FEAT3D
Finite Element Analysis Tools in 3 Dimensions
User Manual
Release 1.2

Joachim Harig, Peter Schreiber, Stefan Turek
Universität Heidelberg, Institut für Angewandte Mathematik
Im Neuenheimer Feld 293, D–69120 Heidelberg

Heidelberg, März 1994

Contents

Short description

The progam package FEAT3D is a general purpose subroutine system for the numerical solution of partial differential equations by the finite element method. As an extension to the 2-D finite element program package FEAT2D (see Blum/Harig/Müller/Turek [2],[3]) the subroutines of FEAT3D are designed to handle problems in three space dimensions.

The collection of subroutines in FEAT3D covers only those tasks in 3-D applications that cannot be treated with FEAT2D. Pseudodynamic memory management (as explained in the FEAT2D manual) and the linear algebra part of finite element applications, for example, are taken from the FEAT2D program package. This manual comments in detail on the new 3-D routines and lists again some main FEAT2D routines that are essential to build up a 3-D finite element application. A sample program, listed at the end of the manual, demonstrates the structure of a typical finite element application. This example can be used as starting program which can be modified for the actual application.

FEAT3D 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 FEAT3D is transparent so that modifications or augmentations of the program package are very easy, for instance the implementation of new basis functions. 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 [4], [5].

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
- Error analysis for test problems where the exact solution is known
- Preparation of data for graphical postprocessing

Most FEAT3D subroutines can be used outside FEAT3D as well since the special memory management is separated from the working program units.

All informations or references to the program package FEAT2D in this manual relate to FEAT2D Release 1.3 (see [3]).

1. General Concepts and Notation

1.1. Groups of subprograms

The subprograms of FEAT3D 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. Those groups of FEAT3D subroutines that are taken from FEAT2D unchanged are marked.

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

`B` Unused

`C` Cubature formulas
Defined on reference elements or in barycentric coordinates

`D` Reserved for finite difference applications

`E` Element library

`F` Unused

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

`H` Unused

`I` FEAT2D – Iterative solvers of linear systems of equations
Versions for solution, smoothing iterations and preconditioning

`J` Reserved for eigenvalue problems

`K` Unused

`L` FEAT2D – 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` FEAT2D/FEAT3D – Input/Output

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

`Q` Unused

`R` FEAT2D – 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` FEAT2D – Error handling
Selective dump of contents of `COMMON` blocks and arrays allocated on the workspace

`X` FEAT2D/FEAT3D – 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` FEAT2D – External subroutines
Preparation of parameter lists for routines `A-W`

`Z` FEAT2D – Handling of the pseudodynamic memory management
Machine dependent system routines

1.2. Symbolic names and reserved starting letters

All routines are written in standard `FORTRAN77`. 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

Example

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. Description of the subdivision

In the current version of FEAT3D only regular subdivisions of three-dimensional domains in hexahedral elements are supported. Here, and throughout this manual with hexahedral elements we denote elements with 6 faces and 8 nodes (cubes, bricks, etc.). 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 `COMMON` blocks; the first, `/TRIAD/`, containing scalar information (dimensions) and the second, `/TRIAA/`, containing numbers of arrays describing the subdivision.

```
      COMMON /TRIAD/  NEL,NVT,NET,NAT,NVE,NEE,NAE,NVEL,NEEL,NVED,
     *                NVAR,NEAR,NBCT,NVBD,NEBD,NABD
      COMMON /TRIAA/  LCORVG,LCORMG,LCORAG,LVERT,LEDGE,LAREA,LADJ,
     *                LVEL,LEEL,LAEL,LVED,LAED,LVAR,LEAR,LEVE,LAVE,
     *                LNPR,LBCT,LVBD,LEBD,LABD
```

Explanation

a) Contents of `/TRIAD/` – Dimensions

`NEL`	Total number of elements
`NVT`	Total number of vertices
`NET`	Total number of edges `NET` is set to `0` if no information about edges is generated, e.g., numbers and/or coordinates
`NAT`	Total number of faces `NAT` is set to `0` if no information about faces is generated, e.g., numbers and/or coordinates
`NVE`	Number of vertices per element `NVE` is set to `8` for purely hexahedral meshes
`NEE`	Number of edges per element `NEE` is set to `12` for purely hexahedral meshes
`NAE`	Number of faces per element `NAE` is set to `6` for purely hexahedral meshes
`NVEL`	Maximum number of elements meeting at one vertex `NVEL` is set to `0` if this information is not available
`NEEL`	Maximum number of elements meeting at one edge `NEEL` is set to `0` if this information is not available
`NVED`	Maximum number of edges meeting at one vertex `NVED` is set to `0` if this information is not available
`NVAR`	Maximum number of faces meeting at one vertex `NVAR` is set to `0` if this information is not available
`NEAR`	Maximum number of faces meeting at one edge `NEAR` is set to `0` if this information is not available
`NBCT`	Total number of boundary components
`NVBD`	Total number of vertices on the boundary (sum over all boundary components)
`NEBD`	Total number of edges on the boundary (sum over all boundary components)
`NABD`	Total number of faces on the boundary (sum over all boundary components)

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(3,NVT)`
Array containing the cartesian coordinates of the vertices
`DCORVG(1,IVT)` – X-coordinate of vertex `IVT`
`DCORVG(2,IVT)` – Y-coordinate of vertex `IVT`
`DCORVG(3,IVT)` – Z-coordinate of vertex `IVT`

`LCORMG` `DIMENSION DCORMG(3,NET)`
Array containing the cartesian coordinates of the midpoints of edges. In the present version of FEAT3D `LCORMG` is set to `0`

`LCORAG` `DIMENSION DCORAG(3,NAT)`
Array containing the cartesian coordinates of the midpoints of faces. In the present version of FEAT3D `LCORAG` is used only in connection with the elements `E030` and `E031` (see Section 3.1)

`LVERT` `DIMENSION KVERT(NNVE,NEL)`
Array containing the numbers of vertices for each element

`LEDGE` `DIMENSION KEDGE(NNEE,NEL)`
Array containing the numbers of midpoints of edges for each element; the midpoints are given numbers ranging from `NVT+1` to `NVT+NET`. `KEDGE` and `NET` are generated by subroutine `SBE`

`LAREA` `DIMENSION KAREA(NNAE,NEL)`
Array containing the numbers of midpoints of faces for each element; the midpoints are given numbers ranging from `NVT+NET+1` to `NVT+NET+NAT`. `KAREA` and `NAT` are generated by subroutine `SBA`

`LADJ` `DIMENSION KADJ(NNAE,NEL)`
Array containing the numbers of the neighboring elements for each element of the subdivision. 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`. `KVEL` and `NVEL` are generated by subroutine `SBVEL`

`LEEL` `DIMENSION KEEL(NEEL,NET)`
Array containing the numbers of the elements meeting at an edge. Remember that `NEEL` is the maximum number of elements meeting at one of the edges. `KEEL` and `NEEL` are generated by subroutine `SBEEL`

`LAEL` `DIMENSION KAEL(2,NAT)`
Array containing the numbers of the elements meeting at a face. `KAEL` is generated by subroutine `SBAEL`

`LVED` `DIMENSION KVED(2,NET)`
Array containing the numbers of the vertices meeting at an edge

`LAED` `DIMENSION KAED(NEAR,NET)`
Array containing the numbers of the faces meeting at an edge

`LVAR` `DIMENSION KVAR(4,NAT)`
Array containing the numbers of the vertices meeting at a face

`LEAR` `DIMENSION KEAR(4,NAT)`
Array containing the numbers of the edges meeting at a face

`LEVE` `DIMENSION KEVE(NVEL,NVT)`
Array containing the numbers of the elements meeting at a vertex

`LAVE` `DIMENSION KAVE(NVAR,NVT)`
Array containing the numbers of the faces meeting at a vertex

`LNPR` `DIMENSION KNPR(NVT)`
Array containing information about the location of vertices

`KNPR(IP)=0`	if `IP` is the number of an interior vertex
`=IBCT`	if `IP` is the number of a vertex on boundary component `IBCT`

`LBCT` `DIMENSION KBCT(NBCT+1)`
Pointer vector for the array `KVBD`. `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`

`LVBD` `DIMENSION KVBD(NVBD)`
Array containing the numbers of the vertices on the boundary. All numbers are stored on a one-dimensional array

`LEBD` `DIMENSION KEBD(NEBD)`
Same as `KVBD`, but for the edges on the boundary. To each vertex `IVT` in `KVBD`, there corresponds an entry in `KEBD` which is just the number of the boundary edge following the vertex `IVT`

`LABD` `DIMENSION KABD(NABD)`
Array containing the numbers of the midpoints of the faces on the boundary. All numbers are stored on a one-dimensional array

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 faces (see elements `E030` or `E031`)
Scalars: `NAT`
Arrays: `KAREA`, `KNPR(NVT+NAT)`, `DCORAG`

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

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

Example

Domain: Unit cube $\bar{\Omega} = [0,1]^3$ – Contents of variables and arrays described above

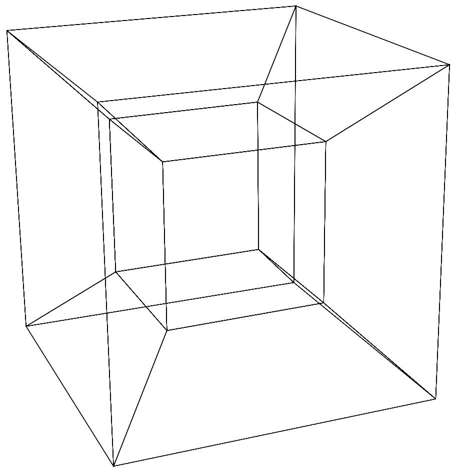

Figure 1.1: Coarse grid unit cube

Scalars

NEL	7
NVT	16
NET	32
NAT	24
NVE	8
NBCT	1
NVBD	8
NABD	6

Arrays

DCORVG - DCORVG(IDIM,IVT), IDIM=1,2,3

```
0.00D0  0.00D0  0.00D0
1.00D0  0.00D0  0.00D0
1.00D0  1.00D0  0.00D0
0.00D0  1.00D0  0.00D0
0.00D0  0.00D0  1.00D0
1.00D0  0.00D0  1.00D0
1.00D0  1.00D0  1.00D0
0.00D0  1.00D0  1.00D0
0.25D0  0.25D0  0.25D0
0.75D0  0.25D0  0.25D0
0.75D0  0.75D0  0.25D0
0.25D0  0.75D0  0.25D0
0.25D0  0.25D0  0.75D0
0.75D0  0.25D0  0.75D0
0.75D0  0.75D0  0.75D0
0.25D0  0.75D0  0.75D0
```

```
KVERT - KVERT(IVE,IEL), IVE=1,...,NNVE
 1   2   6   5   9  10  14  13
 2   3   7   6  10  11  15  14
 3   4   8   7  11  12  16  15
 1   4   8   5   9  12  16  13
 1   2   3   4   9  10  11  12
 5   6   7   8  13  14  15  16
 9  10  11  12  13  14  15  16
```

```
KEDGE - KEDGE(IEE,IEL), IEE=1,...,NNEE
  1   2   3   4   5   6   7   8   9  10  11  12
 13  14  15   2   6  16  17   7  18  19  20  10
 21  22  23  14  16  24  25  17  26  27  28  19
 29  22  30   4   5  24  25   8  31  27  32  12
  1  13  21  29   5   6  16  24   9  18  26  31
  3  15  23  30   8   7  17  25  11  20  28  32
  9  18  26  31  12  10  19  27  11  20  28  32
```

```
KAREA - KAREA(IAE,IEL), IAE=1,...,NNAE
  1   2   3   4   5   6
  7   8   9  10   3  11
 12  13  14  15   9  16
 17  18  14  19   5  20
 21   2   8  13  18  22
 23   4  10  15  19  24
 22   6  11  16  20  24
```

```
KADJ - KADJ(IAE,IEL), IAE=1,...,NNAE
 0  5  2  6  4  7
 0  5  3  6  1  7
 0  5  4  6  2  7
 0  5  3  6  1  7
 0  1  2  3  4  7
 0  1  2  3  4  7
 5  1  2  3  4  6
```

```
KNPR - KNPR(IVT), IVT=1,...,NVT
 1  1  1  1  1  1  1  1
 0  0  0  0  0  0  0  0
```

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 `NNLEV` at most refinement levels. The meaning of the variables can be deduced straightforward from the contents of `/TRIAA/` and `/TRIAD/` (for further information see Section 3.1).

```
      PARAMETER (NNLEV=9)

      COMMON /MGTRD/  KNEL(NNLEV),KNVT(NNLEV),KNET(NNLEV),
     *                KNAT(NNLEV),KNVE(NNLEV),KNEE(NNLEV),
     *                KNAE(NNLEV),KNVEL(NNLEV),KNEEL(NNLEV),
     *                KNVED(NNLEV),KNVAR(NNLEV),KNEAR(NNLEV),
     *                KNBCT(NNLEV),KNVBD(NNLEV),KNEBD(NNLEV),
     *                KNABD(NNLEV)
      COMMON /MGTRA/  KLCVG(NNLEV),KLCMG(NNLEV),KLCAG(NNLEV),
     *                KLVERT(NNLEV),KLEDGE(NNLEV),KLAREA(NNLEV),
     *                KLADJ(NNLEV),KLVEL(NNLEV),KLEEL(NNLEV),
     *                KLAEL(NNLEV),KLVED(NNLEV),KLAED(NNLEV),
     *                KLVAR(NNLEV),KLEAR(NNLEV),KLEVE(NNLEV),
     *                KLAVE(NNLEV),KLNPR(NNLEV),KLBCT(NNLEV),
     *                KLVBD(NNLEV),KLEBD(NNLEV),KLABD(NNLEV)
```

1.4. Storage techniques for matrices

FEAT3D 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
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:`	<u>`0`</u>	`1`	<u>`0`</u>	`-1`	`1`	<u>`0`</u>	`-1`
`KLD:`	`1`	`3`	`6`	`8`			
`KOP:`	`1`	`2`	`2`	`3`			
`NA:`	`7`						
`NOP:`	`3`						

1.5. 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.

```
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 FEAT3D 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 amout 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 FEAT3D 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 FEAT3D 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 occured even if no messages have been displayed. For the next release of FEAT3D 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 occured.

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 of each element is controlled during the calculation of a finite element matrix. Moreover, ICHECK controls the optional tracing of the FEAT3D 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 FEAT3D 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. FEAT3D 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.

1.6. 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, 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 3.2) we use unrolled loops unless we work on vector machines. For adapting FEAT3D 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. Compared to FEAT2D there are changes due to the space dimension in the COMMON blocks /TRIAA/, /TRIAD/ and /ELEM/. Among the quantities in PARAMETER statements the values of NNVE, NNDER and NNBAS have changed.

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,NNAB=21,NNDER=10)
      PARAMETER (NNBAS=27,NNCUBP=36,NNVE=8,NNAE=6,NNDIM=3)
```

Explanation

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, FEAT2D manual)
NNAB	Maximum number of additive terms in integrands of bilinear forms or linear forms (see description of routines AB.. and VB., Section 3.1)
NNDER	Maximum number of combinations of derivatives applied to basis functions, NNDER=10 means derivatives up to the order 2 (see description of element routines E..., Section 3.1)
NNBAS	Maximum number of local degrees of freedom Set to 27 for triquadratic basis functions
NNCUBP	Maximum number of cubature points in numerical integration formulas – Set to 36
NNVE	Maximum number of vertices per element – Set to 8
NNAE	Maximum number of faces per element – Set to 6
NNDIM	Maximum number of coupled basis functions

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, FEAT2D manual).

