFA,BCONT,KABA,KABAN,ICUB,ISYMM,ARRDA,BSNGLA)
      IF (IER.NE.0) GOTO 99998
C
C *** Calculation of the right hand side vector
C *** Number is returned in LF(1)
C *** Number of cubature formula read in as a parameter
      LF(1)=0
      READ(88,*) ICUB
      WRITE(MTERM,*) 'ICUB RIGHT HAND SIDE = ',ICUB
      CALL XVB0(LF,NEQ,NBLOCF,ICLR,E011,
     *          RHS,BCONF,KF,KFN,ICUB,ARRDF,BSNGLF)
      IF (IER.NE.0) GOTO 99998
      READ(88,*) ICUB
      WRITE(MTERM,*) 'ICUB BOUNDARY INTEGRAL= ',ICUB
      CALL XVB1(LF,NEQ,NBLOCF,0,TMAXH,0D0,12D0,
     *          1,E011,COEFF1,BCONF,KF,KFN,ICUB,ARRDF,BSNGLF)
      IF (IER.NE.0) GOTO 99998
C
C *** User provided program for implementation of boundary conditions
```

```
      CALL BDRY(DWORK(L(LA(1))),KWORK(L(LCOLA)),KWORK(L(LLDA)),
     *          KWORK(L(LVBD)),KWORK(L(LBCT)),DWORK(L(LU)),
     *          DWORK(L(LF(1))),NEQ,DWORK(L(LCORVG)))
C
C *** Elements with modulus smaller than 1D-15 are considered as zero and are
C *** deleted from matrix LA(1)), i.e. DA. The pointer vector LCOLA
C *** i.e. the vector KCOLA is changed correspondingly.
      CALL XRC17 (LA,LCOLA,LLDA,NA,NEQ,NBLOCA,1D-15,1,
     *           'DA    ','KCOLA ')
      IF (IER.NE.0) GOTO 99998
C
C *** Parameters for cg solution algorithm are read from data file
C *** NIT maximum number of iterations, EPS desired accuracy,
C *** OMEGA preconditioning parameter
      READ(88,*) NIT
      WRITE(MTERM,*) 'NIT CG ',NIT
      READ(88,*) EPS
      WRITE(MTERM,*) 'EPS CG ',EPS
      READ(88,*) OMEGA
      WRITE(MTERM,*) 'OMEGA CG ',OMEGA
C
C *** The total CPU time for the preparation of the linear system is displayed.
      CALL ZTIME(TIME2)
      WRITE (MPROT,*) 'TIME PREPARATION ',TIME2-TIME1
C
C *** RBNORM is set to the L2-norm of the right hand side
      CALL XLL21(LF(1),NEQ,RBNORM)
C *** EPS is normalized by the norm of the right hand side (relative accuracy)
      EPS=EPS*RBNORM
C *** the preconditioned conjugate gradient method is used for solution of the
C *** linear system
C *** OMEGA<0 - no prec. , OMEGA=0 - scaling by diagonal
C *** 0 < OMEGA < 2 - SSOR preconditioning
C *** For values OMEGA>2, I000 must be replaced by the prec. subprogram
      CALL ZTIME(TIME1)
      CALL XIE017(LA(1),LCOLA,LLDA,LU,LF(1),NEQ,NIT,ITE,EPS,OMEGA,I000)
C
C *** The cpu time for the solution is displayed
      CALL ZTIME(TIME2)
      WRITE (MPROT,*) 'TIME SOLVER ',TIME2-TIME1
C
C *** A user provided subprogram is called to calculate the maximum error
C *** The maximum is calculated over all cubature points of formula ICUB
      CALL ZTIME(TIME1)
      READ(88,*) ICUBM
      WRITE(MTERM,*) 'ICUB ELIQ ',ICUBM
      CALL ELIQ(DWORK(L(LU)),KWORK(L(LVERT)),
     *          KWORK(L(LMID)),DWORK(L(LCORVG)),E011,ICUBM,UE)
      IF (IER.NE.0) GOTO 99998
C
C *** A user provided subprogram is called to calculate the
C *** L2- and H1-error
      READ(88,*) ICUB2
      WRITE(MTERM,*) 'ICUB ELPQ ',ICUB2
      CALL ELPQ(DWORK(L(LU)),KWORK(L(LVERT)),
     *          KWORK(L(LMID)),DWORK(L(LCORVG)),E011,ICUB2,UE,UEX,UEY)
      IF (IER.NE.0) GOTO 99998
C
C
C
```

```
C *** The solution vector is written to file CFILE (unit IFILE)
C *** XOWA uses the standard FORMAT contained in FMT(1), for DOUBLE PRECISION
C *** vectors
      WRITE(MTERM,*) 'OUTPUT DATA ONTO FILE ?'
