FEAP - - A Finite Element Analysis Program

Version 8.6 User Manual

Robert L. Taylor & Sanjay Govindjee
Department of Civil and Environmental Engineering
University of California at Berkeley
Berkeley, California 94720-1710

August 2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

During the last several decades, the finite element method has evolved from a linear structural analysis procedure to a general technique for solving non-linear, transient, partial differential equations. An extensive literature on the method exists which describes the theory necessary to formulate solutions for general classes of problems, as well as, practical guidelines in its application to problem solution.[1–11]

This manual describes many of the features of the general purpose *Finite Element Analysis Program* (*FEAP*) to solve such problems. Many of the descriptions in this manual are directed to the solution of problems in solid mechanics, however, the system may be extended to solve problems in other subject areas by adding user developed modules to address a specific class of new problems. Such extensions have been made by users to solve problems in fluid dynamics, flow through porous media, thermo-electric fields, to name a few. Interested readers are directed to the *FEAP* Programmers Manual for details on adding new features.[12]

Instructions to install the program on personal computers is described in the *FEAP* Installation Manual.[13] All manuals may be downloaded from the web site:

`projects.ce.berkeley.edu/feap/`

This manual is directed at use of *FEAP* to solve problems on computers, such as desktop or laptop types. There are extended modules that allow for solution of problems on multiprocessor, parallel computers. For information on capabilities of this version see the *FEAP* Parallel User Manual.[14] A second extended module is a version for solution of multiscale problems using an FE^2 strategy. For information on this version consult the *FEAP* Multiscale User Manual.[15]

It is assumed that the reader of this manual is familiar with the finite element method as describe in reference books (e.g., *The Finite Element Method*, 6^{th} or 7^{th} edition, by O.C. Zienkiewicz and R.L. Taylor[1–3,16–18]) and desires either to solve a specific problem or to generate new solution capabilities.

The Finite Element Analysis Program (*FEAP*) is a computer analysis system designed for:

- Use in course instruction to illustrate performance of different types of elements and modeling methods;
- In a research, and/or applications environment which requires frequent modifications to address new problem areas or analysis requirements.

The computer system may be used in either a UNIX/Linux/Mac or a Windows environment and includes an integrated set of modules to perform:

- Input of data describing a finite element model;
- An element library for solids, structures and thermal analysis;
- Construction of solution algorithms to address a wide range of applications; and
- Graphical and numerical output of solution results.

A problem *solution* is constructed using a *command language concept* in which the solution algorithm is completely written by the user. Accordingly, with this capability, each application may use a solution strategy which meets its specific needs. There are sufficient commands included in the system for linear and non-linear applications in structural or fluid mechanics, heat transfer, and many other areas requiring solution of problems modeled by differential equations; including those for both steady state and transient problems.

Users also may add new routines for mesh generation and manipulation; model element or material description; new command language statements to meet specific application requirements; and plot outputs for added graphical display. These additions may be used to assist generation of meshes for specific classes of problems; to import meshes generated by other systems; or to interface with other graphical devices.

The current *FEAP* system contains a element library. to model one, two or three dimensional problems in linear and/or non-linear structural and solid mechanics,for linear heat conduction problems, and for acoustic fluid or interface solutions. Each of the provided elements accesses a material model library. Material models are provided for elastic, viscoelastic, elasto-plastic, and heat transfer constitutive equations. Elements also provide capability to generate mass or geometric stiffness matrices for structural problems and to compute output quantities associated for each element (e.g., stress, strain), including capability of projecting these quantities to nodes to permit graphical outputs of result contours.

Users also may add an element to the system by writing and linking a single module to the *FEAP* system. Details on specific requirements to add an element as well as

other optional features available are included in the *FEAP Programmers Manual* (see web site at: `projects.ce.berkeley.edu/feap`).

This manual describes how to use many of the existing capabilities in the *FEAP* system. In the next several sections the general features of *FEAP* are described. The discussion centers on three different phases of problem solution:

1. Finite element mesh description options;
2. Problem solution options; and
3. Graphical display options.

The general structure for an input file consists of alphanumeric data residing in a file called the *Input File* which describes each of the above parts. The generic form for an input file is given as:[1]

```
<*REAl or *COMplex> ! Default is *REAl
FEAP * * Start record and title
  ...
Control and mesh description data
  ...
END mesh
  ...

Solution and graphics commands
  ...
STOP
```

The first line describes whether the solution is performed in *real* or *complex* arithmetic. If the line is omitted it is assumed that the solution will be performed in *real* arithmetic. The *FEAP Example Manual* may be consulted for examples on use of some input and solution options described in this manual (see: `projects.ce.berkeley.edu/feap`).

To illustrate the form of an input file for *FEAP* we consider the simple king-post truss shown in Fig. 1.1. For simplicity we assume that all members are elastic with the same elastic modulus and the same cross-sectional area.

A complete input file to solve this problem is shown in Table 1.1. Note that the `*REAl` record is omitted, thus assuming all the solution will be performed in real arithmetic. This is the default mode. The first two lines of the file are called the *control information* and describe the start record followed by the number of nodes, number of elements, number of material sets, space dimension of the mesh, maximum number of unknowns at any node, and number of nodes/element, respectively. The preparation of the control records is described in Chapter 5.1 of this manual. These records are followed by data

[1] Optional input parameters are enclosed in `<...>` brackets.

sets which describe material and geometric properties (Chapter 7); the coordinates for each node and the nodal connections and material set identifier for each element (Chapter 5.2); and the boundary restraint and load descriptions (Chapter 5.5). Each set of data is normally terminated by a *blank record*. Comments are allowed in the data after any "`!`" character. The first `END` record informs *FEAP* that all data has been provided to define the finite element mesh for the problem.

The next set of records are the *command language statements* that define the solution algorithm. The first record `CHECk` is recommended to ensure that the data provided in the mesh input does not contain errors. *FEAP* is able to perform some checks on the data, however, not all will always be found. The check will provide a list of the boundary restraints that are enforced and for steady state problems it should always be ensured that sufficient number are available to prevent a singular, unphysical, solution. A graphical display of the mesh is also recommended to ensure that the geometry specified is correct, here the `PLOT MESH` command is used, however, for three dimensional meshes a perspective view is usually required (see Sect 15.1.6 for details). In the king-post truss example presented in Table 1.1 the information needed to perform a steady state (static) linear analysis is shown after the `CHECk` command. The second `END` statement informs the program that the set of `BATCh` commands is complete. Note that in the table the names of commands have the first four characters given in upper case form. In general, *FEAP* only processes the first four characters of data names. Names may be given either upper or lower case form.

The `INTEractive` command places *FEAP* in a mode where commands may be given from the keyboard (i.e., *the interactive mode*) and allow for subsequent solution, plot or mesh commands to be processed (multiple sets of `BATCh` or `INTEractive` solution sets are permitted). Chapter 14 describes the construction of command language programs for many of the features available in *FEAP*.

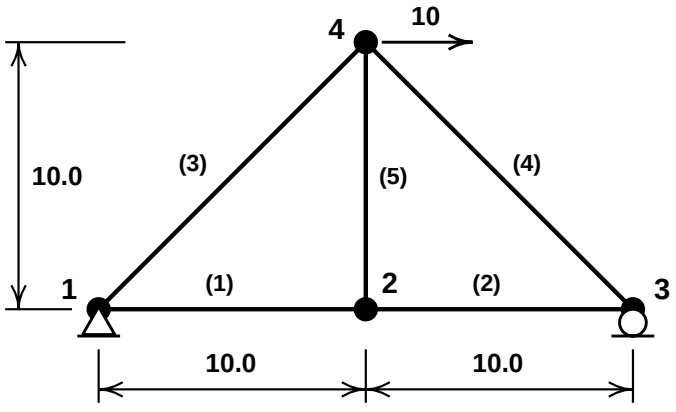


Figure 1.1: King-post truss example. • = Nodes; (n) = Element n

```
FEAP * * King-post truss analysis
    4    5    1    2    2    2

MATErial 1
  TRUSS
    ELAStic isotropic 10000.0
    CROSs   section   0.25
     ! Blank record to terminate
COORdinates
    1    0    0.0    0.0
    2    0   10.0    0.0
    3    0   20.0    0.0
    4    0   10.0   10.0
     ! Blank record to terminate
ELEMents
    1    1     1    1    2
    2    1     1    2    3
    3    1     1    1    4
    4    1     1    4    3
    5    1     1    2    4
     ! Blank record to terminate
BOUNdary restraints
    1    0     1    1
    3    0     0    1
     ! Blank record to terminate
FORCe
    4    0    10.   0.
     ! Blank record to terminate
END mesh

BATCh
  CHECk
  PLOT MESH
  TANGent
  FORM
  SOLV
  DISPlacement ALL
  STREss       ALL
END
INTEractive
STOP
```

Table 1.1: *FEAP* input data for king-post truss

The final `STOP` record informs *FEAP* that all data has been processed and execution ceases. Failure to include this record will usually result in a non-standard exit from the program – usually with some form of an error statement.