2.4. COMMON BLOCKS

Here, we give a complete list and explanation of all the COMMON blocks used for internal communication of the FEAT3D 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 3.2. 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

      PARAMETER (NNARR=299,NNAB=21,NNDER=10)
      PARAMETER (NNBAS=27,NNCUBP=36,NNVE=8,NNDIM=3)

      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,LCORAG,LVERT,LEDGE,LAREA,LADJ,
     *                LVEL,LEEL,LAEL,LVED,LAED,LVAR,LEAR,LEVE,LAVE
     *                LNPR,LBCT,LVBD,LEBD,LABD
      COMMON /TRIAD/  NEL,NVT,NET,NAT,NVE,NEE,NAE,NVEL,NEEL,NVED,
     *                NVAR,NEAR,NBCT,NVBD,NEBD,NABD
      COMMON /ELEM/   DX(NNVE),DY(NNVE),DZ(NNVE),DJAC(3,3),DETJ,
     *                DBAS(NNDIM,NNBAS,NNDER),BDER(NNDER),KVE(NNVE),
     *                IEL,NDIM
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
      COMMON /COAUX1/ KDFG(NNBAS),KDFL(NNBAS),IDFL
      COMMON /COAUX2/ DBAS1(NNDIM,NNBAS,NNDER,3),KDFG1(NNBAS,3),
```

```
     *                KDFL1(NNBAS,3),IDFL1(3),BDER1(NNDER,3)
      COMMON /TABLE/  KTYPE(NNARR),KLEN(NNARR),KLEN8(NNARR),IFLAG
C***  COMMON blocks for multigrid data management
      COMMON /MGPAR/  ILEV,NLEV,NLMIN,NLMAX,
     *                ICYCLE,KPRSM(NNLEV),KPOSM(NNLEV)
      COMMON /MGTRD/  KNEL(NNLEV),KNVT(NNLEV),KNET(NNLEV),
     *                KNAT(NNLEV),KNVE(NNLEV),KNEE(NNLEV),
     *                KNAE(NNLEV),KNVEL(NNLEV),KNEEL(NNLEV),
     *                KNVED(NNLEV),KNVAR(NNLEV),KNEAR(NNLEV),
     *                KNBCT(NNLEV),KNVBD(NNLEV),KNEBD(NNLEV),
     *                KNABD(NNLEV)
      COMMON /MGTIME/ TTMG,TTS,TTE,TTD,TTP,TTR,IMTIME
      COMMON /MGTRA/  KLCVG(NNLEV),KLCMG(NNLEV),KLCAG(NNLEV),
     *                KLVERT(NNLEV),KLEDGE(NNLEV),KLAREA(NNLEV),
     *                KLADJ(NNLEV),KLVEL(NNLEV),KLEEL(NNLEV),
     *                KLAEL(NNLEV),KLVED(NNLEV),KLAED(NNLEV),
     *                KLVAR(NNLEV),KLEAR(NNLEV),KLEVE(NNLEV),
     *                KLAVE(NNLEV),KLNPR(NNLEV),KLBCT(NNLEV),
     *                KLVBD(NNLEV),KLEBD(NNLEV),KLABD(NNLEV)
```

Explanation

The blank `COMMON` contains workspace information and is described in detail in the FEAT2D 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 Section 1.3.

`/OUTPUT/` contains information on output level and I/O units, see Section 1.5.

`/ERRCTL/` The variable `IER` contains the current error indicator, `ICHECK` is used to control consistency checks and tracing of the subprograms, see Section 1.5.

`/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 parameters 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`, `DY`, and `DZ`. `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. `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 element $[-1,1]^3$ 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.

/MGPAR/ contains information about the number of levels ILEV, NLEV, NLMIN and NLMAX. ICYCLE denotes the type of multigrid cycle (V,W or F cycle are possible) and KPOSM resp. KPRSM contain the information about the number of pre– and postsmoothing steps.

/MGTRA/ and /MGTRD/ are the multigrid equivalents of the COMMON blocks /TRIAA/ and /TRIAD/.

/MGTIME/ saves the information about the used time for the different multigrid components.

2.5. SAVE statement

At least the names of all COMMON blocks occuring 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 FEAT3D and some essential FEAT2D routines. Concerning the important group of routines with starting letters `Y`, and `Z` and most of the routines with starting letters `X` we refer to the user manual of FEAT2D. Here, only some major points are listed in Section 3.2.

3.1. FEAT3D Subprograms

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

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

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. Matrices involving boundary integrals are not yet implemented in FEAT3D.

In the present version of FEAT3D 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 1.4, 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 `ABvs` assemble the entries of the global matrices, where `AB..` is used for hexahedral elements. The character `v` denotes the version, `v=0` for area integrals. The final character `s` is used to reference the storage technique for the matrix.

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 a parameter. 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 reserve the remaining free space of DWORK to calculate the column pointer KCOL. After completion, the free space of DWORK is released. The user may speed up the calculation of KCOL by proposing a number NEROW for the (estimated) average number of nonzero entries per row of the matrix. NEROW should be carefully chosen.

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)
NEROW	I*4	Estimated maximum number of nonzero elements per row. For NEROW=0, the default value 27 is chosen in AP7 and AP9.
KVERT	I*4	DIMENSION KVERT(NNVE,NEL) Numbers of vertices of elements
KEDGE	I*4	DIMENSION KEDGE(NNEE,NEL) Numbers of edges of elements, if necessary
KAREA	I*4	DIMENSION KAREA(NNAE,NEL) Numbers of faces 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

Parameters in `COMMON` blocks

```
      COMMON /TRIAD/  NEL,NVT,NET,NAT,NVE,NEE,NAE,NVEL,NEEL,NVED,
     *                NVAR,NEAR,NBCT,NVBD,NEBD,NABD
      COMMON /TRIAA/  LCORVG,LCORMG,LCORAG,LVERT,LEDGE,LAREA,LADJ,
     *                LVEL,LEEL,LAEL,LVED,LAED,LVAR,LEAR,LEVE,LAVE,
     *                LNPR,LBCT,LVBD,LEBD,LABD
```

Information about the subdivision. `/TRIAA/` is only needed in `XAP3`, `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,KEDGE,KAREA,
     *                KDIAH)

      SUBROUTINE XAP7(LCOL,LLD,NA,NEQ,ELE,ISYMM,NEROW)
      SUBROUTINE  AP7(KCOL,KLD,NA,NEQ,ELE,ISYMM,NEROW,KVERT,KEDGE,KAREA)

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

Subgroup AB – Assembly of matrices

The subprograms ABvs assemble global matrices for meshes consisting of hexahedra. 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 a parameter 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 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 and trial functions in the array KAB(2,NNAB,NBLOC). Consider, for example, the bilinear form

$$\int_\Omega (\partial_x\phi_i\partial_x\phi_j + \partial_y\phi_i\partial_y\phi_j + \partial_z\phi_i\partial_z\phi_j + \beta_1\phi_i\partial_x\phi_j + \beta_2\phi_i\partial_y\phi_j + \beta_3\phi_i\partial_z\phi_j)\, dx\,.$$

Here, β_i are some coefficients. The correponding value for the array KABN is 6 since six 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	First z-derivative
5	Second x-derivative
6	Mixed xy-derivative
7	Mixed xz-derivative
8	Second y-derivative
9	Mixed yz-derivative
10	Second z-derivative

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

2	2
3	3
4	4
1	2
1	3
1	4

for the six 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 a parameter

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

Parameters Input

X,Y,Z	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,Z)`. 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.

Example

This example models the non-trivial case of a coefficient function that depends on given finite element basis function representation - i.e. a function that is not given analytically but only as coefficient vector. The according bilinear form (arising in the context of finite element solution of Navier-Stokes equations) reads

$$b_u(\phi_i,\phi_j) = ((u\cdot\nabla)\phi_i,\phi_j) = \int_\Omega (u_1\partial_x\phi_i\phi_j + u_2\partial_y\phi_i\phi_j + u_3\partial_z\phi_i\phi_j)\,dx\,,$$

with given data $u=(u_1,u_2,u_3)$. Of course, the user has to provide the information on u. In our example this is realized with `COMMON /COAUXN/`, providing the location of the coefficient vectors on `DWORK`. The function, then, is evaluated via a `DBAS` representation.

```
      DOUBLE PRECISION FUNCTION COEFFN(X,Y,Z,IA,IB,IDA,BFIRST)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      PARAMETER (NNARR=299,NNVE=8,NNDIM=3,NNBAS=21,NNDER=10)
      DIMENSION VWORK(1),KWORK(1)
      COMMON /COAUXN/ KLU1,KLU2,KLU3
      COMMON /ELEM/   DX(NNVE),DY(NNVE),DZ(NNVE),DJAC(3,3),DETJ,
     *                DBAS(NNDIM,NNBAS,NNDER),BDER(NNDER),KVE(NNVE),
     *                IEL,NDIM
      COMMON /COAUX1/ KDFG(NNBAS),KDFL(NNBAS),IDFL
      COMMON          NWORK,IWORK,IWMAX,L(NNARR),DWORK(1)
      EQUIVALENCE (DWORK(1),VWORK(1),KWORK(1))
      SAVE /COAUXN/,/ELEM/,/COAUX1/
C
      IF (IB.EQ.2) THEN
       U1=0D0
       DO 1 JDOFE=1,IDFL
       IEQ=KDFG(JDOFE)
       ILO=KDFL(JDOFE)
       U1=U1+DWORK(L(KLU1)+IEQ-1)*DBAS(1,ILO,1)
1      CONTINUE
       COEFFN=U1
      ELSE IF (IB.EQ.3) THEN
       U2=0D0
       DO 2 JDOFE=1,IDFL
       IEQ=KDFG(JDOFE)
       ILO=KDFL(JDOFE)
       U2=U2+DWORK(L(KLU2)+IEQ-1)*DBAS(1,ILO,1)
2      CONTINUE
       COEFFN=U2
      ELSE IF (IB.EQ.4) THEN
       U3=0D0
       DO 3 JDOFE=1,IDFL
       IEQ=KDFG(JDOFE)
       ILO=KDFL(JDOFE)
       U3=U3+DWORK(L(KLU3)+IEQ-1)*DBAS(1,ILO,1)
3      CONTINUE
       COEFFN=U3
      ELSE IF ((IA.EQ.-1).AND.(IB.EQ.-1).AND.(IDA.EQ.0)) THEN
       BDER(1)=.TRUE.
      ELSE
       WRITE(*,*) '*** ERROR *** COEFFN : IA,IB,IDA ',IA,IB,IDA
       STOP
      ENDIF
C
      END
```

Parameters Input

LA	I*4	DIMENSION LA(NBLOC) Handles of block matrices (for X-routines only)
KDIA	I*4	DIMENSION KDIA(NDIA) Diagonal offset pointer, generated by AP3
KDIAS	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
KCOL	I*4	DIMENSION KCOL(NA) Column pointer, generated by AP7 and AP9
KLD	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) Numbers of vertices of elements
KEDGE	I*4	DIMENSION KEDGE(NNEE,NEL) Numbers of edges of elements, if necessary
KAREA	I*4	DIMENSION KAREA(NNAE,NEL) Numbers of faces of elements, if necessary
DCORVG	R*8	DIMENSION DCORVG(3,NVT) Coordinates of vertices
DCORMG	R*8	DIMENSION DCORMG(3,NET) Coordinates of midpoints of edges, if necassary
DCORAG	R*8	DIMENSION DCORAG(3,NAT) Coordinates of midpoints of faces, if necassary
KNPR	I*4	DIMENSION KNPR(NVT) Nodal properties, see Section 1.3
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 (in COEFF)
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
ILINT	I*4	=0 Full trilinear transformation to reference element necessary =1 Only linear transformation needed =2 Axiparallel grid
BSNGL	B*4	=.TRUE. Change matrix type to SINGLE PRECISION
ARR	C*6	DIMENSION ARR(NBLOC) Names of block matrices, for messages only

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,NET,NAT,NVE,NEE,NAE,NVEL,NEEL,NVED,
     *                NVAR,NEAR,NBCT,NVBD,NEBD,NABD
      COMMON /TRIAA/  LCORVG,LCORMG,LCORAG,LVERT,LEDGE,LAREA,LADJ,
     *                LVEL,LEEL,LAEL,LVED,LAED,LVAR,LEAR,LEVE,LAVE
     *                LNPR,LBCT,LVBD,LEBD,LABD
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
```

Exchange of information with COEFF

```
      PARAMETER (NNBAS=27,NNDER=10,NNDIM=3)
      COMMON /COAUX1/ KDFG(NNBAS),KDFL(NNBAS),IDFL
      COMMON /COAUX2/ DBAS1(NNDIM,NNBAS,NNDER,3),KDFG1(NNBAS,3),
     *                KDFL1(NNBAS,3),IDFL1(3),BDER1(NNDER,3)
```

List of available subprograms

```
      SUBROUTINE XAB03(LA,LDIA,LDIAS,NDIA,NA,NEQ,NBLOC,ICLEAR,ELE,
     *                 COEFF,BCON,KAB,KABN,ICUB,ISYMM,ILINT,BSNGL,ARR)
      SUBROUTINE  AB03(DA,KDIA,KDIAS,NDIA,NA,NEQ,NBLOC,KOFF,
     *                 KVERT,KEDGE,KAREA,DCORVG,ELE,COEFF,BCON,COECON,
     *                 KAB,KABN,ICUB,ISYMM,ILINT)
      SUBROUTINE XAB07(LA,LCOL,LLD,NA,NEQ,NBLOC,ICLEAR,ELE,
     *                 COEFF,BCON,KAB,KABN,ICUB,ISYMM,ILINT,BSNGL,ARR)
      SUBROUTINE  AB07(DA,KCOL,KLD,NA,NEQ,NBLOC,KOFF,KVERT,KEDGE,KAREA,
     *                 DCORVG,ELE,COEFF,BCON,COECON,KAB,KABN,ICUB,ISYMM,
     *                 ILINT)
      SUBROUTINE XAB09(LA,LCOL,LLD,NA,NEQ,NBLOC,ICLEAR,ELE1,ELE2,ELE3,
```

```
*                 COEFF,BCON,KAB,KABN,ICUB,ILINT,BSNGL,ARR)
 SUBROUTINE  AB09(DA,KCOL,KLD,NA,NEQ,NBLOC,KOFF,KVERT,KEDGE,KAREA,
*                 DCORVG,ELE1,ELE2,ELE3,COEFF,BCON,COECON,KAB,KABN,
*                 ICUB,ILINT)
```

Group C – Numerical integration

The subprograms of this group return information on numerical integration rules in three space dimensions (see, for example, Stroud [6]). The only parameter is `ICUB`, the number of the integration rule. The number and position of the cubature points as well as the weights are returned in the corresponding arrays in `COMMON /CUB/`.