      READ(88,*) IFILE
      WRITE(MTERM,*) IFILE
      WRITE(MTERM,*) 'OUTPUT FILENAME ?'
      READ(88,*) CFILE
      WRITE(MTERM,*) CFILE
      IF (IFILE.NE.0) CALL XOWA(LU,'DU    ',IFILE,CFILE,1)
C
C *** The CPU time for the calculation of the error is displayed
      CALL ZTIME(TIME2)
      WRITE(MTERM,*) 'TIME POSTPROCESSING ',TIME2-TIME1
C
C *** Write triangulation and solution vector (vertices only) to the disk
C *** in MOVIE.BYU format
      CALL ZDISP(NVT,LU,'DU    ')
      IF (IER.NE.0) GOTO 99998
      CALL XGOWFM(LU,69,'DU.TEST2.OUT')
      IF (IER.NE.0) GOTO 99998
      CALL XGOWSM(70,'TRIA.TEST2.OUT')
      IF (IER.NE.0) GOTO 99998
C *** NWORK contains the total number of DOUBLE PRECISION elements of DWORK
C *** IWMAX is the maximum number of elements of DWORK used in the current
C *** program
      WRITE(MTERM,*) 'NWORK ',NWORK
      WRITE(MTERM,*) 'IWMAX ',IWMAX
C
      GOTO 99999
99998 WRITE(MTERM,*) 'IER', IER
      WRITE(MTERM,*) 'IN SUBROUTINE ',SUB
99999 END
C
C
C
C *** Implementation of Dirichlet boundary conditions
C *** for elements with degree of freedom at vertices
      SUBROUTINE BDRY(DA,KCOL,KLD,KVBD,KBCT,DX,DB,NEQ,DCORVG)
C
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      DIMENSION DA(*),KCOL(*),KLD(*),KVBD(*),DX(*),DB(*),KBCT(*)
      DIMENSION DCORVG(2,*)
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      SAVE /TRIAD/
C
C *** UE is the exact solution
      XBD(X,Y)=UE(X,Y)
C
C *** First loop over all boundary vertices
C *** The numbers of the vertices are contained in KVBD (set by XSVEB)
      DO 1 IVT=KBCT(2),KBCT(3)-1
      IEQ=KVBD(IVT)
C *** The diagonal element is set to 1
      DA(KLD(IEQ))=1D0
C *** DCORVG(1,IEQ) contains the parameter of the boundary point
      X=DCORVG(1,IEQ)
      Y=DCORVG(2,IEQ)
      DX(IEQ)=XBD(X,Y)
      DB(IEQ)=XBD(X,Y)
```

```
C *** The nondiagonal elements of the matrix are set to zero
      DO 2 ICOL=KLD(IEQ)+1,KLD(IEQ+1)-1
2     DA(ICOL)=0D0
C
1     CONTINUE
      END
C
C
C
C *** The following three subprograms describe the parametrization of the
C *** boundary of the square\square
C
      DOUBLE PRECISION FUNCTION PARXH(T,IBCT)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      IF (IBCT.EQ.1) THEN
       IF (T.GE.9.D0) GOTO 150
       IF (T.GE.6.D0) GOTO 160
       IF (T.GE.3.D0) GOTO 170
       PARXH=T
       GOTO 99999
170    PARXH=3D0
       GOTO 99999
160    PARXH=9D0-T
       GOTO 99999
150    PARXH=0D0
      ELSE
       IF (T.GE.3.D0) GOTO 250
       IF (T.GE.2.D0) GOTO 260
       IF (T.GE.1.D0) GOTO 270
       PARXH=1D0
       GOTO 99999
270    PARXH=T