This simple example is intended to give an overview of what is required to prepare an

input file for the program. Only very basic commands have been used here and many other options are available to describe the problem data, solution options, and graphics capability available in the program. The remainder of this manual will describe many of these features and the appendices give all the commands available in the current release.

Interactive mode of execution is highly recommended when developing a mesh for a new problem. This allows a user to monitor any difficulties and to quickly correct deficiencies. *A mesh for a new problem should always be carefully checked before specifying any batch solution.*

1.1 Manual organization

The user manual for *FEAP* is separated into several distinct parts. Each part describes a specific function and the input data required for commands currently available in the system. The manual consists of the following general sections:

1. Methods to describe input data records and files (Chapter 4);
2. Description of the start of a problem, control information, and mesh input data (Chapter 5);
3. Description of the element library and material models (Chapters 6 and 7);
4. Specifying lumped parameters for mass, damping and stiffness (Chapter 8).
5. Data reuse by loops and includes (Chapter 9).
6. Terminating mesh description (Chapter 10);
7. Manipulating a mesh to merge parts or boundaries (Chapter 11);
8. Description of contact surface interactions (Chapter 12);
9. Designating some parts of bodies as rigid (Chapter 13);
10. Description of the solution command language (Chapter 14) [This section of the manual includes only basic solution algorithms to solve problems];
11. Plot features contained within the program (Chapter 15).

A complete list of the various options and parameters for each command to describe mesh input, problem solution, and plotting are included in the appendices to this manual. A separate *Example Manual* showing some applications of the program and a *Programmer Manual* describing the procedures to add features and elements are also

available for users who wish to modify or extend the capabilities of *FEAP*. Updated versions of all manuals are maintained at the web site `projects.ce.berkeley.edu/feap`. Exchange of information and questions by users may be given using the *Feap Forum* located at the web site `http://feap.berkeley.edu/forum/index.php`. In addition the *FEAP Wiki* at `http://feap.berkeley.edu/wiki/index.php/FEAP_Wiki_Main_Page` provides useful information on specific topics.

Chapter 2

Problem definition

To perform an analysis using the finite element method the first step is to subdivide the region of interest into elements and nodes. In this process the analyst must make a choice on:

1. The type of elements to use;
2. Where to place nodes;
3. How to apply the loading and boundary restraints;
4. The appropriate material model and parameters values for each element; and
5. Any other aspects relating to the particular problem.

The specification of the node and element data defines what we will subsequently refer to as the *finite element mesh* or, for short, the *mesh* of the problem.

Once the analyst has defined a model of the problem to be solved it is necessary to define the nodal and element data in a form which may be interpreted by the analysis program. For *FEAP* this step requires the user to prepare an *input file* that contains all the necessary steps to perform an analysis. The steps to define the input file for a mesh are contained in Chapters 5 to 11. Each command available to define mesh data is described in Appendix A and those to perform further manipulation on the mesh data are in Appendix B. Commands to perform manipulation on mesh data include merging parts or linking the degrees of freedom of one node to have the same value as at another node. Manipulation data is placed after the mesh `END` command and, if provided, the contact surface input `END` command. They must also appear before the first solution command set defined by a `BATCh` or `INTEractive` command.

Some problems in solid mechanics involve intermittent *contact* between bodies. *FEAP* provides some capability to solve such problems and a description of the necessary input

data is described in Chapter 12 with a description of all options given in Appendix C. Description of contact surfaces and surface behavior are described by data appearing after the mesh `END` command and before mesh manipulation or solution commands.

The second phase of a finite element analysis specifies the solution algorithm for the problem. This may range from a simple linear steady state (*static*) analysis for one loading condition up to a detailed non-linear, time-dependent (*transient*) analysis subjected to a variety of loading conditions. *FEAP* permits the analyst to specify the solution algorithm utilizing *command language* statements described in Chapter 14 and Appendix D. Solution commands are placed between a `BATCh-END` pair in a file or are entered one at a time during an `INTEractive` mode of solution. Each available solution command is described in Appendix D of this manual. Users may add their own solution commands as described in the *Programmers Manual*.

2.1 Execution of FEAP and filename specifications.

Once a file is prepared which contains all the steps necessary to describe the mesh data and solution commands (later we shall see that solution steps may be a minimal set of statements) an execution of *FEAP* is initiated. Depending on the installation this is given as:

- A command line input from a text window by issuing the command:[1]

```
feap
```

  - In a Windows environment it is possible to execute the program in this mode using a 'Command prompt' (MS-DOS type) window and execute with the above command.[2] In a Windows environment an alternative is to have a 'pop-up' window appear which can to be traversed to the location of the folder containing the desired 'input file' to be executed. The file may be selected using the mouse in a standard Windows manner. In a Windows environment all subsequent solution steps are performed within a graphical context permitting both text and screen plots in the same environment.
  - In a UNIX/Linux or Apple OS X environment the command line window where *FEAP* is initiated must be able to launch the graphics window as a graphics X11-window.

[1] The name of the executable program is established during the compilation phase and may be changed by a user from that shown. For simplicity, we use the generic name `feap` here.

[2] It is useful to write a batch program which describes the directory path to the executable so that the solution may be easily initiated from any directory. The batch file should be placed in a directory accessible on the path names.

In a first execution of the program in each directory it is necessary to provide a name for the file containing the 'input data'. *FEAP* will create additional files that collect selected output information. The basic *output files* can contain the numerical values for input information as well as results contained from any analysis. Default names will be provided for the output and other files but may be changed by the user if desired.

In addition to the output file a *log file* will be created for each analysis. The log file contains basic performance information as well as error messages. The log file begins with `L` and contains the name of the input file.

The basic information in the log file contains convergence information that should always be consulted concerning possible lack of convergence for nonlinear problems. In addition a summary of execution time is provided.

- In a Windows environment an 'icon' for the *FEAP* executable may be placed on the 'Desktop'. The program is executed by double clicking on the icon in a standard manner. A 'pop-up' window will appear and needs to be traversed to the location of the directory containing the desired 'input file' to be executed. The 'input data' file may be selected by double clicking the mouse on the desired filename. The name for all other files is provided by the program and may not be changed.

It is recommended that names of input files begin with an upper case `I` to assist *FEAP* in constructing the name for all the other files. In addition to specifying the filename containing the input data (i.e., the *input* file) the names for files containing output and restart information can be provided.

Upon a successful first execution of the program a file named `feapname` will be written in the solution directory to preserves the name for each of the input and output file names for subsequent executions.

For each subsequent execution of the program using a `FEAP` command line input, the user receives a request to accept the current default names (by entering a `y`); to define new set of files (by entering a `r`); or to stop execution (by entering a `s`). If the option `r` is given, prompts for a new input data filename, as well as for the filenames which are to contain the output of results and diagnostics, and restart files (used if subsequent analyses are desired starting with the final results of a previous execution). Default filenames are indicated may be accepted by pressing the return (enter) key without specifying any new data. Once new files are given the option `y` MUST be given.

Prior to running *FEAP* it is necessary to create the input data file using a standard text editor or word processing system. The other files are created automatically by *FEAP*. A large part of the remainder of this manual is directed toward defining a valid input data file and to describe the command language instructions needed to solve and output results for several classes of problems.

Execution of *FEAP* also may be made without specifying filenames interactively. The command line to perform this mode of execution is:[3]

```
feap -iIfile -oOfile -rRfile -sSfile -pPfile
```

Each parameter defines the name of the file which either contains input data or will be used to produce the output data. The files are:

```
i = input       : Ifile is file containing input data
o = output      : Ofile is file for outputs
r = restart     : Rfile is filename restart read/writes
s = save        : Sfile is filename for data saves
p = plot        : Pfile is root name for file
                  containing time history data.
```

Except for the name of the input data file, these parameters are optional. Thus, the minimum command line for this form of execution is:

```
feap -iIfile
```

the other files are given by replacing the first character in the Ifile name by `O`, `R`, `S`, `P`. If the restart `R` filename is specified it is automatically used (see Sect. 14.2).

Note: There can be NO blank characters between the *-i*, *-o*, etc. and the corresponding file name. That is the form

```
feap -i Ifile
```

will cause an error.

The above form is useful for making many sequential runs of the program in which all solution steps are performed in a batch mode (see Chapter 14). In this case a file containing the sequence:

```
feap -iIfile1
feap -iIfile2
  ...
feap -iIfilen
```

may be prepared and used to run the program. *This is much better than running several copies of the program in 'parallel'!*

An alternative to this is to prepare an input file which *INCLudes* each of the examples. An example is to prepare an input file (say named `Istart`) in the form

```
include Ifile1
include Ifile2
  ...
include Ifilen
STOP    ! Ensures normal termination of feap
```

[3]This form of the solution command **must** be given from a 'Command Prompt' window and may not work with all operating systems.

and then execute feap as

```
feap
```

and specifying `Istart` as the input file. Alternatively, one can specify

```
feap -iIstart
```

In a windows environment this form may also be initiated from the pop-up box by selecting the file `Istart`. In this form *FEAP* ignores all the `STOP` commands in the include files and only terminates normally after executing all of the specified problems. Output for each problem will be placed in separate files.