<u>Parameters in `COMMON` blocks</u>

```
PARAMETER (NNCUBP=36)

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

<u>List of available subprograms</u>

```
SUBROUTINE CB3H(ICUB)
```

Integration formulas for the reference cube $[-1,1]^3$.
`DXI(ICUBP,I), I=1,2,3, ICUBP=1,NCUBP` returns the position of the integration points in cartesian coordinates.

<u>List of available cubature formulas</u>

ICUB	NCUBP	Degree	Comment
1	1	1	Gaussian formula
2	6	1	Midpoints of areas
3	8	1	Trapezoidal rule
4	12	1	Midpoints of edges
5	4	2	Stroud formula
6	6	3	Stroud formula
7	8	3	Gaussian formula
8	14	5	Hammer & Stroud formula
9	27	5	Gaussian formula
10	34	7	Sarma & Stroud formula

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 COMMON /ELEM/.

In the present version, FEAT3D supports only hexahedral elements. The transformation to the reference cube is assumed to be trilinear. The parameter list is as follows

```
      SUBROUTINE Ennn(XI1,XI2,XI3,IPAR)
```

Parameters Input

XIn	R*8	n=1,2,3, Cartesian coordinates of the evaluation point in the reference element $[-1,1]^3$
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 information 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.

<u>Parameters in COMMON blocks</u>

```
      PARAMETER (NNBAS=27,NNDER=10,NNVE=8,NNCUBP=36,NNDIM=3)

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

List of available elements

Name	IELTYP	# d.o.f.	Comment
E010	10	1	constant
E011	11	8	trilinear, continuous
E013	13	27	triquadratic, continuous
E030	30	6	rotated trilinear, discontinuous mean values on element faces as nodal values
E031	31	6	rotated trilinear, discontinuous function values at midpoints of faces as nodal values

Group G – Normalized output for graphics packages

This group contains subroutines writing grid information or user supplied data in the special format that is needed by certain graphic packages. The first one which is supported is the MOVIE.BYU graphic package [7]. The subroutine `XGOWSM` writes FEAT3D 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*15	File name

Output

CFILE	C*15	File containing grid or function data in MOVIE.BYU format

The other graphic package which is supported in FEAT3D is the `ADVANCE VISUALIZATION SYSTEM`, (AVS), [8].

```
      SUBROUTINE XAVSUC(NUNIT,CFILE,NEL,NVT,KVERT,DCORVG,
                        NCOMP,NLANG,COMLAB,UNILAB,LU)
```

Parameters Input

NUNIT	I*4	Unit number
CFILE	C*15	File name
NEL	I*4	Total number of elements
NVT	I*4	Total number of vertices
KVERT	I*4	Number of vertices
DCORVG	R*8	Coordinates of the vertices
NCOMP	I*4	Number of components (one possible component could be the velocity)
NLANG	I*4	Number of subcomponents (is set to 3 for the velocity in the 3D–case)
COMLAB	C*15	Label for each component
UNILAB	C*15	Unit for component (e.g. m/s for velocity)
LU	R*8	Solution vector

Output

CFILE	C**	File containing grid or function data in AVS format

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 FEAT3D-package.

First, we recall the list of the multigrid related COMMON-blocks, see also section (1.3). 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 1.3)

```
      PARAMETER (NNLEV=9)
      COMMON /MGTRD/  KNEL(NNLEV),KNVT(NNLEV),KNET(NNLEV),
     *                KNAT(NNLEV),KNVE(NNLEV),KNEE(NNLEV),
     *                KNAE(NNLEV),KNVEL(NNLEV),KNEEL(NNLEV),
     *                KNVED(NNLEV),KNVAR(NNLEV),KNEAR(NNLEV),
     *                KNBCT(NNLEV),KNVBD(NNLEV),KNEBD(NNLEV),
     *                KNABD(NNLEV)
      COMMON /MGTRA/  KLCVG(NNLEV),KLCMG(NNLEV),KLCAG(NNLEV),
     *                KLVERT(NNLEV),KLEDGE(NNLEV),KLAREA(NNLEV),
     *                KLADJ(NNLEV),KLVEL(NNLEV),KLEEL(NNLEV),
     *                KLAEL(NNLEV),KLVED(NNLEV),KLAED(NNLEV),
     *                KLVAR(NNLEV),KLEAR(NNLEV),KLEVE(NNLEV),
     *                KLAVE(NNLEV),KLNPR(NNLEV),KLBCT(NNLEV),
     *                KLVBD(NNLEV),KLEBD(NNLEV),KLABD(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.

ICYCLE   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 XMSB2(ISCAD,ISE,ISA,ISVEL,ISEEL,ISAEL,
     *                 ISVED,ISAED,ISVAR,ISEAR,ISEVE,ISAVE,
     *                 ISVBD,ISEBD,ISABD,IDISP,PARX,PARY,PARZ,
     *                 SEDB,SADB)
      SUBROUTINE XMSCL
```

Explanation

The routine XMSB2 generates 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 XMAP7 (KLCOL,KLLD,KNA,KNEQ,ELE,ISYMM,NROW)
      SUBROUTINE XMAP9 (KLCOL,KLLD,KNA,KNEQ,ELE1,ELE2)
      SUBROUTINE XMAB07(KLA,KLCOL,KLLD,KNA,KNEQ,NBLCA,ICLR,ELE,
     *                  COEFF,BCON,KAB,KABN,ICUB,ISYMM,ILINT,BSNGL,CARR)
      SUBROUTINE XMAB09(KLA,KLCOL,KLLD,KNA,KNEQ,NBLOC,ICLEAR,ELE1,ELE2,
     *                  ELE3,COEFF,BCON,KAB,KABN,ICUB,ILINT,BSNGL,CARR)
```

Multilevel drivers

The standard multigrid driver for the F-, V-, and W-cycle is provided by the subprogram M011. 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  M011 (DX,DB,DD,KOFFX,KOFFB,KOFFD,KNEQ,NIT,ITE,EPS,EPSU,
     *                  EPSP,DAX,DPROL,DREST,DPRSM,DPOSM,DEX,DBC,DSTEP,
     *                  KITO,KIT,IREL,RHOLMG)
```

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
DSTEP	SUBR	DSTEP(DX,DD,DB,DSTEPP) Returns DSTEPP := optimal step size for correction
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
RHOLMG	R*8	Multigrid convergence rate

Prolongations - restrictions

In the present version only standard prolongations and restrictions for the finite elements E011, E030 and E031 (see 3.1) are provided. 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 dimension of the region and is in our case 3. Finally, nn is the number of the element.

```
SUBROUTINE MP311(DX1,DX2,KVERT1,KVERT2,KADJ1,KADJ2,NVT1,NVT2,NEL1,NEL2)
SUBROUTINE MP330(DX1,DX2,KAREA1,KAREA2,KADJ1,KADJ2,NAT1,NAT2,NEL1,NEL2)
SUBROUTINE MP331(DX1,DX2,KAREA1,KAREA2,KADJ1,KADJ2,NAT1,NAT2,NEL1,NEL2)
SUBROUTINE MR311(DF2,DF1,KVERT2,KVERT1,KADJ2,KADJ1,NVT2,NVT1,NEL2,NEL1)
SUBROUTINE MR330(DF2,DF1,KAREA2,KAREA1,KADJ2,KADJ1,NAT2,NAT1,NEL2,NEL1)
SUBROUTINE MR331(DF2,DF1,KAREA2,KAREA1,KADJ2,KADJ1,NAT2,NAT1,NEL2,NEL1)
```

Explanation

The routines assume standard coarsening or standard refinement, respectively. The prolongation programs return a fine grid vector DX2 from a coarse grid vector DX1. Analogously, the restriction routines calculate a coarse grid vector DX1. 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 YPROL(DX1,DX2)
SUBROUTINE YREST(DF2,DF1)
```

Explanation

These subprogram names are passed as EXTERNAL arguments to the driver routine M011.As above, 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 (see 3.1), XOWA and XORA.

```
SUBROUTINE XMOWS (MFILE,CCFILE,IFMT)
SUBROUTINE XMORS (MFILE,CCFILE,IFMT)
SUBROUTINE XMORA7(KLA,KLCOL,KLLD,MFILE,CCFILE,IFMT)
SUBROUTINE XMORA9(KLA,KLCOL,KLLD,NBLOC,MFILE,CCFILE,IFMT)
SUBROUTINE XMOWA7(KLA,KLCOL,KLLD,MFILE,CCFILE,IFMT)
SUBROUTINE XMOWA9(KLA,KLCOL,KLLD,NBLOC,MFILE,CCFILE,IFMT)
```

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 FEAT3D-format. `XMOWAn` and `XMORAn` write or read the matrices for all levels to/from files in FEAT3D-format corresponding to the routines `XMAB0n`.

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,NET,NAT,NVE,NEE,NAE,NVEL,NEEL,NVED,
     *                NVAR,NEAR,NBCT,NVBD,NEBD,NABD
```

List of available subprograms

```
      INTEGER FUNCTION NDFL(IELTYP)
      INTEGER FUNCTION NDFG(IELTYP)
      SUBROUTINE NDFGL(IEL,IPAR,IELTYP,KVERT,KEDGE,KAREA,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 elements
KEDGE	I*4	DIMENSION KEDGE(NNEE,NEL) Numbers of edges of elements, if necessary
KAREA	I*4	DIMENSION KAREA(NNAE,NEL) Numbers of midpoints of faces, 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 O – Input/Output

Input/Output of subdivisions

This subgroup serves for storing and reading of whole subdivisions of the domain. These subroutines use the FEAT2D I/O routines to read and write the information of a whole subdivision of the domain in normalized form. 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 1.3. The parameters MFILE, CFILE, and IFMT are used as above.

Triangulations usually are generated automatically, frequently starting from a coarse initial subdivision. To read coarse mesh information from 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 1.3.

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

Contents of the input file for XORSC:

* - FORMAT used for input
NET and NAT usually are 0.

Comment line
Comment line
NEL NVT NBCT NVE NEE NAE
Comment line
((DCORVG(IDIM,IVT),IDIM=1,3),IVT=1,NVT)
Comment line
((KVERT(IVE,IEL),IVE=1,NVE),IEL=1,NEL)
Comment line
(KNPR(IVT),IVT=1,NVT)

Example

Unit cube with a hole – $\bar{\Omega} = [0,1]^3 \setminus (0.3, 0.7)^3$

```
Coarse grid TRIA2
Unit cube with a hole
      6 16 2 8 12 6    NEL NVT NBCT NVE NEE NAE
DCORVG
      0D0     0D0     0D0
      1D0     0D0     0D0
      1D0     1D0     0D0
      0D0     1D0     0D0
    0.3D0   0.3D0   0.3D0
    0.7D0   0.3D0   0.3D0
    0.7D0   0.7D0   0.3D0
    0.3D0   0.7D0   0.3D0
    0.3D0   0.3D0   0.7D0
    0.7D0   0.3D0   0.7D0
    0.7D0   0.7D0   0.7D0
    0.3D0   0.7D0   0.7D0
      0D0     0D0     1D0
      1D0     0D0     1D0
      1D0     1D0     1D0
      0D0     1D0     1D0
KVERT
      1  2  3  4  5  6  7  8
     13 14 15 16  9 10 11 12
      1  2  6  5 13 14 10  9
      2  3  7  6 14 15 11 10
      3  4  8  7 15 16 12 11
      4  1  5  8 16 13  9 12
KNPR
      1 1 1 1 2 2 2 2 2 2 2 2 1 1 1 1
```

Names of the subprograms

The following programs are used to read and write complete subdivisions 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 S – Generation of subdivisions

The programs of this group are devoted to the generation and modification of subdivisions. In the present version of FEAT3D only few routines for mesh generation are available. 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 (see Section 3.1). Alternatively, meshes that are constructed from graphics programs such as MOVIE.BYU or AVS can be used in FEAT3D (see Section 3.1).

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 SB0 (XSB0), for hexahedral meshes.

For a detailed description of the following parameters see Section 1.3.

Parameters Input

DCORVG	R*8	DIMENSION DCORVG(3,NVT) Coordinates of vertices
DCORMG	R*8	DIMENSION DCORMG(3,NET) Coordinates of midpoints of edges, if necessary
DCORAG	R*8	DIMENSION DCORAG(3,NAT) Coordinates of midpoints of faces, if necessary
KVERT	I*4	DIMENSION KVERT(NNVE,NEL) Numbers of vertices of elements
KEDGE	I*4	DIMENSION KEDGE(NNEE,NEL) Numbers of edges of elements, if necessary
KAREA	I*4	DIMENSION KAREA(NNAE,NEL) Numbers of midpoints of faces, if necessary
KADJ	I*4	DIMENSION KADJ(NNAE,NEL) Numbers of adjacent elements
KNPR	I*4	DIMENSION KNPR(NVT) Nodal properties

Output

KVEL	I*4	DIMENSION KVEL(NVEL,NVT) Numbers of elements meeting at a vertex
KEEL	I*4	DIMENSION KEEL(NEEL,NMT) Numbers of elements meeting at an edge
KAEL	I*4	DIMENSION KAEL(2,NAT) Numbers of elements meeting at a face
KVED	I*4	DIMENSION KVED(2,NET) Numbers of vertices meeting at an edge
KAED	I*4	DIMENSION KAED(NEAR,NAT) Numbers of faces meeting at an edge
KVAR	I*4	DIMENSION KVAR(4,NAT) Numbers of vertices meeting at a face
KEAR	I*4	DIMENSION KEAR(2,NAT)

		Numbers of edges meeting at a face
KEVE	I*4	DIMENSION KEVE(NVEL,NVT)
		Numbers of elements meeting at a vertex
KAVE	I*4	DIMENSION KAVE(NVAR,NVT)
		Numbers of faces meeting at a vertex
KVBD	I*4	DIMENSION KVBD(NVBD)
		Vertices on the boundary
KEBD	I*4	DIMENSION KEBD(NEBD)
		Edges on the boundary
KABD	I*4	DIMENSION KABD(NABD)
		Areas on the boundary
KBCT	I*4	DIMENSION KBCT(NBCT+1)
		Pointer vector for KVBD and KEBD

Parameters in COMMON blocks