       GOTO 99999
260    PARXH=2D0
       GOTO 99999
250    PARXH=5D0-T
      ENDIF
99999 END
C
C
C
      DOUBLE PRECISION FUNCTION PARYH(T,IBCT)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
IF (IBCT.EQ.1) THEN
       IF (T.GE.9.D0) GOTO 150
       IF (T.GE.6.D0) GOTO 160
       IF (T.GE.3.D0) GOTO 170
       PARYH=0D0
       GOTO 99999
170    PARYH=T-3D0
       GOTO 99999
160    PARYH=3D0
       GOTO 99999
150    PARYH=12D0-T
      ELSE
       IF (T.GE.3.D0) GOTO 250
       IF (T.GE.2.D0) GOTO 260
       IF (T.GE.1.D0) GOTO 270
       PARYH=T+1D0
       GOTO 99999
```

```
270    PARYH=2D0
       GOTO 99999
260    PARYH=4D0-T
       GOTO 99999
250    PARYH=1D0
      ENDIF
99999 END
C
C     IBCT=1 denotes outer boundary in parametrization
C
      DOUBLE PRECISION FUNCTION TMAXH(IBCT)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      IF (IBCT.EQ.1) THEN
       TMAXH=12D0
      ELSE
       TMAXH=4D0
      ENDIF
      END
C
C
C *** Coefficient function for the stiffness matrix
C
      DOUBLE PRECISION FUNCTION COEFFA(X,Y,IA,IB,IDA,BFIRST)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      COEFFA=1D0
C
      END
C
C *** Coefficient function for the boundary integral
C
      DOUBLE PRECISION FUNCTION COEFF1(X,Y,IB,IDA,BFIRST)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      IF (X.EQ.3D0) THEN
       COEFF1=.25D0*(-3D0+2D0*X)*(1D0-Y)*(2D0-Y)
      ELSE IF (Y.EQ.3D0) THEN
       COEFF1=.25D0*(-3D0+2D0*Y)*(1D0-X)*(2D0-X)
      ELSE IF (X.EQ.0D0) THEN
       COEFF1=-.250*(-3D0+2D0*X)*(1D0-Y)*(2D0-Y)
      ELSE IF (Y.EQ.0D0) THEN
       COEFF1=-.25D0*(-3D0+2D0*Y)*(1D0-X)*(2D0-X)
      ENDIF
C
      END
C
C *** Right hand side yielding the exact solution UE
C
      DOUBLE PRECISION FUNCTION RHS(X,Y,IA,IDA,BFIRST)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      RHS=-0.5D0*((1D0-Y)*(2D0-Y)+(1D0-X)*(2D0-X))
      END
C
C
C *** Exact solution
C
      DOUBLE PRECISION FUNCTION UE(X,Y)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      UE=0.25D0*(1D0-X)*(2D0-X)*(1D0-Y)*(2D0-Y)
      END
C
C *** Exact UEX
```

```
C
      DOUBLE PRECISION FUNCTION UEX(X,Y)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      UEX=0.25D0*(-3D0+2D0*X)*(1D0-Y)*(2D0-Y)
      END
C
C
C *** Exact UEY
C
      DOUBLE PRECISION FUNCTION UEY(X,Y)
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      UEY=0.25D0*(-3D0+2D0*Y)*(1D0-X)*(2D0-X)
      END
C
C
C
C *** Subprogram to determine the maximum error at the cubature points of
C *** cubature formula ICUB
C *** The procedure for the evaluation of the discrete function is copied from