2.2 Modification of default options

When the executable version of *FEAP* is created default values for several parameters are set in the main program file `feap86.f`. These default parameters may be changed without recompiling the program by creating a file named `feap.ins` which contains the new values for specific parameters. This file must be placed in each directory where problems are to be solved. The `feap.ins` file contains separate records which define the default parameters to be employed during any solution. The current options are given in Table 2.1.

Option	Parameter 1	Parameter 2	Description
Manfile	mesh	path	Path to locate MESH COMMAND manual pages
	macr	path	Path to locate SOLUTION COMMAND manual pages
	plot	path	Path to locate PLOT COMMAND manual pages
	elem	path	Path to locate USER ELEMENT manual pages
noparse			Assumes input data is mostly numeric
parse			Assumes input data contains parameters
graphic	prompt	off	Turns off contour prompts
		on	Turns on contour prompts
	default	off	Turns off graphics defaults
		on	Turns on graphics defaults
postscr	color	reverse	Makes color PostScript files with color reversed order.
	color	normal	Makes normal color PostScript files with normal order.
helplev	basic		Default level for commands Same as: MANU,0
	medium		Default level for commands Same as: MANU,1
	advance		Default level for commands Same as: MANU,2
	expert		Default level for commands Same as: MANU,3
fileche	off		Turns off file checking at startup
	on		Turns on file checking at startup. Note: UNIX should not turn off!

Table 2.1: Options for Changing Default Parameters

Chapter 3

Elements types

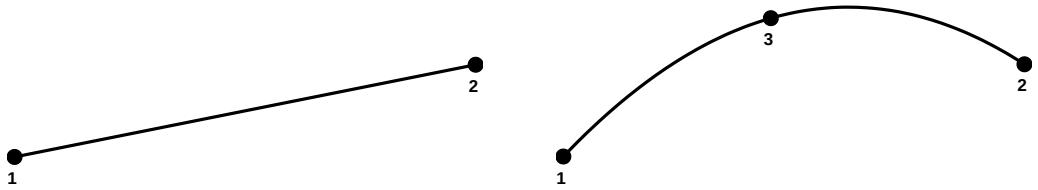
The description of an *element* for *FEAP* is expressed as a set of *node* numbers that describe the geometry and *connectivity*. An element may have a topology that is a *line*, a *surface* or a *volume*. In *FEAP* the nodes for each element are generally associated with unknown parameters of the problem. To describe a problem it is necessary to know what unknowns belong to each node and to specify the *maximum* number of unknowns which will be assigned to any node. This information is specified by the control records (see Chapter 5.1).

3.1 Line Elements

Line elements are defined by 2 or more nodes and two types included in the standard *FEAP* library are shown in Fig. 3.1. Numbers shown with elements describe the ordering that connectivity is to be specified on a data record for each element. In *FEAP* this describes the *local node numbers* of the element. The element library included with the *FEAP* system can generate line elements with up to 10 nodes. Although unlikely needed, elements with more nodes may be added by a user as described in the programmer manual.[12]

Line elements are used with the standard *FEAP* element library to describe one-dimensional *solids* in a state of plane stress, plane strain, or axisymmetric deformation (see Chapter 6). They may also be used as *truss* and *frame* type elements for two and three dimensional problems. Two and three node elements also may be used to describe *shell segments* for the meridian of an axisymmetric shell modeled in a two dimensional analysis (for a one radian segment in the circumferential θ direction). Other uses for line type elements include description of *pressure boundary loads* (see Section 6.12) and *thermal convection surface conditions* for heat conduction problems (see Section 6.3).

Elements with the minimum number of nodes to create an appropriate geometrical space are called *simplex elements* and those with more nodes are of *higher order elements*. An advantage of higher order elements, with quadratic or higher edges, is that they may be curved to better match boundaries or the shape of a body, as shown in Fig. 3.1 for the 3-node element. They also provide higher order functions in the element and thus attain better accuracy for a given number of nodes. That is, one 3-node line element will generally give better accuracy than two 2-node line elements.



2-Node Element 3-Node Element

Figure 3.1: Line type elements in *FEAP* library

3.2 Surface Elements

Surface finite elements are generally described by triangular or quadrilateral shapes. Triangular elements included with the *FEAP* system may be described by the 3-node simplex or by the 6-node (quadratic) or 10-node (cubic) higher order element as shown in Fig. 3.2. A 6-node or 10-node element may have curved sides, as shown for the 3-node line element in Fig. 3.1. For most element formulations the geometric shape of each element is accomplished using a *parametric mapping* described by

$$\mathbf{x} = \sum_a N_a(\boldsymbol{\xi})\, \mathbf{x}_a$$

where $\boldsymbol{\xi}$ are local 'parent' coordinates, N_a are the element *shape functions*, $\mathbf{x}$ are the global coordinates, and $\mathbf{x}_a$ are nodal parameters as described in standard reference books on finite elements (e.g., see Zienkiewicz, Taylor & Zhu[1,16]).

Surface elements may also be of quadrilateral shape as shown in Fig. 3.3 The basic (bilinear) element has 4-nodes and can be mapped into a general quadrilateral shape with straight sides. The elements with 8 and 12 nodes belong to a family named *Serendipity* and the elements with 9 and 16 nodes to a family named *Lagrangian*.[1,16] These elements are of higher order and may have curved sides when mapped. In general, it is preferable to use the 9 and 16 node elements rather than those with 8 or 12 nodes. This is especially true if problems in solid or fluid mechanics are solved in

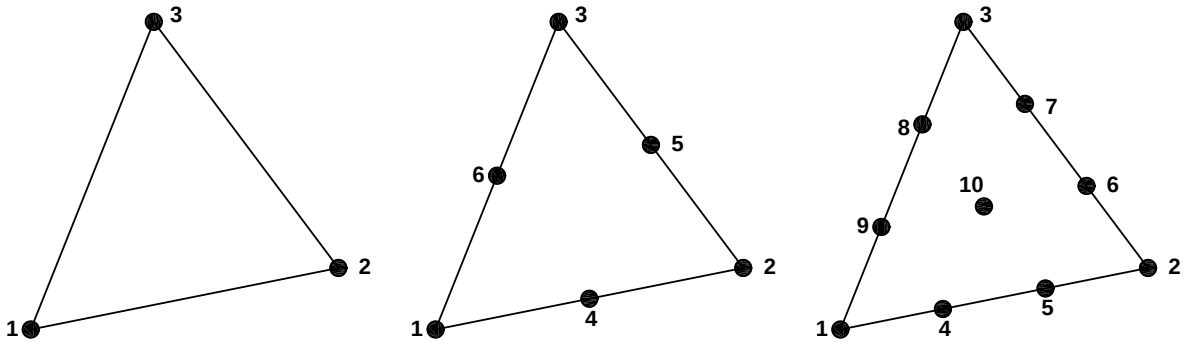


Figure 3.2: Triangular surface type elements in *FEAP* library

which *near incompressibility* conditions can exist. Also, when *lumped* or *diagonal* mass matrices are used in transient situations the numerical properties for Lagrangian type elements are better than those for Serendipity type elements. For further discussion on this topic consult standard reference books on finite element theory.[1,16]

Surface elements are used in *FEAP* to model solids in a state of plane stress, plane strain, or axisymmetric deformation. For the axisymmetric loading nodal forces are computed for a one radian segment in the circumferential θ direction. Most of the *solid* mechanics type problems in two dimensions can use any of the types of elements shown in Figs. 3.2 and 3.3 (an exception is the class called *enhanced strain elements* where only 4 node quadrilaterals may be used). This class of element topology is also used in two dimensional heat conduction analysis. In addition, surface elements are used to model plate and general shell problems; however, in the library of the current release of *FEAP*, the element shape is restricted to a 3-node triangle or a 4-node quadrilateral.

3.3 Volume Elements

Volume elements included in the *FEAP* library may be of tetrahedral or hexagonal (brick) shape. The simplex element is a 4-node tetrahedron and the first higher order elements are a 10-node and 14-node tetrahedron as shown in Fig. 3.4. In addition the 10-node tetrahedron allows an internal node 11 and the 14-node tetrahedron allows an internal node 15 - both at the barycenter of the volume (not shown in the figure). The 10- to 15-node elements may have curved edges and faces when mapped.

Volume elements may also have a *hexagon* (*brick*) shape with the lowest order element described by 8-nodes as shown in Fig. 3.5. The next higher order elements may have either 20-, 27- or 64-nodes. The 20-node element is a member of the Serendipity family. The 27-node element is the quadratic order member of the Lagrangian family and the

64-node element is the cubic order Lagrangian element. Figure 3.5 shows these three types of elements.