```
      COMMON /TRIAD/  NEL,NVT,NET,NAT,NVE,NEE,NAE,NVEL,NEEL,NVED,
     *                NVAR,NEAR,NBCT,NVBD,NEBD,NABD
      COMMON /TRIAA/  LCORVG,LCORMG,LCORAG,LVERT,LEDGE,LAREA,LADJ,
     *                LVEL,LEEL,LAEL,LVED,LAED,LVAR,LEAR,LEVE,LAVE
     *                LNPR,LBCT,LVBD,LEBD,LABD
```

Further parameters will be explained together with the specific routines.

The central routine XSBOX

For creating a regularly refined mesh and allocating all arrays on the workspace vector the user may invoke the subprogram XSBOX. The second letter X denotes that these routines again only invoke the X-routines XSBCA and XSBO. All elements are subdivided into eight subelements. All edges are subdivided and connected. The subelement containing the "first" vertex of the original element keeps the old number, the other seven subelements are enumbered as described above. The old vertices keep their number. The new vertices are enumbered in the following way: First the new vertices on edges of the old subdivision, then the new vertices in the center of the areas of the old subdivision and, finally, the new interior vertices.

Parameters Input

NFINE	I*4	Desired number of regular subdivisions of the given mesh
ISCAD	I*4	=1: Determine array KADJ from coarse grid
ISE	I*4	=1: Determine numbers of midpoints
ISA	I*4	=0: Release array KADJ on return after determination of the new subdivision on finest mesh
ISEEL	I*4	=1: Determine numbers of elements meeting at each edge (array KEEL)
ISAEL	I*4	=1: Determine numbers of elements meeting at each face (array KAEL)
ISVEL	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
PARX	SUBR	Subroutines for
PARY	SUBR	the parameterization

PARZ	SUBR	of the domain
SEBD	SUBR	Subroutines for the control
SABD	SUBR	of the refinement

The remaining parameters (ISVED, ISAED, ISAED, ISVAR, ISEAR, ISEVE, ISAVE, ISVBD, ISEBD, ISABD) are provided to build up special information for the refined mesh, in the present version they are all set to 0.

```
      SUBROUTINE XSBOX(NFINE,ISCAD,ISE,ISA,ISVEL,ISEEL,ISAEL,ISVED,ISAED,
     *                 ISVAR,ISEAR,ISEVE,ISAVE,ISVBD,ISEBD,ISABD,IDISP,
     *                 PARX,PARY,PARZ,SEDB,SADB)
```

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

Generation of adjacent element information (KADJ) from coarse grid

```
      SUBROUTINE XSBCA(IDISP)
      SUBROUTINE  SBCA(KVERT,KADJ)
```

Generation of uniform subdivision (makes use of SBVEL, SBE, SBEEL, SBA and SBAEL)

```
      SUBROUTINE XSBO(NFINE,ISE,ISA,ISVEL,ISEEL,ISAEL,IDISP)
      SUBROUTINE SBO(DCORVG,DCOREG,KVERT,KADJ,KEDGE,KNPR,KVEL,NFINE,
     *               NNEL,NNVT,PARX,PARY,PARZ,SEDB,SADB)
```

Determine KVEL and NVEL

```
      SUBROUTINE SBVEL(KVERT,KVEL,IPAR)
```

Determination of KEDGE and NET

```
      SUBROUTINE SBE(KVERT,KVEL,KEDGE)
```

Determination of KEEL and NEEL

```
      SUBROUTINE SBEEL(KEDGE,KEEL,IPAR)
```

Determination of KAREA and NAT

```
      SUBROUTINE SBA(KADJ,KAREA)
```

Determination of KAEL

```
      SUBROUTINE SBAEL(KAREA,KAEL)
```

Generation of vertex element connectivity (KVEL)

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

Adjust dimensions of DCORVG, KVERT, KEDGE, KAREA, KADJ and KNPR

```
      SUBROUTINE SBV(DCORVG,KVERT,KEDGE,KADJ,KNPR)
```

Determination of KNPR from MOVIE.BYU grid

```
      SUBROUTINE XSBCB
      SUBROUTINE  SBCB(KVERT,KADJ,KVEL,KNPR)
```

Group V – Linear forms

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

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

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, the number of additive terms and the abbreviations for the partial derivatives applied to the basis functions, are 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,Z,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 VBv, where the character v denotes the version, v is set to 0 for volume 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) Numbers of vertices of elements
KEDGE	I*4	DIMENSION KEDGE(NNEE,NEL) Numbers of edges of elements, if necessary
KAREA	I*4	DIMENSION KAREA(NNAE,NEL) Numbers of faces of elements, if necessary
DCORVG	R*8	DIMENSION DCORVG(3,NVT) Coordinates of vertices
KNPR	I*4	DIMENSION KNPR(NVT) Nodal properties, see Section 1.3
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
ILINT	I*4	=0 Full trilinear transformation to reference element necessary =1 Only linear transformation needed =2 Axiparallel grid
BSNGL	B*4	=.TRUE. Change vector type to SINGLE PRECISION
ARR	C*6	DIMENSION ARR(NBLOC) Names of block vectors, for messages only

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,NET,NAT,NVE,NEE,NAE,NVEL,NEEL,NVED,
     *                NVAR,NEAR,NBCT,NVBD,NEBD,NABD
      COMMON /TRIAA/  LCORVG,LCORMG,LCORAG,LVERT,LEDGE,LAREA,LADJ,
     *                LVEL,LEEL,LAEL,LVED,LAED,LVAR,LEAR,LEVE,LAVE
     *                LNPR,LBCT,LVBD,LEBD,LABD
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
```

Exchange of information with COEFF

```
      PARAMETER (NNBAS=27,NNDER=10,NNDIM=3)

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

List of available subprograms

```
      SUBROUTINE XVBO(LB,NEQ,NBLOC,ICLEAR,ELE,COEFF,BCON,KB,KBN,
     *                ICUB,ILINT,BSNGL,ARR)
      SUBROUTINE  VBO(DB,NBLOC,KOFF,KVERT,KEDGE,KAREA,DCORVG,
     *                ELE,COEFF,BCON,COECON,KB,KBN,ICUB,ILINT)
```

Group Z – Handling of the pseudo-dynamic memory management

The 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 C. In particular, the following DATA statements initialize the blocks /OUTPUT/ and /ERRCTL/.

```
      BLOCK DATA ZVALUE
C
      IMPLICIT DOUBLE PRECISION (A,C-H,O-U,W-Z),LOGICAL(B)
      PARAMETER (NNARR=299)
      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
      COMMON /TRIAD/  NEL,NVT,NET,NAT,NVE,NEE,NAE,NVEL,NEEL,NVED,
     *                NVAR,NEAR,NBCT,NVBD,NEBD,NABD
      COMMON /TRIAA/  LCORVG,LCORMG,LCORAG,LVERT,LEDGE,LAREA,LADJ,
     *                LVEL,LEEL,LAEL,LVED,LAED,LVAR,LEAR,LEVE,LAVE,
     *                LNPR,LBCT,LVBD,LEBD,LABD
      COMMON /TABLE/  KTYPE(NNARR),KLEN(NNARR),KLEN8(NNARR),IFLAG
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/'(3D24.16)','(5E14.7)','(6I12)'/
      DATA CPARAM/'                                                   '/
      DATA NEL/0/,NVT/0/,NET/0/,NAT/0/,NVE/0/,NEE/0/,NAE/0/,NVEL/0/,
     *     NEEL/0/,NVED/0/,NVAR/0/,NEAR/0/,NBCT/0/,NVBD/0/,NEBD/0/,
     *     NABD/0/
      DATA LCORVG/0/,LCORMG/0/,LCORAG/0/,LVERT/0/,LEDGE/0/,LAREA/0/,
     *     LADJ/0/,LVEL/0/,LEEL/0/,LAEL/0/,LVED/0/,LAED/0/,LVAR/0/,
     *     LEAR/0/,LEVE/0/,LAVE/0/,LNPR/0/,LBCT/0/,LVBD/0/,LEBD/0/,
     *     LABD/0/
      DATA KTYPE/NNARR*0/,KLEN/NNARR*0/,KLEN8/NNARR*0/,IFLAG/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 values 11-14 are used as default values for the I/O files for FEAT3D. IRECL8 denotes the maximum record length for format-free I/O (machine dependent). 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.

3.2. FEAT2D Subprograms

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 (scaling, SSOR-preconditioning, ILU),

2 Smoothing (i.e. performing of 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),

`I` (Preconditioned) BICGSTAB algorithm (BICGSTAB),

`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 1.4.

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 parameters.

For algorithms that are used as smoothers or preconditioners only the corresponding `Y`-routines exist. For a complete list of subroutines in group `I` see FEAT2D manual or Appendix B.

Group L – Elementary linear algebra

This group is devoted to the basic linear algebra tasks. We distinguish between two subgoups, namely vector operations and matrix-vector operations. The routines of the first subgroup internally use BLAS routines (see Section 1.6) 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.

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 parameters.

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)
```

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)
```

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)
```

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)
```

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)
```

Scaling of a vector

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

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)
```

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)
```

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` matrix, `REAL` vectors, 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 1.4.

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)
```

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,NEQ,DX,DTX,A1,A2)
SUBROUTINE LTX1A(DA,KCOL,KLD,KOP,NEQ,DX,DTX,A1,A2)
```

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)
```

Group O – Input/Output

The programs of group O 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.

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	Name 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` 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.

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 W – Error handling

The programs of group W are intended for handling of errors occuring in FEAT3D 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 FEAT3D 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/.

Group Z – Handling of the pseudo-dynamic memory management Machine dependent system routines

1. Pseudodynamic memory management

The first subgroup of subprograms has been described in detail in the FEAT2D manual 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. 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.

Bibliography

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

[2] Blum, H., Harig, J., Müller, S.: *FEAT2D, Finite Element Analysis Tools, User Manual, Release 1.1*, Univ. Heidelberg (1990)

[3] Blum, H., Harig, J., Müller, S., Turek, S.: *FEAT2D, Finite Element Analysis Tools, User Manual, Release 1.3*, Univ. Heidelberg (1992)

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

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

[6] Stroud, A.H.: *Approximate Calculation of Multiple Integrals*, Prentice–Hall, Englewood Cliffs, New Jersey 1971

[7] *MOVIE.BYU User Manual*, January 1987 Edition, Version 6.2, Engineering Computer Graphics Laboratory, Brigham Young University, Provo, Utah

[8] *AVS User Manual*, January 1994 Edition, Version 5, Vistec, Wiesbaden

A. List of Feat3d subprograms

Routine	Filename	Short description	Page
AB03	AB03.F	Bilinear form, dbl./sgl. precision, hexahedra	25
AB07	AB07.F	Bilinear form, dbl./sgl. precision, hexahedra	25
AB09	AB09.F	Bilinear form, dbl./sgl. precision, hexahedra	25
AP3	AP3.F	Pointer vectors for bilin. forms, storage techn. 3	23
AP7	AP7.F	Pointer vectors for bilin. forms, storage techn. 7	23
AP9	AP9.F	Pointer vectors for bilin. forms, storage techn. 9	23
CB3H	CB3H.F	3–dimensional cubature formulas, hexahedra	31
E010	E010.F	Element, constant, hexahedral	32
E011	E011.F	Element, trilinear, hexahedral	32
E011A	E011.F	Auxiliary routine for E011	32
E013	E013.F	Element, triquadratic, hexahedral	32
E013A	E013.F	Auxiliary routine for E011	32
E030	E030.F	Element, rotated trilinear, hexahedral	32
E030A	E030.F	Auxiliary routine for E030	32
E031	E031.F	Element, rotated trilinear, hexahedral	32
E031A	E031.F	Auxiliary routine for E031	32
GORSM	GORSM.F	Read subdivision in Movie.byu format	34
GOWSM	GOWSM.F	Write subdivision in Movie.byu format	34
M011	MO11.F	Multigrid solver, double precision	35
MP311	MP311.F	Multigrid prolongation for element E011	35
MP330	MP330.F	Multigrid prolongation for element E030	35
MP331	MP331.F	Multigrid prolongation for element E031	35
MR311	MR311.F	Multigrid restriction for element E011	35
MR330	MR330.F	Multigrid restriction for element E030	35
MR331	MR331.F	Multigrid restriction for element E031	35
NDFG	NDFG.F	Global number of d.o.f.	41
NDFGL	NDFGL.F	Relation global–local number of d.o.f.	41
NDFL	NDFL.F	Local numer of d.o.f.	41
NGLS	NDFGL.F	Auxiliary sorting routine for NDFGL	41
ORSC	ORSC.F	Read coarse grid	42
SB0	SB0.F	Regular refinement of a given hexahedral subdivision	44
SBA	SBA.F	Adjust dimensions of KAREA and NAT	44
SBAEL	SBAEL.F	Determination of KAEL	44
SBCA	SBCA.F	Determination of KADJ from coarse grid	44
SBCB	SBCB.F	Determination of KNPR from Movie.byu grid	44