C *** the FEAT subprogram AB017
C
      SUBROUTINE ELIQ(DU,KVERT,KMID,DCORVG,ELE,ICUB,U)
C
      IMPLICIT DOUBLE PRECISION(A,C-H,O-U,W-Z),LOGICAL(B)
      CHARACTER FMT*15,SUB*6,CPARAM*120
      PARAMETER (NNBAS=21,NNDER=6,NNCUBP=36,NNVE=4)
      DIMENSION DU(*)
      DIMENSION KVERT(NNVE,*),KMID(NNVE,*),DCORVG(2,*)
      COMMON /ERRCTL/ IER,ICHECK
      COMMON /ELEM/   DX(NNVE),DY(NNVE),DJAC(2,2),DETJ,
     1                DBAS(NNBAS,NNDER),BDER(NNDER),KVE(NNVE),IEL
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /TRIAA/  LCORVG,LCORMG,LVERT,LMID,LADJ,LVEL,LMEL,LNPR,LMM,
     *                LVBD,LEBD,LBCT,LVBDP,LMBDP
      COMMON /CHAR/   SUB,FMT(3),CPARAM
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
      COMMON /OUTPUT/ M,MT,MKEYB,MTERM,MERR,MPROT,MSYS,MTRC,IRECL8
      COMMON /COAUX1/ KDFG(NNBAS),KDFL(NNBAS),IDFL
      SAVE
C
      IER=0
C
C *** Preparation
C *** Set number of element
      IELTYP=-1
      CALL ELE(0D0,0D0,IELTYP)
      IF (IER.NE.0) GOTO 99999
C *** Determine number of d.o.f. per element
      IDFL=NDFL(IELTYP)
      IF (IER.LT.0) GOTO 99999
C *** Determine coordinates (and weights) of cubature points
      CALL CB2Q(ICUB)
      IF (IER.NE.0) GOTO 99999
      DO 1 IDER=1,NNDER
1     BDER(IDER)=.FALSE.
C *** Only function values are needed
      BDER(1)=.TRUE.
C
      ERRM=0D0
      DNM=0D0
```

```
C
C *** Let the element subprogram save arithmetic operations
      CALL ELE(0D0,0D0,-2)
      IF (IER.LT.0) GOTO 99999
C
C *** Loop over all elements
C
      DO 100 IEL=1,NEL
C
C *** Find global and local d.o.f. for element IEL
      CALL NDFGL(IEL,1,IELTYP,KVERT,KMID,KDFG,KDFL)
      IF (IER.LT.0) GOTO 99999
C
C *** Determine cartesian coordinates of all vertices
      DO 110 IVE=1,NVE
      JP=KVERT(IVE,IEL)
      KVE(IVE)=JP
      DX(IVE)=DCORVG(1,JP)
      DY(IVE)=DCORVG(2,JP)
110   CONTINUE
C
C *** Calculate jacobian of transformation to the (4 fold) unit square at the
C *** center of gravity
      DJ1=0.5D0*(-DX(1)-DX(2)+DX(3)+DX(4))
      DJ2=0.5D0*( DX(1)-DX(2)+DX(3)-DX(4))
      DJ3=0.5D0*(-DY(1)+DY(2)-DY(3)+DY(4))
      DJ4=0.5D0*(-DY(1)+DY(2)+DY(3)-DY(4))
C
C *** Loop over all cubature points
      DO 200 ICUBP=1,NCUBP
C
C *** Position of cubature points
      XI1=DXI(ICUBP,1)
      XI2=DXI(ICUBP,2)
C
C *** Jacobian and determinant at cubature point ICUBP
      DJAC(1,1)=0.5D0*(DX(2)-DX(1)+DJ2)+0.5D0*DJ2*XI2
      DJAC(1,2)=0.5D0*DJ1+0.5D0*DJ2*XI1
      DJAC(2,1)=0.5D0*DJ4-0.5D0*DJ3*XI2
      DJAC(2,2)=0.5D0*(DY(3)-DY(1)-DJ4)-0.5D0*DJ3*XI1
      DETJ=DJAC(1,1)*DJAC(2,2)-DJAC(1,2)*DJAC(2,1)
C