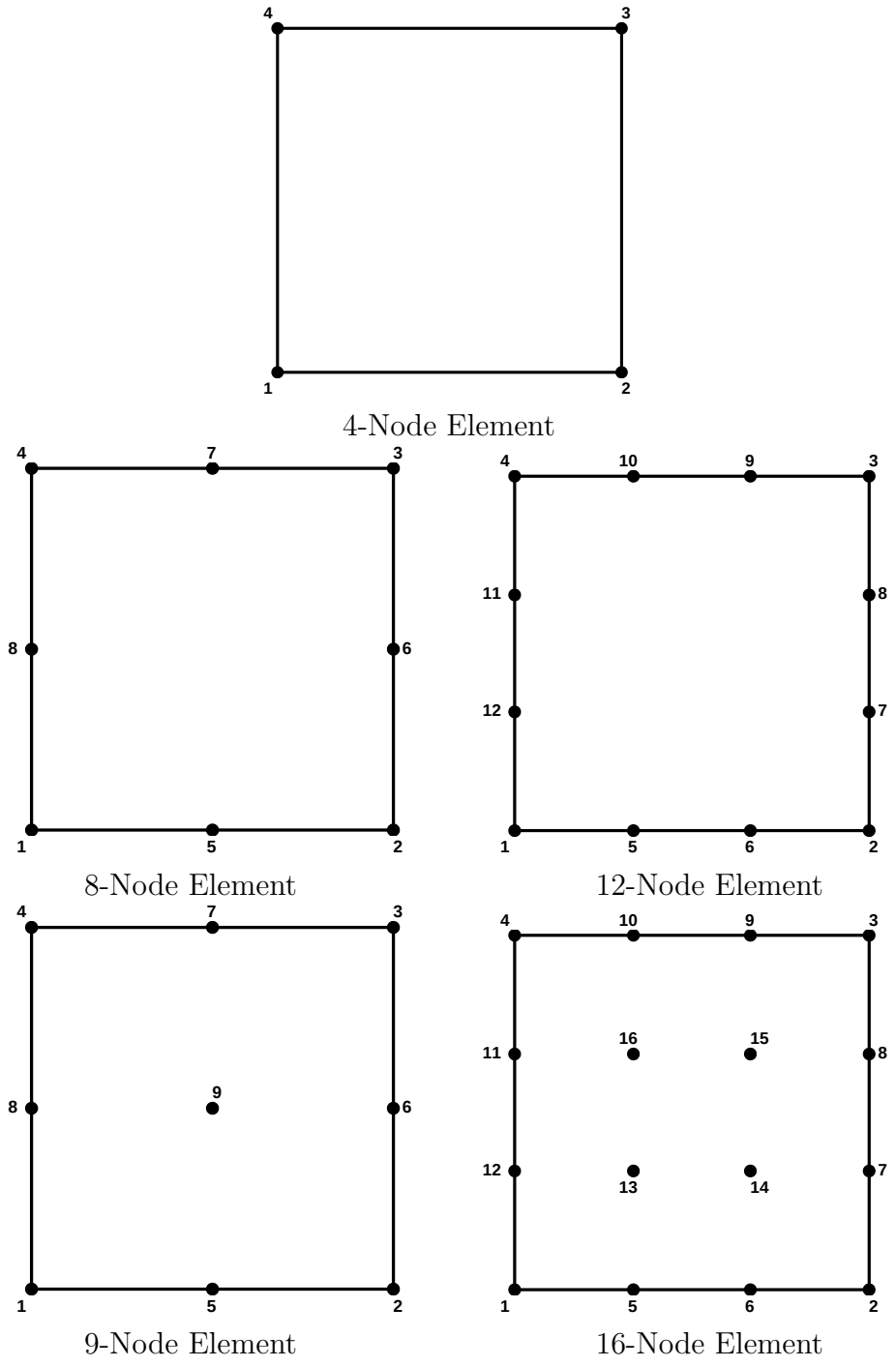
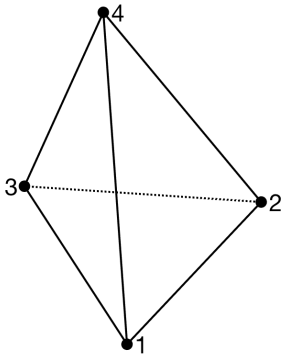


Figure 3.3: Quadrilateral surface type elements in *FEAP* library

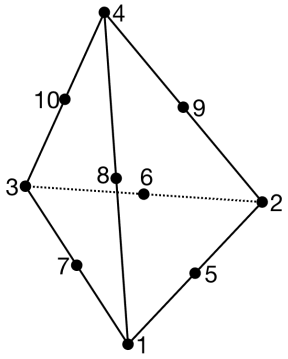
Figure 3.5 only shows the vertex node numbers for the 64-node brick element. The remaining numbers appear in increasing order of layers from $\xi_3 = -1$ to $\xi_3 = 1$ as shown in Figure 3.6. Note that the numbering system differs from the lower order elements.

In addition to the tetrahedron and hexahedron shapes, transition elements with a wedge (Figure 3.7(a)) and pyramid (Figure 3.7(b)) shape are allowed - but only for the lowest order elements.

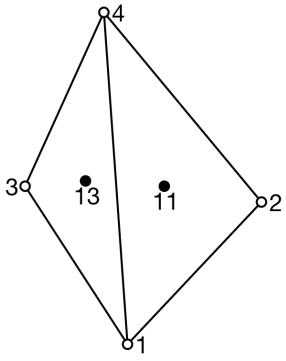
Volume elements are used in *FEAP* to model general three-dimensional problems in solid mechanics, fluid mechanics and heat conduction. Solid mechanics elements are available based on displacement, mixed, and enhanced strain formulations.[1] All formulation types permit use of 8-node hexagon/brick elements. Displacement and mixed formulations permit use of 27-node and 64-node hexagon/brick elements of Lagrangian type. Displacement and mixed formulations also permit use of 4-node or 10 or 11-node tetrahedra (N.B. The tetrahedral elements are not as robust in mixed form as the

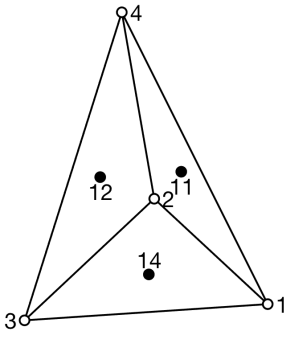


(a) 4-Node simplex



(b) 10- & 14-Node vertices





(c) 14-Node face nodes

Figure 3.4: Tetrahedron volume elements in *FEAP* library

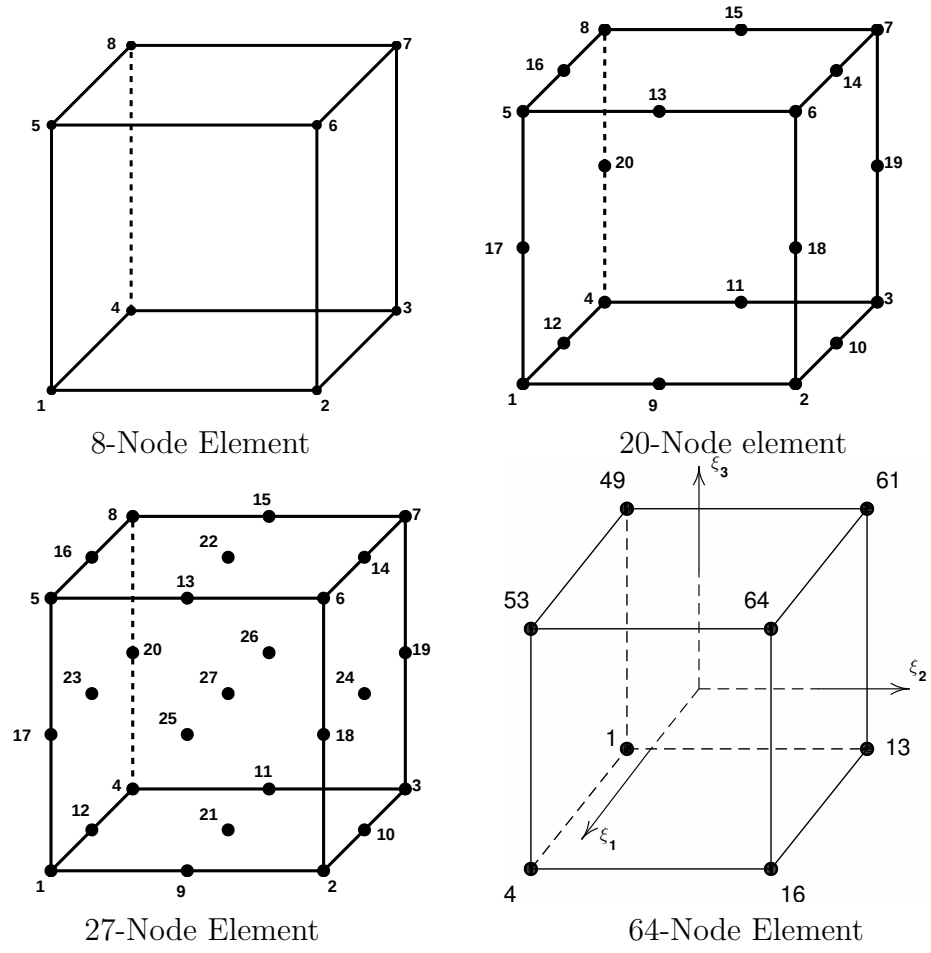
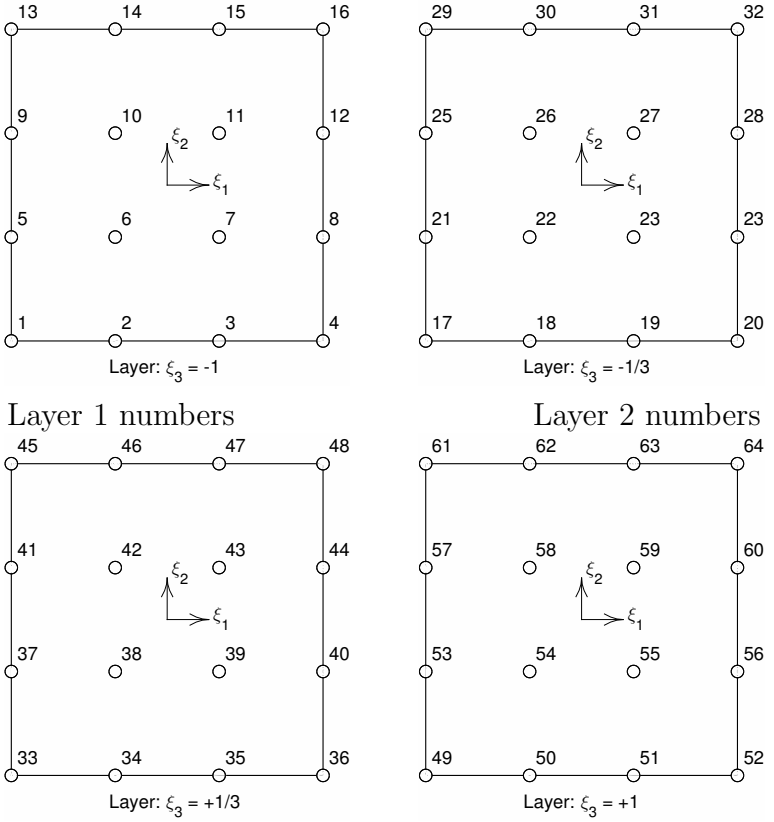