B. List of Feat2d subprograms used in Feat3d

Routine	Filename	Short description
I000	I000.F	NOP (Dummy subroutine)
IA013	IA01.F	Jacobi-method, solver, precision (D/D), stor. techn. 3
IA017	IA01.F	Jacobi-method, solver, precision (D/D), stor. techn. 7
IA01A	IA01.F	Jacobi-method, solver, precision (D/D), stor. techn. A
IA023	IA02.F	Jacobi-method, solver, precision (S/S), stor. techn. 3
IA027	IA02.F	Jacobi-method, solver, precision (S/S), stor. techn. 7
IA02A	IA02.F	Jacobi-method, solver, precision (S/S), stor. techn. A
IA033	IA03.F	Jacobi-method, solver, precision (S/D), stor. techn. 3
IA037	IA03.F	Jacobi-method, solver, precision (S/D), stor. techn. 7
IA03A	IA03.F	Jacobi-method, solver, precision (S/D), stor. techn. A
IA113	IA11.F	Jacobi-method, precond., precision (D/D), storage techn. 3
IA117	IA11.F	Jacobi-method, precond., precision (D/D), storage techn. 7
IA11A	IA11.F	Jacobi-method, precond., precision (D/D), storage techn. A
IA123	IA12.F	Jacobi-method, precond., precision (S/S), storage techn. 3
IA127	IA12.F	Jacobi-method, precond., precision (S/S), storage techn. 7
IA12A	IA12.F	Jacobi-method, precond., precision (S/S), storage techn. A
IA133	IA13.F	Jacobi-method, precond., precision (S/D), storage techn. 3
IA137	IA13.F	Jacobi-method, precond., precision (S/D), storage techn. 7
IA13A	IA13.F	Jacobi-method, precond., precision (S/D), storage techn. A
IA213	IA21.F	Jacobi-method, smooth., precision (D/D), storage techn. 3
IA217	IA21.F	Jacobi-method, smooth., precision (D/D), storage techn. 7
IA21A	IA21.F	Jacobi-method, smooth., precision (D/D), storage techn. A
IA223	IA22.F	Jacobi-method, smooth., precision (S/S), storage techn. 3
IA227	IA22.F	Jacobi-method, smooth., precision (S/S), storage techn. 7
IA22A	IA22.F	Jacobi-method, smooth., precision (S/S), storage techn. A
IA233	IA23.F	Jacobi-method, smooth., precision (S/D), storage techn. 3
IA237	IA23.F	Jacobi-method, smooth., precision (S/D), storage techn. 7
IA23A	IA23.F	Jacobi-method, smooth., precision (S/D), storage techn. A
IB017	IB01.F	G-S-method, solver, precision (D/D), storage techn. 7
IB01A	IB01.F	G-S-method, solver, precision (D/D), storage techn. A
IB027	IB02.F	G-S-method, solver, precision (S/S), storage techn. 7
IB02A	IB02.F	G-S-method, solver, precision (S/S), storage techn. A
IB037	IB03.F	G-S-method, solver, precision (S/D), storage techn. 7
IB03A	IB03.F	G-S-method, solver, precision (S/D), storage techn. A
IB217	IB21.F	G-S-method, smooth., precision (D/D), storage techn. 7

Routine	Filename	Short description
IB21A	IB21.F	G-S-method, smooth., precision (D/D), storage techn. A
IB227	IB22.F	G-S-method, smooth., precision (S/S), storage techn. 7
IB22A	IB22.F	G-S-method, smooth., precision (S/S), storage techn. A
IB237	IB23.F	G-S-method, smooth., precision (S/D), storage techn. 7
IB23A	IB23.F	G-S-method, smooth., precision (S/D), storage techn. A
IC017	IC01.F	SOR-method, solver, precision (D/D), storage techn. 7
IC01A	IC01.F	SOR-method, solver, precision (D/D), storage techn. A
IC027	IC02.F	SOR-method, solver, precision (S/S), storage techn. 7
IC02A	IC02.F	SOR-method, solver, precision (S/S), storage techn. A
IC037	IC03.F	SOR-method, solver, precision (S/D), storage techn. 7
IC03A	IC03.F	SOR-method, solver, precision (S/D), storage techn. A
IC217	IC21.F	SOR-method, smooth., precision (D/D), storage techn. 7
IC21A	IC21.F	SOR-method, smooth., precision (D/D), storage techn. A
IC227	IC22.F	SOR-method, smooth., precision (S/S), storage techn. 7
IC22A	IC22.F	SOR-method, smooth., precision (S/S), storage techn. A
IC237	IC23.F	SOR-method, smooth., precision (S/D), storage techn. 7
IC23A	IC23.F	SOR-method, smooth., precision (S/D), storage techn. A
ID117	ID11.F	SSOR-method, precond., precision (D/D), storage techn. 7
ID118	ID11.F	SSOR-method, precond., precision (D/D), storage techn. 8
ID11A	ID11.F	SSOR-method, precond., precision (D/D), storage techn. A
ID127	ID12.F	SSOR-method, precond., precision (S/S), storage techn. 7
ID128	ID12.F	SSOR-method, precond., precision (S/S), storage techn. 8
ID12A	ID12.F	SSOR-method, precond., precision (S/S), storage techn. A
ID137	ID13.F	SSOR-method, precond., precision (S/D), storage techn. 7
ID138	ID13.F	SSOR-method, precond., precision (S/D), storage techn. 8
ID13A	ID13.F	SSOR-method, precond., precision (S/D), storage techn. A
ID217	ID21.F	SSOR-method, smooth., precision (D/D), storage techn. 7
ID218	ID21.F	SSOR-method, smooth., precision (D/D), storage techn. 8
ID21A	ID21.F	SSOR-method, smooth., precision (D/D), storage techn. A
ID227	ID22.F	SSOR-method, smooth., precision (S/S), storage techn. 7
ID228	ID22.F	SSOR-method, smooth., precision (S/S), storage techn. 8
ID22A	ID22.F	SSOR-method, smooth., precision (S/S), storage techn. A
ID237	ID23.F	SSOR-method, smooth., precision (S/D), storage techn. 7
ID238	ID23.F	SSOR-method, smooth., precision (S/D), storage techn. 8
ID23A	ID23.F	SSOR-method, smooth., precision (S/D), storage techn. A
IE010	IE010.F	(preconditioned) CG-method, solver, precision (D/D)
IE013	IE01.F	(precond.) CG-method, solver, prec. (D/D), storage techn. 3
IE014	IE01.F	(precond.) CG-method, solver, prec. (D/D), storage techn. 4
IE017	IE01.F	(precond.) CG-method, solver, prec. (D/D), storage techn. 7
IE018	IE01.F	(precond.) CG-method, solver, prec. (D/D), storage techn. 8
IE01A	IE01.F	(precond.) CG-method, solver, prec. (D/D), storage techn. A
IE020	IE020.F	(preconditioned) CG-method, solver, single precision
IE023	IE02.F	(precond.) CG-method, solver, prec. (S/S), storage techn. 3

Routine	Filename	Short description
IE024	IE02.F	(precond.) CG-method, solver, prec. (S/S), storage techn. 4
IE027	IE02.F	(precond.) CG-method, solver, prec. (S/S), storage techn. 7
IE028	IE02.F	(precond.) CG-method, solver, prec. (S/S), storage techn. 8
IE02A	IE02.F	(precond.) CG-method, solver, prec. (S/S), storage techn. A
IE033	IE03.F	(precond.) CG-method, solver, prec. (S/D), storage techn. 3
IE034	IE03.F	(precond.) CG-method, solver, prec. (S/D), storage techn. 4
IE037	IE03.F	(precond.) CG-method, solver, prec. (S/D), storage techn. 7
IE038	IE03.F	(precond.) CG-method, solver, prec. (S/D), storage techn. 8
IE03A	IE03.F	(precond.) CG-method, solver, prec. (S/D), storage techn. A
IF117	IF11.F	ILU decomp., precond., precision (D/D), storage techn. 7
IF127	IF21.F	ILU decomp., precond., precision (S/S), storage techn. 7
IF137	IF31.F	ILU decomp., precond., precision (S/D), storage techn. 7
IFD17	IFD1.F	Calculate ILU decomp., precision (D), storage techn. 7
IFD27	IFD2.F	Calculate ILU decomp., precision (S), storage techn. 7
LAX13	LAX1.F	Matrix-vector-mult., precision (D/D), storage techn. 3
LAX14	LAX1.F	Matrix-vector-mult., precision (D/D), storage techn. 4
LAX17	LAX1.F	Matrix-vector-mult., precision (D/D), storage techn. 7
LAX18	LAX1.F	Matrix-vector-mult., precision (D/D), storage techn. 8
LAX19	LAX1.F	Matrix-vector-mult., precision (D/D), storage techn. 9
LAX1A	LAX1.F	Matrix-vector-mult., precision (D/D), storage techn. A
LAX23	LAX2.F	Matrix-vector-mult., precision (S/S), storage techn. 3
LAX24	LAX2.F	Matrix-vector-mult., precision (S/S), storage techn. 4
LAX27	LAX2.F	Matrix-vector-mult., precision (S/S), storage techn. 7
LAX28	LAX2.F	Matrix-vector-mult., precision (S/S), storage techn. 8
LAX29	LAX2.F	Matrix-vector-mult., precision (S/S), storage techn. 9
LAX2A	LAX2.F	Matrix-vector-mult., precision (S/S), storage techn. A
LAX33	LAX3.F	Matrix-vector-mult., precision (S/D), storage techn. 3
LAX34	LAX3.F	Matrix-vector-mult., precision (S/D), storage techn. 4
LAX37	LAX3.F	Matrix-vector-mult., precision (S/D), storage techn. 7
LAX38	LAX3.F	Matrix-vector-mult., precision (S/D), storage techn. 8
LAX39	LAX3.F	Matrix-vector-mult., precision (S/D), storage techn. 9
LAX3A	LAX3.F	Matrix-vector-mult., precision (S/D), storage techn. A
LCL1	LCL1.F	Clear vector, double precision
LCL2	LCL2.F	Clear vector, single precision
LCL3	LCL3.F	Clear vector, integer
LCP1	LCP1.F	Copy vector, double precision
LCP2	LCP2.F	Copy vector, single precision
LCP3	LCP3.F	Copy vector, integer
LL21	LL21.F	l^2-norm of vector, double precision
LL22	LL22.F	l^2-norm of vector, single precision
LLC1	LLC1.F	Linear combination of two vectors, double precision
LLC2	LLC2.F	Linear combination of two vectors, single precision
LLI1	LLI1.F	Maximum-norm of vector, double precision

Routine	Filename	Short description
LLI2	LLI2.F	Maximum-norm of vector, single precision
LSC1	LSC1.F	Scaling of vector, double precision
LSC2	LSC2.F	Scaling of vector, single precision
LSP1	LSP1.F	Scalar product of two vectors, double precision
LSP2	LSP2.F	Scalar product of two vectors, single precision
LTX13	LTX1.F	Transp. matrix-vector-mult., precision (D/D), storage techn. 3
LTX17	LTX1.F	Transp. matrix-vector-mult., precision (D/D), storage techn. 7
LTX19	LTX1.F	Transp. matrix-vector-mult., precision (D/D), storage techn. 9
LTX1A	LTX1.F	Transp. matrix-vector-mult., precision (D/D), storage techn. A
LTX23	LTX2.F	Transp. matrix-vector-mult., precision (S/S), storage techn. 3
LTX27	LTX2.F	Transp. matrix-vector-mult., precision (S/S), storage techn. 7
LTX29	LTX2.F	Transp. matrix-vector-mult., precision (S/S), storage techn. 9
LTX2A	LTX2.F	Transp. matrix-vector-mult., precision (S/S), storage techn. A
LTX33	LTX3.F	Transp. matrix-vector-mult., precision (S/D), storage techn. 3
LTX37	LTX3.F	Transp. matrix-vector-mult., precision (S/D), storage techn. 7
LTX39	LTX3.F	Transp. matrix-vector-mult., precision (S/D), storage techn. 9
LTX3A	LTX3.F	Transp. matrix-vector-mult., precision (S/D), storage techn. A
LVM1	LVM.F	Vector multiply and add, precision (D/D)
LVM2	LVM.F	Vector multiply and add, precision (S/S)
LVM3	LVM.F	Vector multiply and add, precision (S/D)
LWS13	LWS1.F	Weighted scalar product, precision (D/D), storage techn. 3
LWS14	LWS1.F	Weighted scalar product, precision (D/D), storage techn. 4
LWS17	LWS1.F	Weighted scalar product, precision (D/D), storage techn. 7
LWS18	LWS1.F	Weighted scalar product, precision (D/D), storage techn. 8
LWS19	LWS1.F	Weighted scalar product, precision (D/D), storage techn. 9
LWS1A	LWS1.F	Weighted scalar product, precision (D/D), storage techn. A
LWS23	LWS2.F	Weighted scalar product, precision (S/S), storage techn. 3
LWS24	LWS2.F	Weighted scalar product, precision (S/S), storage techn. 4
LWS27	LWS2.F	Weighted scalar product, precision (S/S), storage techn. 7
LWS28	LWS2.F	Weighted scalar product, precision (S/S), storage techn. 8
LWS29	LWS2.F	Weighted scalar product, precision (S/S), storage techn. 9
LWS2A	LWS2.F	Weighted scalar product, precision (S/S), storage techn. A
LWS33	LWS3.F	Weighted scalar product, precision (S/D), storage techn. 3
LWS34	LWS3.F	Weighted scalar product, precision (S/D), storage techn. 4
LWS37	LWS3.F	Weighted scalar product, precision (S/D), storage techn. 7
LWS38	LWS3.F	Weighted scalar product, precision (S/D), storage techn. 8
LWS39	LWS3.F	Weighted scalar product, precision (S/D), storage techn. 9
LWS3A	LWS3.F	Weighted scalar product, precision (S/D), storage techn. A
OERR	OERR.F	Write error messages
OF0	OF0.F	Open file
OMSG	OMSG.F	Write messsages and notes
ORA0	XORA.F	Auxiliary routine for ORAn, XORA
ORA1	ORA.F	Read array (normalized), double precision

Routine	Filename	Short description
ORA2	ORA.F	Read array (normalized), single precision
ORA3	ORA.F	Read array (normalized), integer
OTRC	OTRC.F	Trace programs
OWA0	XOWA.F	Auxiliary routine for OWAn, XOWA
OWA1	OWA.F	Write array (normalized), double precision
OWA2	OWA.F	Write array (normalized), single precision
OWA3	OWA.F	Write array (normalized), integer
RC13	RC13.F	Compress (block) matrices, storage techn. 3, double precision
RC17	RC17.F	Compress (block) matrices, storage techn. 7, double precision
RC19	RC19.F	Compress (block) matrices, storage techn. 9, double precision
RC23	RC23.F	Compress (block) matrices, storage techn. 3, single precision
RC27	RC27.F	Compress (block) matrices, storage techn. 7, single precision
RC29	RC29.F	Compress (block) matrices, storage techn. 9, single precision
WERR	WERR.F	Basic error handling
XIC017	XIC01.F	SOR-method, solver, storage techn. 7
XIC01A	XIC01.F	SOR-method, solver, storage techn. A
XIC027	XIC02.F	SOR-method, solver, storage techn. 7
XIC02A	XIC02.F	SOR-method, solver, storage techn. A
XIE013	XIE01.F	CG-method, solver, storage techn. 3
XIE014	XIE01.F	CG-method, solver, storage techn. 4
XIE017	XIE01.F	CG-method, solver, storage techn. 7
XIE018	XIE01.F	CG-method, solver, storage techn. 8
XIE01A	XIE01.F	CG-method, solver, storage techn. A
XIE023	XIE02.F	CG-method, solver, storage techn. 3
XIE024	XIE02.F	CG-method, solver, storage techn. 4
XIE027	XIE02.F	CG-method, solver, storage techn. 7
XIE028	XIE02.F	CG-method, solver, storage techn. 8
XIE02A	XIE02.F	CG-method, solver, storage techn. A
XIE033	XIE03.F	CG-method, solver, storage techn. 3
XIE034	XIE03.F	CG-method, solver, storage techn. 4
XIE037	XIE03.F	CG-method, solver, storage techn. 7
XIE038	XIE03.F	CG-method, solver, storage techn. 8
XIE03A	XIE03.F	CG-method, solver, storage techn. A
XORA	XORA.F	Read array (normalized)
XOWA	XOWA.F	Write array (normalized)
YIA113	YIA11.F	Jacobi-method, preconditioning by scaling
YIA117	YIA11.F	Jacobi-method, preconditioning by scaling
YIA11A	YIA11.F	Jacobi-method, preconditioning by scaling
YIA123	YIA12.F	Jacobi-method, preconditioning by scaling
YIA127	YIA12.F	Jacobi-method, preconditioning by scaling
YIA12A	YIA12.F	Jacobi-method, preconditioning by scaling
YIA133	YIA13.F	Jacobi-method, preconditioning by scaling
YIA137	YIA13.F	Jacobi-method, preconditioning by scaling