C *** Call element subprogram to calculate function values of all basis
C *** functions at the cubature point, ELE may use information previously
C *** stored
      CALL ELE(XI1,XI2,-3)
      IF (IER.LT.0) GOTO 99999
C
C *** Cartesian coordiantes of cubature point
      XX=0.5D0*(DX(1)+DX(2)+DJ1)+0.5D0*(DX(2)-DX(1)+DJ2)*XI1
     *  +0.5D0*DJ1*XI2+0.5D0*DJ2*XI1*XI2
      YY=0.5D0*(DY(1)+DY(3)+DJ3)+0.5D0*DJ4*XI1+0.5D0*
     *         (DY(3)-DY(1)-DJ4)*XI2-0.5D0*DJ3*XI1*XI2
C
      UH=0D0
C
C *** Calculate discrete solution by summing over all d.o.f.
      DO 210 JDFL=1,IDFL
      IEQ=KDFG(JDFL)
      ILO=KDFL(JDFL)
```

```
      UH=UH+DU(IEQ)*DBAS(ILO,1)
210   CONTINUE
C
C *** Compare with exact solution U
      ELML=ABS(U(XX,YY)-UH)
C
C *** ERRM contains the maximum error
C *** DNM  contains the maximum function value
      ERRM=MAX(ERRM,ELML)
      DNM= MAX(DNM,ABS(U(XX,YY)))
C
200   CONTINUE
100   CONTINUE
C
C *** Errors are displayed on the screen and/or on the output unit MPROT
      WRITE(MTERM,*)
      WRITE(MTERM,*) '* ELIQ * DISCRETE SOLUTION   ',IELTYP
      WRITE(MTERM,*) '* ELIQ * EVALUATION PTS AS FORMULA ',ICUB
      WRITE(MTERM,*) '* ELIQ * MAX ERR FUNCTION-VALUES ',ERRM
      IF (M.GE.1)
     * WRITE(MTERM,*) '* ELIQ * MAXIMUM FUNCTION-VALUES ',DNM
      IF (MTERM.NE.MPROT) THEN
       WRITE(MPROT,*)
       WRITE(MPROT,*) '* ELIQ * DISCRETE SOLUTION   ',IELTYP
       WRITE(MPROT,*) '* ELIQ * EVALUATION PTS AS FORMULA ',ICUB
       WRITE(MPROT,*) '* ELIQ * MAX ERR FUNCTION-VALUES ',ERRM
       IF (M.GE.1)
     *  WRITE(MPROT,*) '* ELIQ * MAXIMUM FUNCTION-VALUES ',DNM
      ENDIF
C
99999 END
C
C
      SUBROUTINE ELPQ(DU,KVERT,KMID,DCORVG,ELE,ICUB,U,UX,UY)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      PARAMETER (NNBAS=21,NNDER=6,NNCUBP=36,NNVE=4)
      CHARACTER FMT*15,SUB*6,CPARAM*120
      DIMENSION DU(*),KVERT(NNVE,*),KMID(NNVE,*),DCORVG(2,*)
      DIMENSION KDFG(NNBAS),KDFL(NNBAS)
      COMMON /OUTPUT/ M,MT,MKEYB,MTERM,MERR,MPROT,MSYS,MTRC,IRECL8
      COMMON /ERRCTL/ IER,ICHECK
      COMMON /CHAR/   SUB,FMT(3),CPARAM
      COMMON /ELEM/   DX(NNVE),DY(NNVE),DJAC(2,2),DETJ,
     *                DBAS(NNBAS,NNDER),BDER(NNDER),KVE(NNVE),IEL
      COMMON /TRIAD/  NEL,NVT,NMT,NVE,NVEL,NBCT,NVBD
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
      COMMON /COAUX1/ KDFG,KDFL,IDFL
      SAVE
C
      SUB='ELPQ'
      IER=0
C
      IELTYP=-1
      CALL ELE(0D0,0D0,IELTYP)
      IF (IER.NE.0) GOTO 99999
      IDFL=NDFL(IELTYP)
      IF (IER.LT.0) GOTO 99999
      CALL CB2Q(ICUB)
      IF (IER.NE.0) GOTO 99999
      DO 1 IDER=1,NNDER
```