Figure 3.5: Hexahedron (Brick) volume elements in *FEAP* library

27-node or 64-node brick elements), as well as the 5-node pyramid or 6-node wedge shapes.

Except for the 20-node brick element, shape functions for Serendipity type elements are not provided in the standard *FEAP* system. Similar to two-dimensional element shapes, it is preferable to use the Lagrangian 27-node element than the Serendipity 20-node element. However, contrasted with the two dimensional case the cost of such use is much greater due to the added mid-face nodes.

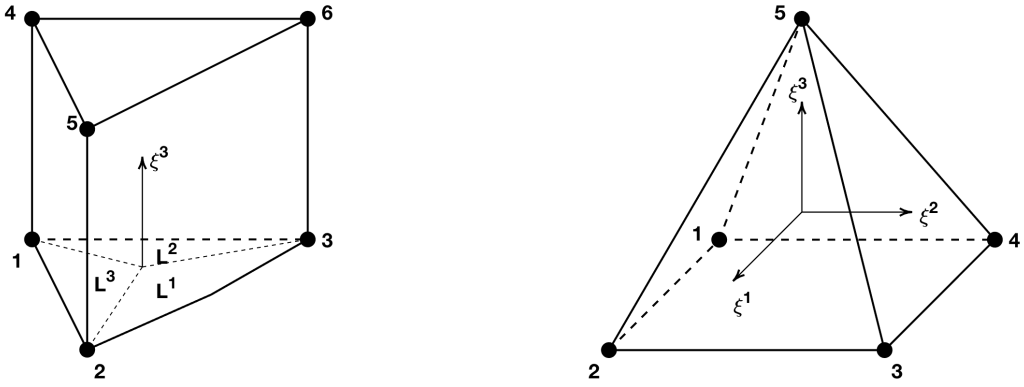
Note that, except for the 64-node volume element, the local numbering of nodes on each element start with vertex (corner) nodes and define the lowest order element of the class. this is followed by edge nodes, face nodes, and internal nodes.



Layer 1 numbers Layer 2 numbers

Layer 3 numbers Layer 4 numbers

Figure 3.6: 64-node brick volume element numbering



Wedge element Pyramid element

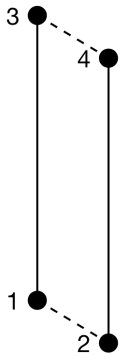
Figure 3.7: 6-node wedge and 5-node pyramid volume element.

3.4 Interface Elements

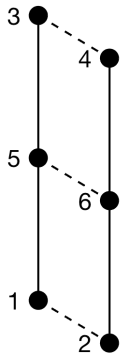
Interface elements are used to model internal surfaces of a mesh in which physics happens on a reduced scale. The two-dimensions the interface is pair of lines (straight or curved) where one line is assoiated with the surface of a volume element and the other line with the surface of a second volume element. An interface element has no thickness. Figure 3.8(a) & (b) show a linear and a quadratic interface element and the local numbering of nodes. The two lines are separated for illustration purposes only.

In three dimensions an interface consists of two trianglular or quadrilateral surfaces as shown in Figure 3.8(c) & (d). It is also possible to use quadratic interfaces in which additional mid-side and, for the quadrilateral, mid-face nodes may be added. Numbering always goes sequentially from face 1 to face 2.

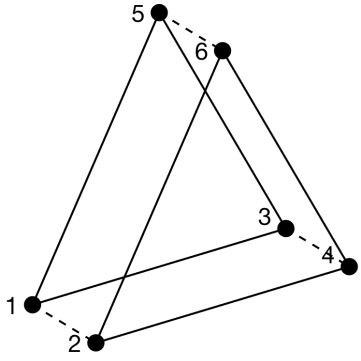
The edge or surface of each interface element must have the same number of local



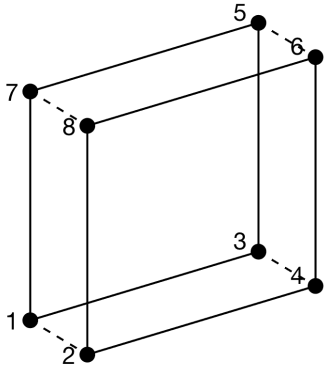
(a) Linear edge interface



(b) Quadratic edge interface



(c) Triangular 3-d interface



(d) Quadrilateral 3-d interface

Figure 3.8: Two and three dimensional interface elements.

nodes and the coordinates of nodal pairs must also be the same. See Section 14.1.10 for details on automatic generation of interface element meshes from a mesh of volume elements.

Chapter 4

Input record specification

Data input specifications in *FEAP* consist of individual records that may contain from 1 to 255 characters of information in free format form. Each record can contain up to 16 alphanumeric data items – but NO MORE. If more information is required another record must be provided. The default maximum field width for any single data item is 15 characters (14 characters of data and 1 character for separating fields).[1] Specific types of data items are discussed below. Sets of records, called *data sets*, start with a text command which controls input of one or more data items. Only the first four characters of each text command are interpreted by *FEAP*. To emphasize this restriction, the first four letters of each text command are shown in upper case letters, while the remainder are in lower case. However, users may specify data in either upper or lower case in the input data. It is only necessary to give the first four characters for each text command; however, additional characters may be added for clarification of meanings. Data sets may be grouped into a single file (called the input data file) or may be separated into several files and joined together using the `INCLude` command described in Section 9.1. Sets of records may also be designated as a *save* set and later *read* again for reuse (see Section 9.2).

Generally, each input record may be in the form of text and/or numerical constants, parameters, or expressions. Some exceptions to this do exist – for example input of coordinates by the `COORdinate ALL` command in which data must be strictly numerical and all fields in each record must be given. Similarly for the `ELEMent ALL` command.

- *Text* fields all start with the letters `A` through `Z` (either upper or lower case may be used, however, internally *FEAP* converts all upper case letters to lower case). The remaining characters may be either letters or numbers.

- *Constants* are conventional forms for specifying input data and may be integer

[1] Exceptions occur for input of coordinates and elements when all node and element records are provided in the input file.

or real quantities as needed.

- *Parameters* consist of one or two characters to which values are assigned. The first character of a parameter must be a letter (`a` to `z`); the second may be a letter (`a` to `z`) or numeral (`0` to `9`).
- *Expressions* are combinations of constants, parameters, and/or functions which can be evaluated as the required data input item.

Each of these forms is described below in greater detail.