Routine	Filename	Short description
YIA13A	YIA13.F	Jacobi-method, preconditioning by scaling
YID117	YID11.F	SSOR-method, preconditioning
YID118	YID11.F	SSOR-method, preconditioning
YID11A	YID11.F	SSOR-method, preconditioning
YID127	YID12.F	SSOR-method, preconditioning
YID128	YID12.F	SSOR-method, preconditioning
YID12A	YID12.F	SSOR-method, preconditioning
YID137	YID13.F	SSOR-method, preconditioning
YID138	YID13.F	SSOR-method, preconditioning
YID13A	YID13.F	SSOR-method, preconditioning
YLAX13	YLAX1.F	Matrix-vector-multiplication
YLAX14	YLAX1.F	Matrix-vector-multiplication
YLAX17	YLAX1.F	Matrix-vector-multiplication
YLAX18	YLAX1.F	Matrix-vector-multiplication
YLAX19	YLAX1.F	Matrix-vector-multiplication
YLAX1A	YLAX1.F	Matrix-vector-multiplication
YLAX23	YLAX2.F	Matrix-vector-multiplication
YLAX24	YLAX2.F	Matrix-vector-multiplication
YLAX27	YLAX2.F	Matrix-vector-multiplication
YLAX28	YLAX2.F	Matrix-vector-multiplication
YLAX29	YLAX2.F	Matrix-vector-multiplication
YLAX2A	YLAX2.F	Matrix-vector-multiplication
YLAX33	YLAX3.F	Matrix-vector-multiplication
YLAX34	YLAX3.F	Matrix-vector-multiplication
YLAX37	YLAX3.F	Matrix-vector-multiplication
YLAX38	YLAX3.F	Matrix-vector-multiplication
YLAX39	YLAX3.F	Matrix-vector-multiplication
YLAX3A	YLAX3.F	Matrix-vector-multiplication
YLTX13	YLTX1.F	Matrix-vector-multiplication, transposed matrix
YLTX17	YLTX1.F	Matrix-vector-multiplication, transposed matrix
YLTX19	YLTX1.F	Matrix-vector-multiplication, transposed matrix
YLTX1A	YLTX1.F	Matrix-vector-multiplication, transposed matrix
YLTX23	YLTX2.F	Matrix-vector-multiplication, transposed matrix
YLTX27	YLTX2.F	Matrix-vector-multiplication, transposed matrix
YLTX29	YLTX2.F	Matrix-vector-multiplication, transposed matrix
YLTX2A	YLTX2.F	Matrix-vector-multiplication, transposed matrix
YLTX33	YLTX3.F	Matrix-vector-multiplication, transposed matrix
YLTX37	YLTX3.F	Matrix-vector-multiplication, transposed matrix
YLTX39	YLTX3.F	Matrix-vector-multiplication, transposed matrix
YLTX3A	YLTX3.F	Matrix-vector-multiplication, transposed matrix
ZCLEAR	ZCLEAR.F	Clear vector on workspace
ZCPY	ZCPY.F	Copy vector on workspace
ZCTYPE	ZCTYPE.F	Change data type of vector on workspace

Routine	Filename	Short description
ZDISP	ZDISP.F	Shorten or delete vector on workspace
ZFREE	ZFREE.F	Calculate free space on workspace
ZINIT	ZINIT.F	General initialization
ZLEN	ZLEN.F	Return length of vector on workspace
ZLEN8	ZLEN8.F	Return length of vector on workspace in double words
ZNEW	ZNEW.F	Allocate vector on workspace
ZTIME	ZTIME.F	Return system time since last call
ZTYPE	ZTYPE.F	Return data type of vector on workspace

C. List of COMMON blocks

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

      PARAMETER (NNARR=299,NNAB=21,NNDER=10)
      PARAMETER (NNBAS=27,NNCUBP=36,NNVE=8,NNDIM=3)

      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,LCORAG,LVERT,LEDGE,LAREA,LADJ,
     *                LVEL,LEEL,LAEL,LVED,LAED,LVAR,LEAR,LEVE,LAVE
     *                LNPR,LBCT,LVBD,LEBD,LABD
      COMMON /TRIAD/  NEL,NVT,NET,NAT,NVE,NEE,NAE,NVEL,NEEL,NVED,
     *                NVAR,NEAR,NBCT,NVBD,NEBD,NABD
      COMMON /ELEM/   DX(NNVE),DY(NNVE),DZ(NNVE),DJAC(3,3),DETJ,
     *                DBAS(NNDIM,NNBAS,NNDER),BDER(NNDER),KVE(NNVE),
     *                IEL,NDIM
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
      COMMON /COAUX1/ KDFG(NNBAS),KDFL(NNBAS),IDFL
      COMMON /COAUX2/ DBAS1(NNDIM,NNBAS,NNDER,3),KDFG1(NNBAS,3),
     *                KDFL1(NNBAS,3),IDFL1(3),BDER1(NNDER,3)
      COMMON /TABLE/  KTYPE(NNARR),KLEN(NNARR),KLEN8(NNARR),IFLAG

C *** COMMON blocks for multigrid data management

      COMMON /MGPAR/  ILEV,NLEV,NLMIN,NLMAX,
     *                ICYCLE,KPRSM(NNLEV),KPOSM(NNLEV)
      COMMON /MGTRD/  KNEL(NNLEV),KNVT(NNLEV),KNET(NNLEV),
     *                KNAT(NNLEV),KNVE(NNLEV),KNEE(NNLEV),
     *                KNAE(NNLEV),KNVEL(NNLEV),KNEEL(NNLEV),
     *                KNVED(NNLEV),KNVAR(NNLEV),KNEAR(NNLEV),
     *                KNBCT(NNLEV),KNVBD(NNLEV),KNEBD(NNLEV),
     *                KNABD(NNLEV)
      COMMON /MGTRA/  KLCVG(NNLEV),KLCMG(NNLEV),KLCAG(NNLEV),
     *                KLVERT(NNLEV),KLEDGE(NNLEV),KLAREA(NNLEV),
     *                KLADJ(NNLEV),KLVEL(NNLEV),KLEEL(NNLEV),
     *                KLAEL(NNLEV),KLVED(NNLEV),KLAED(NNLEV),
     *                KLVAR(NNLEV),KLEAR(NNLEV),KLEVE(NNLEV),
```

```
*                KLAVE(NNLEV),KLNPR(NNLEV),KLBCT(NNLEV),
*                KLVBD(NNLEV),KLEBD(NNLEV),KLABD(NNLEV)
 COMMON /MGTIME/ TTMG,TTS,TTE,TTD,TTP,TTR,IMTIME
```

D. 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'
```