```
1     BDER(IDER)=.FALSE.
      BDER(1)=.TRUE.
      BDER(2)=.TRUE.
      BDER(3)=.TRUE.
C
      ERRL2=0D0
      ERRH1=0D0
      DNL2=0D0
      DNH1=0D0
      CALL ELE(0D0,0D0,-2)
C
      DO 100 IEL=1,NEL
C
      CALL NDFGL(IEL,1,IELTYP,KVERT,KMID,KDFG,KDFL)
      IF (IER.LT.0) GOTO 99999
C
      DO 110 IVE=1,NVE
      JP=KVERT(IVE,IEL)
      KVE(IVE)=JP
      DX(IVE)=DCORVG(1,JP)
      DY(IVE)=DCORVG(2,JP)
110   CONTINUE
C
      DJ1=0.5D0*(-DX(1)-DX(2)+DX(3)+DX(4))
      DJ2=0.5D0*( DX(1)-DX(2)+DX(3)-DX(4))
      DJ3=0.5D0*(-DY(1)+DY(2)-DY(3)+DY(4))
      DJ4=0.5D0*(-DY(1)+DY(2)+DY(3)-DY(4))
C
      DO 200 ICUBP=1,NCUBP
C
      XI1=DXI(ICUBP,1)
      XI2=DXI(ICUBP,2)
C
      DJAC(1,1)=0.5D0*(DX(2)-DX(1)+DJ2)+0.5D0*DJ2*XI2
      DJAC(1,2)=0.5D0*DJ1+0.5D0*DJ2*XI1
      DJAC(2,1)=0.5D0*DJ4-0.5D0*DJ3*XI2
      DJAC(2,2)=0.5D0*(DY(3)-DY(1)-DJ4)-0.5D0*DJ3*XI1
      DETJ=DJAC(1,1)*DJAC(2,2)-DJAC(1,2)*DJAC(2,1)
C
      OM=DOMEGA(ICUBP)*DETJ
C
      CALL ELE(XI1,XI2,-3)
      IF (IER.LT.0) GOTO 99999
C
      XX=0.5D0*(DX(1)+DX(2)+DJ1)+0.5D0*(DX(2)-DX(1)+DJ2)*XI1
     *  +0.5D0*DJ1*XI2+0.5D0*DJ2*XI1*XI2
      YY=0.5D0*(DY(1)+DY(3)+DJ3)+0.5D0*DJ4*XI1+0.5D0*
     *  (DY(3)-DY(1)-DJ4)*XI2-0.5D0*DJ3*XI1*XI2

C
      UH =0D0
      UHX=0D0
      UHY=0D0
      DO 210 JDOFE=1,IDFL
      IEQ=KDFG(JDOFE)
      ILO=KDFL(JDOFE)
      UH =UH +DU(IEQ)*DBAS(ILO,1)
      UHX=UHX+DU(IEQ)*DBAS(ILO,2)
      UHY=UHY+DU(IEQ)*DBAS(ILO,3)
210   CONTINUE
```

```
C
      ERRL2=ERRL2+OM*(U(XX,YY)-UH)**2
      ERRH1=ERRH1+OM*((UX(XX,YY)-UHX)**2+(UY(XX,YY)-UHY)**2)
      DNL2=DNL2+OM*U(XX,YY)**2
      DNH1=DNH1+OM*(UX(XX,YY)**2+UY(XX,YY)**2)
C
200   CONTINUE
100   CONTINUE
C
      IF (DNL2.LT.1D-15) THEN
       WRITE(MTERM,*) '* ELPQ * EXACT SOLUTION ZERO !!!'