4.1 Constants

Constants may be represented as integers or floating point numbers. Integers are specified without a decimal point as

```
1, -10, 345
```

etc.; floating point numbers may only be expressed in the forms

```
3.56,   -12.37,   1.34e+5,   -4.36d-05
```

In particular, the forms

```
1.0+3,  -3.456-03
```

may not be used since they will be evaluated as an *expression* (see below) and the above two examples will yield data input values

```
4.0,  -6.456
```

respectively.

4.2 Parameters

The use of parameters can simplify the data input required to define steps for a *FEAP* solution. Data may be specified as a single character parameter (e.g., `a`, `b`, through `z`), two character parameters (e.g., `aa`, `ab` through `zz`), or a character and a numeral (e.g., `e0` through `e9`). All alphabetic input characters are automatically converted to lower case, hence there are 962 unique parameters permitted at any one time. Values are assigned to parameters by the `PARA`meter data command during mesh generation or modification. The general form to assign a constant to a parameter is

```
PARAmeter
  a  = 3.567
  e1 = 200.0e9
  nu = 0.3
              ! Terminate input of parameters
```

Except in expressions, blanks are permitted and are ignored in the processing of a record. Once a parameter is defined it may be used in place of any constant in a data input. For example, the following input could use the value of the parameter `a` defined above

```
COORdinates
  1,,a,0.
```

and with this assignment the 1-coordinate of the 1-node would have a value of 3.567.

Parameters may have their values redefined as many times as needed by using the `PARAmeter` data command followed by other commands and data using the values of assigned parameters. A user may then specify another `PARAmeter` command to redefine parameters, followed by additional data inputs, etc.

As noted above, the specification of each constant is restricted to 14 significant figures (including the exponent value) plus a separator (either a comma or a blank). If more significant figures are needed in an exponent form, parameters or an expression may be used. For example,

```
a1 =  1.234567890123*1.e-5
```

produces a number with the full 14 digits but with an exponent larger than could otherwise be obtained with this precision and stay within the 14 character limit.

4.3 Expressions

The most powerful form of data input in *FEAP* is through the use of expressions in combination with parameters. An expression may include parameters and/or constants. Expressions may include operations of addition, subtraction, multiplication, division, and exponentiation. In addition, some functions may be used. A hierarchical evaluation is performed according to the rules defined in Table 4.1.

Order	Operation	Notation
1.	Parenthetical expressions	`( )`
2.	Functions	
3.	Exponentiation	`^`
4.	Multiplication or Division	`*` or `/`
5.	Addition or Subtraction	`+` or `-`

Table 4.1: Hierarchy for expression evaluation

Evaluations within the hierarchy proceed from left to right in each expression. At the present time only one level of parenthesis may appear in any expression. Using the hierarchy from Table 4.1, the expression

```
3/4 + 4
```

is evaluated as `4.75`, whereas

```
3/(4 + 4)
```

is evaluated as `0.375`.

All constants, parameters, and expressions are evaluated as double precision real quantities, however, they are permitted in place of integer data also with the result computed as the *nearest* integer of the real value obtained. In Fortran this is accomplished using the statement

```
i = nint(a)
```

Thus a parameter $a = 4.75$ would have an integer value of 5 when evaluated by the above statement. Expressions may appear in any location in place of a constant or a parameter. Accordingly, a force may be assigned to node 1 as

```
FORCe
  1,,a/12. + 3.
```

By default, the input of a force specifies only the *real* part, the imaginary part may be specified by

```
IFORce
  1,,-2.
```

and would set the imaginary part to -2 units. A similar form is available for `DISP` and `IDIS`. Additionally, node and element numbers could also appear as expressions; however, the use of the `*AUTo` or `*NODe` and/or `*ELEment` option described in Section 9.4 is a better way to reuse mesh parts with different assigned node or element numbers.

4.4 Functions

The following functions may appear in an expression, a statement, or a parameter definition:

```
abs   dec,  exp,  inc,   int,   nint,  log,   sqrt,
sin,  cos,  tan,  atan,  asin,  acos,
sind, cosd, tand, atand, asind, acosd,
cosh, sinh, tanh,
```

The trigonometric and inverse trigonometric functions such as `sind`, etc., involve values of angles in *degrees*; whereas, those such as `sin`, etc., involve values in *radians*.

Each function has one argument which is contained between a parenthesis (which counts as the allowed one level of parenthesis depth). The argument may be an expression but may not contain any additional parentheses or functions. Thus, the expression

```
pi = 4.*atan(1)
```

or

```
pi = acos(-1)
```

will compute the value of π to full numerical precision of the computer used and assign it to the parameter `pi`. Internal computations are all performed in double precision arithmetic (e.g., as `REAL (KIND=8)` variables). We note that the function parenthesis count as one level, hence

```
q = tan(1./(3.+a))
```

is an illegal expression. It can be replaced by the pair of statements

```
q = 1./(3.+a)
q = tan(q)
```

to avoid the double parentheses.

The integer functions `int` and `nint` are used to compute integer values from real expressions. They are not needed if single parameters are expressed by integer values. However, if integers are computed from expressions involving functions or division they can prevent unexpected results. Note that the function `int` can have rounding problems near whole numbers: That is `int(0.9999...` yields the value zero (0); whereas `nint(0.9999...` yields the value one (1); similarly around `0.5` reverse rounding can occur.

Chapter 5

Mesh input data specification

The description of the mesh data for a problem to be solved by *FEAP* consists of several parts described in the following sections.

5.1 Start of problem and control information

The first record of an input file describes whether the problem is to solved in real or complex arithmetic. The record is either `*REAl` or `*COMplex`. The default mode is `*REAl` and the record may be omitted. In addition to the solution type, parameters may precede the problem control records. This permits setting sizes such as maximum nodes/element as parameters to facilitate behavior with different order interpolation.

The next part of an input data file contains the *control data* which consists of two records:

- A start/title record which must have as the first four non-blank characters `FEAP` (either upper or lower case letters may be used with the remainder up to character 80 used as a problem title in the output file).
- A second record contains problem size information with *required* data consisting of:
  - `NUMNP` - Number of nodal points;
  - `NUMEL` - Number of elements;
  - `NUMMAT` - Number of material property sets;
  - `NDM` - Space dimension of mesh;
  - `NDF` - Maximum number of unknowns per node; and
  - `NEN` - Maximum number of nodes per element.

> WARNING: Do not place data beyond `NEN` as additional fields exist on the control record for advanced features. See Appendix A for details.

As described in Chapter 4, input records for *FEAP* are in free format. Each data item is separated by a *comma*, an *equal sign* or a *blank* characters. If blank characters are used without commas, each data item *must* be included. That is multiple blank fields are not considered to be a zero. Each data item is restricted to 14 characters (15 including the blank, equal or comma).

For standard input options included in the program modules, *FEAP* can automatically determine the number of nodes (`NUMNP`), elements (`NUMEL`), and the number of material sets (`NUMMAT`). Thus, their values on the control record may be specified as zero (0). When using this automatic numbering feature it is generally advisable to use mesh input options which avoid direct specification of a node or element number. Specification of many types of inputs sets have options which begin with `E` for *edge* and `C` for *coordinate* related options (e.g., `CFORce` for input of nodal forces by their coordinate location; or `EBOUndary` for input of boundary restraint codes for nodes). It is recommended these options be used whenever possible as it avoids the direct specification of a node number.

The use of the automatic determination of number of nodes, elements and material sets requires the mesh data to be read twice: Once to do counting and once to perform actual input of the data. For problems with a large number of data records, this may result in some time lapse during the input data phase. The need for a second read may be avoided by inserting a `NOCOunt` record *before* the `FEAP` record and then providing the actual number of nodes, elements and material sets on the control record. For other improvements in input speed see the use of the `ALL` option on `COORdinate` and `ELEMent` data input in 5.2.1 and 5.2.2, respectively.

The commands used to describe the remainder of the finite element mesh are considered next. As noted above, in *FEAP* each data set starts with a command text name of which only the first four characters are used as identifiers. Appendix A describes options for each mesh input command and Appendix B each mesh manipulation command. Immediately following each command record the data to be processed must appear with no blank records between. Where a variable number of records is needed to define the data set a blank record is used to terminate input of the data set. Extra blank records after each complete data sets are ignored.

Text commands may be in any order. If there is any order dependence *FEAP* will transfer the input data to temporary files and process each one after the mesh specification is terminated by the mesh `END` command. Thus, information will not necessarily appear in the output file in the same order that data is placed in the input file.

5.1.1 Use of PRINt and NOPRint commands

By default all data from a mesh input is written to the output file. For very large problems the size of the output file may become excessively large. Once a mesh has been checked for correctness it may not be necessary to retain this information for subsequent analyses. Control of the data retained in the output file is provided by using the `PRINt` and `NOPRint` commands. By default `PRINt` is assumed and all data is written to the output file. Insertion of a `NOPRint` record before any data set (but not within a data set) suspends writing the data to the output file until another `PRINt` command is encountered.