E. Sample Program

```
      PROGRAM SAMPLE
C
C *** Sample program
C *** Demonstration of FEAT3D, multigrid solver
C *** conforming finite elements
C
C *** Problem  -2u  - u   - u   + u = f
C                 xx   yy    zz
C
C *** Exact solution  u = X**2+Y**2+Z**2
C *** Inhomogeneneous Dirichlet boundary conditions implemented
C
C *** Trilinear hexahedral element
C
C *** Domain: Unit cube
C
C *** Standard declarations and parameter settings
C
      IMPLICIT DOUBLE PRECISION (A,C-H,O-U,W-Z),LOGICAL(B)
      PARAMETER (NNARR=299,NNLEV=9,NNWORK=6500000)
      PARAMETER (NNVE=8,NNDIM=3,NNAB=21,NNBAS=27,NNDER=10)
C
C *** NBLCA=1 means that the stiffness matrix only consists of one
C *** block, analogously NBLCF stands for the right hand side
C
      PARAMETER (NBLCA=1,NBLCF=1)
      PARAMETER (NBLA1=NBLCA*NNLEV,NBLF1=NBLCF*NNLEV)
      CHARACTER FMT*15,CFILE*15,SUB*6,CPARAM*120
      CHARACTER ARRDA*6,ARRDF*6
      CHARACTER CARRDA*6,CARRDF*6
      CHARACTER CCFILE*15
      DIMENSION ARRDA(NBLCA),ARRDF(NBLCF)
      DIMENSION CARRDA(NBLCA,NNLEV),CARRDF(NBLCF,NNLEV)
      DIMENSION CCFILE(NNLEV)
C
C *** KABA  - structure of the stiffness matrix
C *** KABAN - number of terms in the integrand for the stiffness matrix
C *** KBF   - structure of the right hand side
C *** KBFN  - number of terms in the integrand for the right hand side
```

```
      DIMENSION LA(NBLCA),KABAN(NBLCA),KABA(2,NNAB,NBLCA),BCONA(NBLCA)
      DIMENSION LF(NBLCF),KBFN(NBLCF),KBF(NNAB,NBLCF),BCONF(NBLCF)
      DIMENSION LU(NBLCA)
      DIMENSION VWORK(1),KWORK(1)
      DIMENSION KLA(NBLCA,NNLEV),KLF(NBLCF,NNLEV)
      DIMENSION KLCOLA(NNLEV),KLLDA(NNLEV),KNA(NNLEV),KNEQ(NNLEV)
      COMMON          NWORK,IWORK,IWMAX,L(NNARR),DWORK(NNWORK)
      COMMON /ERRCTL/ IER,ICHECK
      COMMON /CHAR/   SUB,FMT(3),CPARAM
      COMMON /TRIAD/  NEL,NVT,NET,NAT,NVE,NEE,NAE,NVEL,NEEL,NVED,
     *                NVAR,NEAR,NBCT,NVBD,NEBD,NABD
      COMMON /TRIAA/  LCORVG,LCORMG,LCORAG,LVERT,LEDGE,LAREA,LADJ,
     *                LVEL,LEEL,LAEL,LVED,LAED,LVAR,LEAR,LEVE,LAVE,
     *                LNPR,LBCT,LVBD,LEBD,LABD
      COMMON /ELEM/   DX(NNVE),DY(NNVE),DZ(NNVE),DJAC(3,3),DETJ,
     *                DBAS(NNDIM,NNBAS,NNDER),BDER(NNDER),KVE(NNVE),
     *                IEL,NDIM
      COMMON /OUTPUT/ M,MT,MKEYB,MTERM,MERR,MPROT,MSYS,MTRC,IRECL8
      COMMON /MGTRD/  KNEL(NNLEV),KNVT(NNLEV),KNET(NNLEV),
     *                KNAT(NNLEV),KNVE(NNLEV),KNEE(NNLEV),
     *                KNAE(NNLEV),KNVEL(NNLEV),KNEEL(NNLEV),
     *                KNVED(NNLEV),KNVAR(NNLEV),KNEAR(NNLEV),
     *                KNBCT(NNLEV),KNVBD(NNLEV),KNEBD(NNLEV),
     *                KNABD(NNLEV)
      COMMON /MGTRA/  KLCVG(NNLEV),KLCMG(NNLEV),KLCAG(NNLEV),
     *                KLVERT(NNLEV),KLEDGE(NNLEV),KLAREA(NNLEV),
     *                KLADJ(NNLEV),KLVEL(NNLEV),KLEEL(NNLEV),
     *                KLAEL(NNLEV),KLVED(NNLEV),KLAED(NNLEV),
     *                KLVAR(NNLEV),KLEAR(NNLEV),KLEVE(NNLEV),
     *                KLAVE(NNLEV),KLNPR(NNLEV),KLBCT(NNLEV),
     *                KLVBD(NNLEV),KLEBD(NNLEV),KLABD(NNLEV)
      COMMON /MGPAR/  ILEV,NLEV,NLMIN,NLMAX,
     *                ICYCLE,KPRSM(NNLEV),KPOSM(NNLEV)
      COMMON /MGPARO/ DOMPOS(NNLEV),DOMPRS(NNLEV),OMEX,EPSEX,
     *                KIPOSM(NNLEV),KIPRSM(NNLEV),IEX,NITEX,IELE
      COMMON /MGTIME/ TTMG,TTS,TTE,TTD,TTP,TTR,IMTIME
C
      EQUIVALENCE (DWORK(1),VWORK(1),KWORK(1))
C
      EXTERNAL E011,RHS,COEFFA,I000,U,UX,UY,UZ
      EXTERNAL PARX,PARY,PARZ,SEDB,SADB
      EXTERNAL YMAX17,YREST,YPROL,YPOSM,YPRSM,YEX,YMBC01
      DATA LA/0/,KABAN/4/,ARRDA/'DA    '/,BCONA/.TRUE./ ,BSNGLA/.FALSE./
      DATA LF/0/,KBFN /1/,ARRDF/'DF    '/,BCONF/.FALSE./,BSNGLF/.FALSE./
      DATA NROW/27/,ISYMMA/0/
      DATA ICLR0/0/,ICLR1/1/
      DATA KLCOLA/NNLEV*0/,KLLDA/NNLEV*0/
      DATA KLA/NBLA1*0/,KLF/NBLF1*0/
      DATA CCFILE/'TRIA.1','TRIA.2','TRIA.3','TRIA.4','TRIA.5',
```

```
     *            'TRIA.6','TRIA.7','TRIA.8','TRIA.9'/
      DATA CARRDA/'DA.1  ','DA.2  ','DA.3  ','DA.4  ','DA.5  ',
     *            'DA.6  ','DA.7  ','DA.8  ','DA.9  '/
      DATA CARRDF/'DF.1  ','DF.2  ','DF.3  ','DF.4  ','DF.5  ',
     *            'DF.6  ','DF.7  ','DF.8  ','DF.9  '/

C
C
      OPEN(18,FILE='FEAT.PRT')
      CLOSE(18,STATUS='DELETE')
C
C *** Initialization
C *** Default names are used for output devices
C
      CALL ZINIT(NNWORK,'feat3d.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
      KABA(1,3,1)=4
      KABA(2,3,1)=4
      KABA(1,4,1)=1
      KABA(2,4,1)=1
C
      KBF(1,1)=1
C
      CALL ZTIME(TIME1)
C *** Open data file for input
      MDATA=79
      OPEN (MDATA,FILE='TEST1.DAT')
      CALL GDAT(MDATA,CFILE,ICUBA,ICUBF,NIT,EPS,NPRSM,
     *          NPOSM,OMPOS,OMPRS,IPOSM,IPRSM,ICUBP)
C
C
C *** Read in coarse grid from file 'TRIAQ', opened as unit 55
      MUNITT=55
      CALL XORSC(MUNITT,CFILE)
      IF (IER.NE.0) GOTO 99999
C
C *** No use of divergence free elements is made
C
      NDIM=1
C

C
```

```
C
C *** Generate grid information from coarse grid data
C
      CALL XSBOX(0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,PARX,PARY,PARZ,
     *           SEDB,SADB)
C
C *** Refine coarse grid
C
C *** Construct a regular subdivision of the given coarse grid
C *** ISE=1   - Determine numbers of midpoints
C *** ISA=0   - Release array KADJ on return after determination of
C ***           the new subdivision on finest mesh
C *** ISVEL=0 - Numbers of elements meeting at each vertex can be
C ***           neglected
C *** ISEEL=0 - Numbers of elements meeting at each edge can be
C ***           neglected
C *** ISAEL=0 - Numbers of elements meeting at each face can be
C ***           neglected
C *** IDISPT=1- Release all unused arrays on DWORK on return
C
C
      ISE=0
      ISA=0
      ISVEL=0
      ISEEL=0
      ISAEL=0
      ISVED=0
      ISAED=0
      ISVAR=0
      ISEAR=0
      ISEVE=0
      ISAVE=0
      ISVBD=0
      ISEBD=0
      ISABD=0
      IDISPT=1
      CALL XMSB0(0,MAX(1,ISE),ISA,ISVEL,ISEEL,ISAEL,
     *           ISVED,ISAED,ISVAR,ISEAR,ISEVE,ISAVE,
     *           ISVBD,ISEBD,ISABD,IDISP,PARX,PARY,PARZ,
     *           SEDB,SADB)
       IF (IER.NE.0) GOTO 99999
      IF (INIT.EQ.1) THEN
       CALL XMOWS(69,CCFILE,1)
      ELSE
       CALL XMORS(69,CCFILE,1)
      ENDIF
C
C
C *** NET=0  - No information about edges is generated
```

```
C *** NAT=0  - No information about faces is generated
C *** NVEL=0 - No information available about maximum number of element
C ***          meeting at one vertex
C *** NEEL=0 - No information available about maximum number of element
C ***          meeting at one edge
C *** ISYMMA =0: No symmetry assumed, full matrix calculated
C ***        =1: Only upper triangular part calculated
C ***        =2: Symmetry assumed but full matrix calculated,
C ***            lower triangular part obtained by reflection
      DO 11 ILEV=NLMIN,NLMAX
      NET=0
      NAT=0
      KNAT(ILEV)=NAT
      KNET(ILEV)=NET
11    CONTINUE
      ISYMMA=0
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
C
      CALL XMAP7(KLCOLA,KLLDA,KNA,KNEQ,E011,ISYMMA,NROW)
      IF (IER.NE.0) GOTO 99999
C *** Allocate solution vector DU in the correct length NEQ, determined by XAP7
      NEQ=KNEQ(NLEV)
      CALL ZNEW(NEQ,1,LU(1),'DU    ')
      IF (IER.NE.0) GOTO 99999
C
C
C *** Calculation of the stiffness matrix
C *** Number is returned in LA(1)
C *** Number of cubature formula read in as a parameter
C
      ICLR=1
      CALL XMAB07(KLA,KLCOLA,KLLDA,KNA,KNEQ,NBLCA,ICLR,E011,
     *            COEFFA,BCONA,
     *            KABA,KABAN,ICUBA,ISYMMA,ILINT,BSNGLA,CARRDA)
      IF (IER.NE.0) GOTO 99999
      CALL XMOWA7(KLA,KLCOLA,KLLDA,69,CARRDA,1)
      IF (IER.NE.0) GOTO 99999
C
C *** Calculation of the right hand side vector
C *** Number is returnes in LF(1)
C *** Number of cubature formula read in as a parameter
C
      LF(1)=0
      CALL XVB0(LF(1),KNEQ(NLMAX),NBLCF,ICLR1,E011,RHS,
     *          BCONF,KBF,KBFN,
     *          ICUBF,ILINT,BSNGLF,CARRDF)
```

```
      IF (IER.NE.0) GOTO 99999
      CALL XOWA(LF(1),'DF    ',69,CARRDF(1,NLMAX),1)
      IF (IER.NE.0) GOTO 99999
C
       LU(1)=0
       CALL ZNEW(NEQ,1,LU(1),'DU    ')
       IF (IER.NE.0) GOTO 99999
C
C
C
C *** User provided program for implementation of boundary conditions
C *** Determine dof's where boundary conditions have to be prescribed
      DO 10 ILEV=NLMIN,NLMAX
      LA(1)=KLA(1,ILEV)
      NA   =KNA(ILEV)
      LCOLA=KLCOLA(ILEV)
      LLDA =KLLDA(ILEV)
      LNPR =KLNPR(ILEV)
      LCORVG =KLCVG(ILEV)
      NEQ =KNEQ(ILEV)
      NVT =KNVT(ILEV)
      LVBD=0
      CALL ZNEW(NVT,3,LVBD,'KVBD  ')
      IF (IER.NE.0) GOTO 99999
C
C
C
      CALL BDCD1(KWORK(L(LVBD)),KWORK(L(LNPR)))
      CALL ZDISP(NVBD,LVBD,'KVBD  ')
      IF (IER.NE.0) GOTO 99999
C
C *** User provided program for implementation of boundary conditions
C *** Adjust matrix rows
C
      CALL BDCD2(DWORK(L(LA(1))),KWORK(L(LLDA)),KWORK(L(LVBD)),NVBD)
C
C *** Elements with modulus smaller than 1D-15 are considered as zero
C *** and are deleted from matrix LA(1)), i.e. DA. The pointer vector
C *** LCOLA, i.e. the vector KCOLA is changed correspondingly
C
      CALL XRC17(LA(1),LCOLA,LLDA,NA,NEQ,NBLCA,1D-15,1,
     *           'DA    ' ,'KCOLA ')
      IF (IER.NE.0) GOTO 99999
C
C *** User provided program for implementation of boundary conditions
C
      CALL BDCD3(DWORK(L(LU(1))),DWORK(L(LF(1))),KWORK(L(LVBD)),NVBD,
     *           U,DWORK(L(LCORVG)))
      IF (IER.NE.0) GOTO 99999
```

```
10    CONTINUE
C
C *** Parameters for mg solution algorithm are read from data file
C *** NIT maximum number of iterations, EPS desired accuracy,
C *** OMEGA preconditioning parameter
C *** The total CPU time for the preparation of the linear system is
C *** displayed
      CALL ZTIME(TIME2)
      WRITE(MTERM,*) '* TEST1 * TIME PREPERATION ',TIME2-TIME1
C
C
C *** RBNORM is set to the L2-norm of the right hand side
      CALL XLL21(LF(1),KNEQ(NLMAX),RBNORM)
C *** EPS is normalized by the norm of the right hand side (rel. accuracy)
      EPS=EPS*RBNORM
C
      DO 500 ILEV=NLMIN,NLMAX
      KPRSM(ILEV)=NPRSM
      KPOSM(ILEV)=NPOSM
      DOMPOS(ILEV)=OMPOS
      DOMPRS(ILEV)=OMPRS
      KIPOSM(ILEV)=IPOSM
      KIPRSM(ILEV)=IPRSM
500   CONTINUE
C
      IMTIME=1
      CALL XM017(KLA,KLCOLA,KLLDA,LU,LF(1),KNEQ,NIT,ITE,EPS,
     *           YMAX17,YPROL,YREST,YPRSM,YPOSM,YEX,YMBC01,IDISP)
C
C
      WRITE(MTERM,*)'ANZAHL DER UNBEKANNTEN: ',NVT
      IF (IER.NE.0) GOTO 99999
      WRITE (*,*) 'TTMG,TTS,TTE,TTD,TTP,TTR'
      WRITE (*,*) TTMG,TTS,TTE,TTD,TTP,TTR
C
C *** The cpu time for the solution is displayed
C
      CALL ZTIME(TIME2)
      WRITE(MTERM,*) '* TEST1 * TIME SOLVER ',TIME2-TIME1
C
C *** A user provided subprogram is called to calculate the l2- and
C *** h1-error. The error is integrated with cubature formula ICUB
C      CALL ZTIME(TIME1)
C
C
      CALL PQL2U(DWORK(L(LU(1))),KWORK(L(LVERT)),KWORK(L(LEDGE)),
     *           KWORK(L(LAREA)),DWORK(L(LCORVG)),E011,ICUBP,ILINT,
     *           U,UX,UY,UZ)
      IF (IER.NE.0) GOTO 99999
```

```
C
C *** The solution vector is written to file CFILE (unit IFILE)
C *** XOWA uses the standard FORMAT contained in FMT(1), for DOUBLE
C *** PRECISION vectors
C
C
C *** The CPU time for the calculation of the error is displayed
C
      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 XGOWFM(LU,69,'DX.MOV')
      IF (IER.NE.0) GOTO 99999
      CALL GOWSM(70,'TRIA.MOV')