       WRITE(MPROT,*) '* ELPQ * EXACT SOLUTION ZERO !!!'
       DNL2=1D0
       DNH1=1D0
      ENDIF

       IF (MT.GT.0) THEN
       WRITE(MTERM,*)
       WRITE(MTERM,*) '* ELPQ * SOLUTION ELE      ',IELTYP
       WRITE(MTERM,*) '* ELPQ * CUBATURE FORMULA ',ICUB
       WRITE(MTERM,*) '* ELPQ * REL. L2-ERROR     ',SQRT(ERRL2/DNL2)
       WRITE(MTERM,*) '* ELPQ * REL. H1-ERROR     ',SQRT(ERRH1/DNH1)
        IF (MT.GT.1) THEN
        WRITE(MTERM,*) '* ELPQ * FUNCTION-VALUES L2-NORM ',SQRT(DNL2)
         WRITE(MTERM,*) '* ELPQ * FUNCTION-VALUES H1-NORM ',SQRT(DNH1)
        ENDIF
      ENDIF
       IF (MTERM.NE.MPROT) THEN
       WRITE(MPROT,*)
       WRITE(MPROT,*) '* ELPQ * SOLUTION ELE      ',IELTYP
       WRITE(MPROT,*) '* ELPQ * CUBATURE FORMULA ',ICUB
       WRITE(MPROT,*) '* ELPQ * REL. L2-ERROR     ',SQRT(ERRL2/DNL2)
       WRITE(MPROT,*) '* ELPQ * REL. H1-ERROR     ',SQRT(ERRH1/DNH1)
        IF (M.GT.1) THEN
         WRITE(MPROT,*) '* ELPQ * FUNCTION-VALUES L2-NORM ',SQRT(DNL2)
         WRITE(MPROT,*) '* ELPQ * FUNCTION-VALUES H1-NORM ',SQRT(DNH1)
        ENDIF
       ENDIF
99999 END
C
C
C
C *** Input data file CTEST3.DAT
C
1 M
1 MT
4 NFINE
4 ICUB
4 ICUB RHS
5 ICUB BDRY INTEGRAL
400 NIT CG
1D-6 EPS CG
1.0D0 OMEGA CG
2 ICUBM ELIQ
8 ICUB2 ELPQ
66 OUTPUT UNIT
'TEST3.OUT' OUTPUT FILENAME
C
C
C
```

```
C *** Coarse grid data file triaq2
C
Coarse grid triaq2
Parametrisation PARXH, PARYH, TMAXH
      8  16   0   4   2           NEL NVT NMT NVE NBCT
DCORVG
      0D0                         0D0
      1D0                         0D0
      2D0                         0D0
      3D0                         0D0
      4D0                         0D0
      5D0                         0D0
      6D0                         0D0
      7D0                         0D0
      8D0                         0D0
      9D0                         0D0
     10D0                         0D0
     11D0                         0D0
      0D0                         0D0
      3D0                         0D0
      2D0                         0D0
      1D0                         0D0
KVERT
     1    2   13   12
     2    3   14   13
     3    4    5   14
     5    6   15   14
     6    7    8   15
     8    9   16   15
     9   10   11   16
    11   12   13   16
KNPR
      1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2
KMM
      1  12  13  14
```