5.2 Coordinate and element connections

The basic mesh for *FEAP* consists of nodes and elements. For the finite elements included with the program the *mesh* is described relative to a global Cartesian coordinate frame. For two-dimensional plane problems the mesh lies in the x_1-x_2 plane (or the $x-y$ plane). For axisymmetric problems the mesh lies in the $r-z$ plane (which is placed in the x_1-x_2 plane). All elements elements for axisymmetric problems provided in *FEAP* compute stiffness and residual arrays for a *one radian segment* in the circumferential direction (i.e., the factor 2π is omitted). For three dimensional problems a general x_1, x_2, x_3 (or x, y, z) coordinate system is used. In the sequel we will discuss the specification of the input data relative to the x_i components. While eventually all nodal coordinates must be specified relative to the x_i frame, it is possible to use other coordinate systems (e.g., polar and spherical) as the input data and then transform these coordinates to a Cartesian frame (see Section 5.3 for more details). For example, the mesh for the curved beam shown in Figure 5.1 may be input in polar coordinates and then, subsequently transformed to Cartesian coordinates.

5.2.1 COORdinate input command

The coordinates for nodes may be specified using the `COORdinate` command as

```
COORdinate
```

followed by individual records defining each node and its coordinates as:

```
N, NG, X_N, Y_N, Z_N
```

where

`N`	Number of nodal point.
`NG`	Generation increment to next node.
`X_N`	Value of x_1 coordinate.
`Y_N`	Value of x_2 coordinate.
`Z_N`	Value of x_3 coordinate.

It is only necessary to specify the components up to the spatial dimension of the mesh (NDM on the control record). Thus for 2-dimensional meshes only X_N and Y_N need be given.

As an example consider the commands needed to generate the coordinates for an eleven node mesh of a circular beam with radius 5. These may be generated in two steps:

1. Polar coordinate form given by:

```
COORdinates
  1  1  5.0  90.0
 11  0  5.0   0.0
 ! Termination record
```

   followed by

2. Conversion from polar to Cartesian form using the POLAr command. For the coordinate input shown above this is given as:

```
POLAr
  NODES  1  11  1
 ! Termination record
```

   which converts the nodes 1 to 11 in increments of 1.

Generation of missing data is performed from data pairs given as:

```
M, MG, X_M, Y_M, Z_M
N, NG, X_N, Y_N, Z_N
```

Here, the missing data is generated from node M to node N in increments of MG; that is the first generated node will be M+MG, the second M+2*MG, etc. Linear interpolation of coordinates is used to define the intermediate values for the generated nodes. If MG

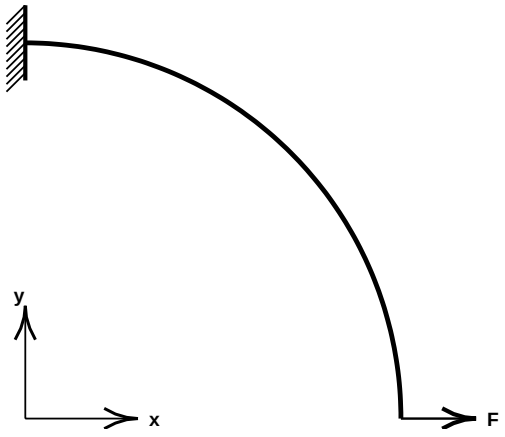


Figure 5.1: Curved Beam

is zero no generation is performed. Nodes may be in either increasing or decreasing order. The sign of any non-zero MG will be determined to ensure that generation is in the correct direction.

Coordinate data is processed to determine the total number of nodes (NUMNP) in a mesh. Nodal coordinates may also be defined using the BLOCk and the BLENd commands (see Sections 5.2.3 and 5.2.5 below) or any combination of the three command forms.

When no generation is required to input all the coordinate information the option

```
COORdinate ALL
  1 0  X_1 Y_1 Z_1
  2 0  X_2 Y_2 Z_2
      ..
```

or if the generation parameter is omitted by

```
COORdinate ALL
  NOGEneration
  1  X_1 Y_1 Z_1
  2  X_2 Y_2 Z_2
      ..
```

may be used. In this option all the data *must* be given as constants – with no parameters or expressions permitted.

If the numeric values are in a separate file, an INCLude form may be used as

```
COORdinate ALL
  INCLude filename
```

or

```
COORdinate ALL
  INCLude NOGEneration
```

Usually, this form of data results when the coordinate (and element) values are created by an external mesh generation program. If both elements and coordinates use the ALL option the NOCOunt option may be used as described in Section 5.1.

5.2.2 ELEMent input command

The ELEMent command may be used to input the list of nodes connected to an individual element. In addition to the ALL parameter, the ELEMent command has additional optional parameters specified (in any order or omitted) by:

```
ELEMent NODE=nd MATE=ma TYPE=ty ORDEr=or
```

where

nd	Nodes/element (may be less than NEN.
ma	Material set for all elements.
ty	LINE, TRIA, QUAD, TETR, HEXA, WEDG, PYRA, POIN
or	1, 2, etc..

The use of the parameters permits easier mixture of element types by splitting the data into multiple ELEMent sets, especially for cases where some elements require multiple records (see below).

For elements where the maximum number of nodes is less or equal to 13 (i.e., the NEN parameter on the control record or the value of the NODE parameter above), the records following the command are given as:

```
N, NG, MA, (ND_i, i=1,NEN)
```

where

N	Number of element.
NG	Generation increment for node numbers.
MA	Material identifier associated with element.
ND_i	i-Node number defining element .

For meshes which have elements with more than 13 nodes on *any element*, the sets of records following the command are given as:

```
N, NG, MA, (ND_i, i=1,13)
           (ND_i, i=14,29)
            ...
           (ND_i, i=..,NEN)
```

That is, each record must contain no more than 16 items of data as mentioned in Chapter 4. WARNING: When some elements in the set have fewer nodes needed to define the connection list it is still necessary that each element description have the same number of records (extra records may be blank). For this case it is recommended to split the element sets into groups with the same number of element nodes using the general form of the ELEMent command.

This ELEMent TYPE parameter is useful to assist *FEAP* prepare graphical representation for plots or outputs to ParaView. If multiple element types are used it is recommended that each type appear in a separate set, each specifying the specific TYPE parameter for the set. If desired, each set can be numbered starting with element 1, with the first ELEMent set preceded by the *AUTO command. Table 5.1 shows types of elements supported in the *FEAP* graphics and those that can be exported for ParaView display. See the ELEMent command section in Appendix A for additional details.

The element numbers following each ELEMent command *must be in increasing numerical order*. If gaps appear in consecutive records for the number of the element the missing elements will be generated by adding the generation value NG to each non-zero ND_i of the preceding element. Thus, the pair of records:

TYPE Parameter	Nodes/ Element	Feap Value	ParaView Value
LINE	2	-1	3
	3	-1	21
TRIAngle	3	-2	5
	6/7	-2	22
	10	-2	-
QUADrilateral	4	-3	9
	8/9	-3	23
	12/16	-3	-
TETRahedron	4	-4	10
	10	-4	24
HEXAhedron	8	-5	12
	20/27	-5	25
	64	-5	-
WEDGe	6	-6	13
PYRAmid	5	-7	14

Table 5.1: Element TYPE specification on connectivity input.

```
M, MG, MA, (MD_i, i=1,NEN)
N, NG, NA, (ND_i, i=1,NEN)
```

with `N - M` > 0 will generate the records:

```
M+1, -, MA, (MD_i+MG  , i=1,NEN)
M+2, -, MA, (MD_i+MG*2, i=1,NEN)
....
N-1, -, MA,  ......
```

until element `N` is reached. Using this form, care must be given to not generate a node number larger than `NUMNP`.

If the generation increment for nodes are not all the same (i.e., equal to `MG` and NG shown above) the generation may be set on the `ELEMent` record between the square braces `[` and `]`. For example, on a set of 6-node wedges the increments could be

```
ELEM NODE=6 [2 2 2 2 1 1]
```

Note that a separator may be a blank, comma or equal sign on any record. The above indicates that the first 4 nodes will be incremented by 2 and the last 2 nodes by 1.