      IF (IER.NE.0) GOTO 99999
C
C
C
C *** NWORK contains the total number of DOUBLE PRECISION elements
C *** of DWORK
C *** IWMAX is the maximum number of elements of DWORK used in the
C *** current program
C
      WRITE(MTERM,*) '* TEST1 * NWORK ',NWORK
      WRITE(MTERM,*) '* TEST1 * IWMAX ',IWMAX
C
99999 WRITE(MTERM,*) '* TEST1 * IER', IER
      WRITE(MTERM,*) '* TEST1 * IN SUBROUTINE ',SUB
C
      END
C
C
C
      SUBROUTINE BDCD1(KVBD,KNPR)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      DIMENSION KVBD(*),KNPR(*)
      COMMON /TRIAD/  NEL,NVT,NET,NAT,NVE,NEE,NAE,NVEL,NEEL,NVED,
     *                NVAR,NEAR,NBCT,NVBD,NEBD,NABD
      SAVE
C
      NVBD=0
      DO 1 IVT=1,NVT
      IF (KNPR(IVT).NE.0) THEN
      NVBD=NVBD+1
      KVBD(NVBD)=IVT
      ENDIF
1     CONTINUE
```

```
C
      END
C
C
C
      SUBROUTINE BDCD2(DA,KLD,KVBD,NVBD)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      DIMENSION DA(*),KLD(*),KVBD(*)
C
      DO 3 IVT=1,NVBD
      IVBD=KVBD(IVT)
      DA(KLD(IVBD))=1D0
      DO 4 ICOL=KLD(IVBD)+1,KLD(IVBD+1)-1
4     DA(ICOL)=0D0
3     CONTINUE
C
      END
C
C
C
      SUBROUTINE BDCD3(DU,DF,KVBD,NVBD,U,DCORVG)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      DIMENSION DU(*),DF(*),KVBD(*)
      DIMENSION DCORVG(3,*)
      EXTERNAL U
C
      DO 1 IVT=1,NVBD
      IVBD=KVBD(IVT)
      PX=DCORVG(1,IVBD)
      PY=DCORVG(2,IVBD)
      PZ=DCORVG(3,IVBD)
      DU(IVBD)= U(PX,PY,PZ)
      DF(IVBD)= U(PX,PY,PZ)
1     CONTINUE
C
      END
C
C *** Coefficient function for the stiffness matrix
C
      DOUBLE PRECISION FUNCTION COEFFA(X,Y,Z,IA,IB,IDA,BFIRST)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      IF (IA.EQ.1) THEN
C *** Coefficient for the absolute term
       COEFFA= 1D0
      ELSE IF (IA.EQ.2) THEN
C *** Coefficient for the second x-derivative
       COEFFA=2D0
      ELSE IF (IA.EQ.3) THEN
C *** Coefficient for the second y-derivative
```

```
        COEFFA=1D0
       ELSE IF (IA.EQ.4) THEN
C *** Coefficient for the second z-derivative
        COEFFA=1D0
       ENDIF
C
      END
C
C *** Right hand side yielding the exact solution UE
C
      DOUBLE PRECISION FUNCTION RHS(X,Y,Z,IA,IDA,BFIRST)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      RHS=X**2+Y**2+Z**2-8D0
      END
C
C *** Exact solution and derivatives
C
      DOUBLE PRECISION FUNCTION U(X,Y,Z)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      U=X**2+Y**2+Z**2
      END
C
      DOUBLE PRECISION FUNCTION UX(X,Y,Z)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      UX=2D0*X
      END
C
      DOUBLE PRECISION FUNCTION UY(X,Y,Z)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      UY=2D0*Y
      END
C
      DOUBLE PRECISION FUNCTION UZ(X,Y,Z)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      UZ=2D0*Z
      END
C
C *** Error check routine
C
      SUBROUTINE PQL2U(DU,KVERT,KEDGE,KAREA,DCORVG,ELE,ICUB,ILINT,
     *                 U,UX,UY,UZ)
      IMPLICIT REAL*8 (A,C-H,O-U,W-Z),LOGICAL(B)
      PARAMETER (NNBAS=27,NNDER=10,NNCUBP=36,NNVE=8,NNEE=12,NNAE=6)
      PARAMETER (NNDIM=3,Q2=0.5D0,Q8=0.125D0)
      CHARACTER FMT*15,SUB*6,CPARAM*120
      DIMENSION DU(*)
      DIMENSION KVERT(NNVE,*),KEDGE(NNEE,*),KAREA(NNAE,*),DCORVG(3,*)
      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),DZ(NNVE),DJAC(3,3),DETJ,
     *                DBAS(NNDIM,NNBAS,NNDER),BDER(NNDER),KVE(NNVE),
     *                IEL,NDIM
      COMMON /TRIAD/  NEL,NVT,NET,NAT,NVE,NEE,NAE,NVEL,NEEL,NVED,
     *                NVAR,NEAR,NBCT,NVBD,NEBD,NABD
      COMMON /CUB/    DXI(NNCUBP,3),DOMEGA(NNCUBP),NCUBP,ICUBP
      COMMON /COAUX1/ KDFG,KDFL,IDFL
      SAVE /OUTPUT/,/ERRCTL/,/CHAR/,/ELEM/,/TRIAD/,/CUB/,/COAUX1/
C
      SUB='PQL2U'
      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 CB3H(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.
      BDER(4)=.TRUE.
C
      ERRL2=0D0
      ERRH1=0D0
      DNL2=0D0
      DNH1=0D0
C
      ICUBP=ICUB
      CALL ELE(0D0,0D0,0D0,-2)
      IF (ILINT.EQ.2) THEN
       DJAC(1,3)=0D0
       DJAC(2,3)=0D0
       DJAC(3,1)=0D0
       DJAC(3,2)=0D0
      ENDIF
C
      DO 100 IEL=1,NEL
C
      CALL NDFGL(IEL,1,IELTYP,KVERT,KEDGE,KAREA,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)
      DZ(IVE)=DCORVG(3,JP)
110   CONTINUE
C
      IF (ILINT.EQ.2) THEN
       DJ11=(DX(2)+DX(4))*Q2
       DJ12=(DY(2)+DY(4))*Q2
       DJ13=(DZ(1)+DZ(5))*Q2
       DJAC(1,1)=(-DX(1)+DX(2))*Q2
       DJAC(2,1)=(-DY(1)+DY(2))*Q2
       DJAC(1,2)=(-DX(1)+DX(4))*Q2
       DJAC(2,2)=(-DY(1)+DY(4))*Q2
       DJAC(3,3)=(-DZ(1)+DZ(5))*Q2
       DETJ=DJAC(3,3)*(DJAC(1,1)*DJAC(2,2)-DJAC(2,1)*DJAC(1,2))
      ELSE IF (ILINT.EQ.1) THEN
       DJ11=(DX(1)+DX(2)+DX(3)+DX(4)+DX(5)+DX(6)+DX(7)+DX(8))*Q8
       DJ12=(DY(1)+DY(2)+DY(3)+DY(4)+DY(5)+DY(6)+DY(7)+DY(8))*Q8
       DJ13=(DZ(1)+DZ(2)+DZ(3)+DZ(4)+DZ(5)+DZ(6)+DZ(7)+DZ(8))*Q8
       DJAC(1,1)=(-DX(1)+DX(2)+DX(3)-DX(4)-DX(5)+DX(6)+DX(7)-DX(8))*Q8
       DJAC(2,1)=(-DY(1)+DY(2)+DY(3)-DY(4)-DY(5)+DY(6)+DY(7)-DY(8))*Q8
       DJAC(3,1)=(-DZ(1)+DZ(2)+DZ(3)-DZ(4)-DZ(5)+DZ(6)+DZ(7)-DZ(8))*Q8
       DJAC(1,2)=(-DX(1)-DX(2)+DX(3)+DX(4)-DX(5)-DX(6)+DX(7)+DX(8))*Q8
       DJAC(2,2)=(-DY(1)-DY(2)+DY(3)+DY(4)-DY(5)-DY(6)+DY(7)+DY(8))*Q8
       DJAC(3,2)=(-DZ(1)-DZ(2)+DZ(3)+DZ(4)-DZ(5)-DZ(6)+DZ(7)+DZ(8))*Q8
       DJAC(1,3)=(-DX(1)-DX(2)-DX(3)-DX(4)+DX(5)+DX(6)+DX(7)+DX(8))*Q8
       DJAC(2,3)=(-DY(1)-DY(2)-DY(3)-DY(4)+DY(5)+DY(6)+DY(7)+DY(8))*Q8
       DJAC(3,3)=(-DZ(1)-DZ(2)-DZ(3)-DZ(4)+DZ(5)+DZ(6)+DZ(7)+DZ(8))*Q8
       DETJ= DJAC(1,1)*(DJAC(2,2)*DJAC(3,3)-DJAC(3,2)*DJAC(2,3))
     *      -DJAC(2,1)*(DJAC(1,2)*DJAC(3,3)-DJAC(3,2)*DJAC(1,3))
     *      +DJAC(3,1)*(DJAC(1,2)*DJAC(2,3)-DJAC(2,2)*DJAC(1,3))
      ELSE
       DJ11=( DX(1)+DX(2)+DX(3)+DX(4)+DX(5)+DX(6)+DX(7)+DX(8))*Q8
       DJ12=( DY(1)+DY(2)+DY(3)+DY(4)+DY(5)+DY(6)+DY(7)+DY(8))*Q8
       DJ13=( DZ(1)+DZ(2)+DZ(3)+DZ(4)+DZ(5)+DZ(6)+DZ(7)+DZ(8))*Q8
       DJ21=(-DX(1)+DX(2)+DX(3)-DX(4)-DX(5)+DX(6)+DX(7)-DX(8))*Q8
       DJ22=(-DY(1)+DY(2)+DY(3)-DY(4)-DY(5)+DY(6)+DY(7)-DY(8))*Q8
       DJ23=(-DZ(1)+DZ(2)+DZ(3)-DZ(4)-DZ(5)+DZ(6)+DZ(7)-DZ(8))*Q8
       DJ31=(-DX(1)-DX(2)+DX(3)+DX(4)-DX(5)-DX(6)+DX(7)+DX(8))*Q8
       DJ32=(-DY(1)-DY(2)+DY(3)+DY(4)-DY(5)-DY(6)+DY(7)+DY(8))*Q8
       DJ33=(-DZ(1)-DZ(2)+DZ(3)+DZ(4)-DZ(5)-DZ(6)+DZ(7)+DZ(8))*Q8
       DJ41=(-DX(1)-DX(2)-DX(3)-DX(4)+DX(5)+DX(6)+DX(7)+DX(8))*Q8
       DJ42=(-DY(1)-DY(2)-DY(3)-DY(4)+DY(5)+DY(6)+DY(7)+DY(8))*Q8
       DJ43=(-DZ(1)-DZ(2)-DZ(3)-DZ(4)+DZ(5)+DZ(6)+DZ(7)+DZ(8))*Q8
       DJ51=( DX(1)-DX(2)+DX(3)-DX(4)+DX(5)-DX(6)+DX(7)-DX(8))*Q8
       DJ52=( DY(1)-DY(2)+DY(3)-DY(4)+DY(5)-DY(6)+DY(7)-DY(8))*Q8
       DJ53=( DZ(1)-DZ(2)+DZ(3)-DZ(4)+DZ(5)-DZ(6)+DZ(7)-DZ(8))*Q8
       DJ61=( DX(1)-DX(2)-DX(3)+DX(4)-DX(5)+DX(6)+DX(7)-DX(8))*Q8
```

```
       DJ62=( DY(1)-DY(2)-DY(3)+DY(4)-DY(5)+DY(6)+DY(7)-DY(8))*Q8
       DJ63=( DZ(1)-DZ(2)-DZ(3)+DZ(4)-DZ(5)+DZ(6)+DZ(7)-DZ(8))*Q8
       DJ71=( DX(1)+DX(2)-DX(3)-DX(4)-DX(5)-DX(6)+DX(7)+DX(8))*Q8
       DJ72=( DY(1)+DY(2)-DY(3)-DY(4)-DY(5)-DY(6)+DY(7)+DY(8))*Q8
       DJ73=( DZ(1)+DZ(2)-DZ(3)-DZ(4)-DZ(5)-DZ(6)+DZ(7)+DZ(8))*Q8
       DJ81=(-DX(1)+DX(2)-DX(3)+DX(4)+DX(5)-DX(6)+DX(7)-DX(8))*Q8
       DJ82=(-DY(1)+DY(2)-DY(3)+DY(4)+DY(5)-DY(6)+DY(7)-DY(8))*Q8
       DJ83=(-DZ(1)+DZ(2)-DZ(3)+DZ(4)+DZ(5)-DZ(6)+DZ(7)-DZ(8))*Q8
      ENDIF
C
      DO 200 ICUBP=1,NCUBP
C
      XI1=DXI(ICUBP,1)
      XI2=DXI(ICUBP,2)
      XI3=DXI(ICUBP,3)
C
      IF (ILINT.EQ.0) THEN
       DJAC(1,1)=DJ21+DJ51*XI2+DJ61*XI3+DJ81*XI2*XI3
       DJAC(1,2)=DJ31+DJ51*XI1+DJ71*XI3+DJ81*XI1*XI3
       DJAC(1,3)=DJ41+DJ61*XI1+DJ71*XI2+DJ81*XI1*XI2
       DJAC(2,1)=DJ22+DJ52*XI2+DJ62*XI3+DJ82*XI2*XI3
       DJAC(2,2)=DJ32+DJ52*XI1+DJ72*XI3+DJ82*XI1*XI3
       DJAC(2,3)=DJ42+DJ62*XI1+DJ72*XI2+DJ82*XI1*XI2
       DJAC(3,1)=DJ23+DJ53*XI2+DJ63*XI3+DJ83*XI2*XI3
       DJAC(3,2)=DJ33+DJ53*XI1+DJ73*XI3+DJ83*XI1*XI3
       DJAC(3,3)=DJ43+DJ63*XI1+DJ73*XI2+DJ83*XI1*XI2
       DETJ= DJAC(1,1)*(DJAC(2,2)*DJAC(3,3)-DJAC(3,2)*DJAC(2,3))
     *      -DJAC(2,1)*(DJAC(1,2)*DJAC(3,3)-DJAC(3,2)*DJAC(1,3))
     *      +DJAC(3,1)*(DJAC(1,2)*DJAC(2,3)-DJAC(2,2)*DJAC(1,3))
      ENDIF
C      DETJ=ABS(DETJ)
      OM=DOMEGA(ICUBP)*ABS(DETJ)
C
      CALL ELE(XI1,XI2,XI3,-3)
      IF (IER.LT.0) GOTO 99999
C
      IF (ILINT.EQ.2) THEN
       XX= DJ11+DJAC(1,1)*XI1+DJAC(1,2)*XI2
       YY= DJ12+DJAC(2,1)*XI1+DJAC(2,2)*XI2
       ZZ= DJ13+DJAC(3,3)*XI3
      ELSE IF (ILINT.EQ.1) THEN
       XX= DJ11+DJAC(1,1)*XI1+DJAC(1,2)*XI2+DJAC(1,3)*XI3
       YY= DJ12+DJAC(2,1)*XI1+DJAC(2,2)*XI2+DJAC(2,3)*XI3
       ZZ= DJ13+DJAC(3,1)*XI1+DJAC(3,2)*XI2+DJAC(3,3)*XI3
      ELSE
       XX= DJ11+DJAC(1,1)*XI1+DJ31*XI2+DJ41*XI3+DJ71*XI2*XI3
       YY= DJ12+DJ22*XI1+DJAC(2,2)*XI2+DJ42*XI3+DJ62*XI1*XI3
       ZZ= DJ13+DJ23*XI1+DJ33*XI2+DJAC(3,3)*XI3+DJ53*XI1*XI2
      ENDIF
```

```
C
      UH =0D0
      UHX=0D0
      UHY=0D0
      UHZ=0D0
C
      DO 210 JDOFE=1,IDFL
      IEQ=KDFG(JDOFE)
      ILO=KDFL(JDOFE)
      UH =UH +DU(IEQ)*DBAS(1,ILO,1)
      UHX=UHX+DU(IEQ)*DBAS(1,ILO,2)
      UHY=UHY+DU(IEQ)*DBAS(1,ILO,3)
      UHZ=UHZ+DU(IEQ)*DBAS(1,ILO,4)
210   CONTINUE
C
      ERRL2=ERRL2+OM*(U(XX,YY,ZZ)-UH)**2
      ERRH1=ERRH1+OM*((UX(XX,YY,ZZ)-UHX)**2+
     *                (UY(XX,YY,ZZ)-UHY)**2+
     *                (UZ(XX,YY,ZZ)-UHZ)**2)
C      ERRH1=ERRH1+OM*((ABS(UX(XX,YY,ZZ))-ABS(UHX))**2+
C     *                (ABS(UY(XX,YY,ZZ))-ABS(UHY))**2+
C     *                (ABS(UZ(XX,YY,ZZ))-ABS(UHZ))**2)
      DNL2=DNL2+OM*U(XX,YY,ZZ)**2
      DNH1=DNH1+OM*(UX(XX,YY,ZZ)**2+UY(XX,YY,ZZ)**2+UZ(XX,YY,ZZ)**2)
C
200   CONTINUE
100   CONTINUE
C
      IF (MT.GT.0) THEN
       WRITE(MTERM,*)
       WRITE(MTERM,*) '* PQL2U * SOLUTION ELE     ',IELTYP
       WRITE(MTERM,*) '* PQL2U * CUBATURE FORMULA ',ICUB
       WRITE(MTERM,*) '* PQL2U * REL. L2-ERROR    ',SQRT(ERRL2/DNL2)
       WRITE(MTERM,*) '* PQL2U * REL. H1-ERROR    ',SQRT(ERRH1/DNH1)
       IF (MT.GT.1) THEN
        WRITE(MTERM,*) '* PQL2U * FUNCTION-VALUES L2-NORM ',SQRT(DNL2)
        WRITE(MTERM,*) '* PQL2U * FUNCTION-VALUES H1-NORM ',SQRT(DNH1)
       ENDIF
      ENDIF
      IF (MTERM.NE.MPROT) THEN
       WRITE(MPROT,*)
       WRITE(MPROT,*) '* PQL2U * SOLUTION ELE     ',IELTYP
       WRITE(MPROT,*) '* PQL2U * CUBATURE FORMULA ',ICUB
       WRITE(MPROT,*) '* PQL2U * REL. L2-ERROR    ',SQRT(ERRL2/DNL2)
       WRITE(MPROT,*) '* PQL2U * REL. H1-ERROR    ',SQRT(ERRH1/DNH1)
       IF (M.GT.1) THEN
        WRITE(MPROT,*) '* PQL2U * FUNCTION-VALUES L2-NORM ',SQRT(DNL2)
        WRITE(MPROT,*) '* PQL2U * FUNCTION-VALUES H1-NORM ',SQRT(DNH1)
       ENDIF
```

```
      ENDIF
C
99999 END
```

```
C
C *** Coarse grid triangulation
C
Coarse grid TRIA2
Unit cube - non cartesian grid (cube in cube)
      7 16 1 8 12 6   NEL NVT NBCT NVE NEE NAE
DCORVG
      0.00D0  0.00D0  0.00D0
      1.00D0  0.00D0  0.00D0
      1.00D0  1.00D0  0.00D0
      0.00D0  1.00D0  0.00D0
      0.00D0  0.00D0  1.00D0
      1.00D0  0.00D0  1.00D0
      1.00D0  1.00D0  1.00D0
      0.00D0  1.00D0  1.00D0
      0.25D0  0.25D0  0.25D0
      0.75D0  0.25D0  0.25D0
      0.75D0  0.75D0  0.25D0
      0.25D0  0.75D0  0.25D0
      0.25D0  0.25D0  0.75D0
      0.75D0  0.25D0  0.75D0
      0.75D0  0.75D0  0.75D0
      0.25D0  0.75D0  0.75D0
KVERT
      1  2  6  5  9 10 14 13
      2  3  7  6 10 11 15 14
      3  4  8  7 11 12 16 15
      1  4  8  5  9 12 16 13
      1  2  3  4  9 10 11 12
      5  6  7  8 13 14 15 16
      9 10 11 12 13 14 15 16
KNPR
      1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
```