Example:

Element mesh data for the curved line shown in Figure 5.1 is given by:

```
ELEMents
```

```
 1   1   1   1   2
10   0   1  10  11
                  ! Termination record
```

The mesh produced by this set of commands is shown in Figure 5.3

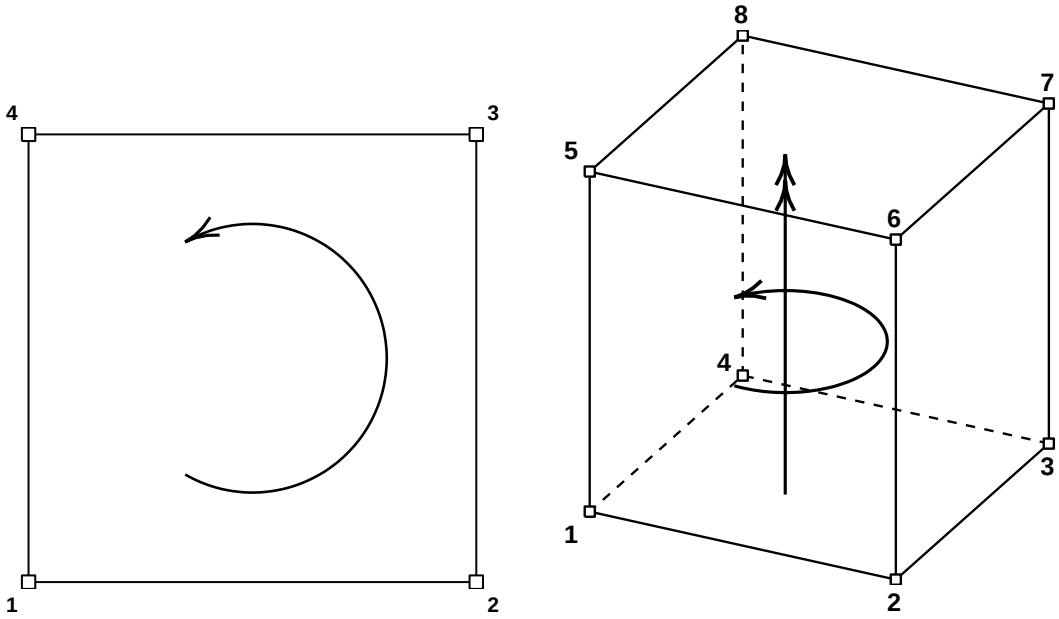
The elements included in *FEAP* are input with nodal connections numbered by *right hand rule* as indicated in Fig. 5.2 for a two-dimensional 4-node quadrilateral element and a three-dimensional 8-node brick element. Users may check that elements are properly numbered using the `CHECk` and/or `PLOT MESH` *solution command*. WARNING: Failure to number elements correctly results in stiffness and residual arrays with incorrect algebraic signs in their individual terms.

Element data is preprocessed to determine the total number of elements, `NUMEL` ,in a mesh. Element data may also be defined using the `BLOCk` and `BLENd` commands (see below).

When no generation is required to input all the element information the option

```
ELEMent ALL
  1 0  M_1 N1_1 N1_2 ...
  2 0  M_2 N2_1 N2_2 ...
       ..
```

or if the generation field is omitted by



(a) 2-d 4-node quadrilateral (b) 3-d 8-node brick

Figure 5.2: Right hand rule numbering of element nodes

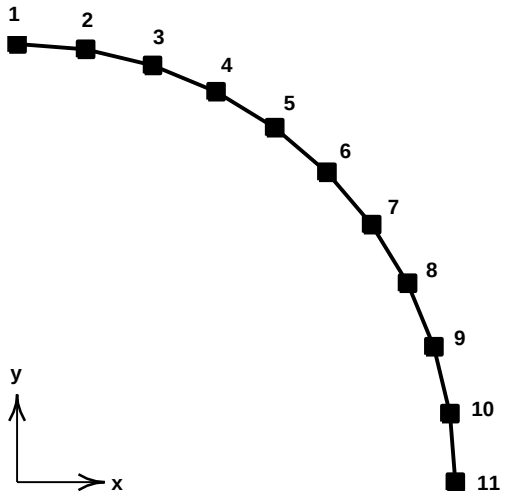


Figure 5.3: Mesh for Curved Beam. 10 Elements

```
ELEMent ALL
  NOGEneration
  1  M_1 N1_1 N1_2 ...
  2  M_2 N2_1 N2_2 ...
      ..
```

should be used. Similar to coordinate input the element connection data may be placed in a separate file and input using an `INCLude` option. In this option all the data *must* be given as constants – with no parameters or expressions permitted. Usually, this form of data results when the data is created by an external mesh generation program. If both elements and coordinates use the `ALL` option the NOCOunt option should be employed as described in Section 5.1.

5.2.3 BLOCk command for nodes & elements

Regular patterns of nodes and elements may be input using the `BLOCk` command. The block command can input patches of line elements (e.g., truss or frame elements); triangular and quadrilateral elements for surface elements and three dimensional hexahedral (brick) or tetrahedral elements for volume element types.

The data to input a *line* of elements is defined as:

```
BLOCk
  ctype,r_inc,,<node1>,<elmt1>,<mat>,r_skip,<b_type>
  <LINE n_e>
  <MATErial mat>
  1,X_1,Y_1,Z_1
```

```
   ...
N,X_N,Y_N,Z_N
          ! Termination record
```

where `ctype` is the coordinate type definition for the block master nodes and may be `CARTesian` (default), `POLAr`, `CYLIndrical` or `SPHErical`. The first record is followed by a set of master node numbers and coordinates with ordering as defined for the line, element types given in Section 3.1. The parameter `n_e` defines the number of nodes on the line element. For example `n_e = 3` generates 3-node line elements.

The data to input a patch of *triangular or quadrilateral* element types is defined as:

```
BLOCk
  ctype,r_inc,s_inc,<node1>,<elmt1>,<mat>,r_skip,<b_type>
  <LAYEr dir_b>
    < ma_1 ma_2 ....>
  <[TRIAngle,QUADrilateral] n_e g_type>
  <MATErial mat>
  1,X_1,Y_1,Z_1
     ...
  N,X_N,Y_N,Z_N
          ! Termination record
```

Node ordering is defined as for the quadrilateral element types defined in Section 3.2. The parameter `n_e` defines the number of nodes on the triangle or quadrilateral. Generally, elements up to cubic order may be generated using this option.

The data to input a three dimensional block of *hexahedral or tetrahedral* elements are defined as:

```
BLOCk
  ctype,r_inc,s_inc,t_inc,<node1>,<elmt1>,<mat>,<b_type>
  <LAYEr dir_b>
    < ma_1 ma_2 ....>
  <[TETRahedron BRICk] n_e>
  <MATErial mat>
  1,X_1,Y_1,Z_1
     ...
  N,X_N,Y_N,Z_N
          ! Termination record
```

Node ordering is defined as for the brick element types defined in Section 3.3.[1].

[1]The node numbering on the block has changed with Version 8.3. The numbering as shown in the Mesh Command BLOCk command of previous manuals may be used by giving the block command as `BLOCk OLD`

The parameters of the `BLOCk` command are defined in Tables 5.2 to 5.4. The type of elements to be generated are specified by **either** the `b_type` (see Table 5.3) **or** the shape (`TRIA`, etc.) record (see Table 5.4).

The optional records for `LAYEr` or `MATE` permit specification of different material properties or uniform material type, respectively. `LAYEr` and `MATErial` may not both be given. If the `LAYEr` option is used the parameter `dir_b` defines the direction in the block for the various material numbers and the record must be immediately followed by a set of records (of length 16 items or less) defining the material set number for each layer.

`Type`	- Master node coordinate type `CART` or `POLA`, `CYLI`, or `SPHE`.
`r_inc`	- Number of nodal increments to be generated along r-direction of the patch.
`s_inc`	- Number of nodal increments to be generated along s-direction of the patch.
`t_inc`	- Number of nodal increments to be generated along t-direction of the patch (N.B. Not input for 2-d).
`node1`	- Number to be assigned to first generated node in patch (default = automatic). First node is located at same location as master node 1.
`elmt1`	- Number to be assigned to first element generated in patch; if zero no elements are generated (default = automatic)
`matl`	- Material identifier to be assigned to all generated elements elements in patch (default = 1 or last input value)
`r_skip`	- For surfaces, number of nodes to skip between end of an r-line and start of next r-line (default = 1) (N.B. Not input for 3-d).

Table 5.2: `BLOCk` coordinate and size specification.

An example mesh input using the `BLOCk` command is the line elements shown in Figure 5.3. For two node elements the necessary data is:

```
BLOCk
  POLAr 10
    LINE 2
    MATE 1
      1   5.0   90.0
      2   5.0    0.0
           ! Termination record
```

Two Dimensional Elements	
`b_type`	=0: 4-node elements on surface patch; 2-node elements on a line; 8-node elements in a block;
	=1: 3-node triangles (diagonals in 1-3 direction of block);
	=2: 3-node triangles (diagonals in 2-4 direction of block);
	=3: 3-node triangles (diagonals alternate 1-3 then 2-4);
	=4: 3-node triangles (diagonals alternate 2-4 then 1-3);
	=5: 3-node triangles (diagonals in union-jack pattern);
	=6: 3-node triangles (diagonals in inverse union-jack pattern);
	=7: 6-node triangles (similar to =1 orientation);
	=8: 8-node quadrilaterals (*r-inc* and *s-inc* must be even numbers); N.B. Interior node generated but not used;
	=9: 9-node quadrilaterals (*r-inc* and *s-inc* must be even
	=16: 16-node quadrilaterals (*r-inc* and *s-inc* must be multiples of three);
Three Dimensional Elements	
	=10: 8-node hexahedra (bricks).
	=11: 4-node tetrahedra.
	=12: 27-node quadratic hexahedra. (r-, s-, t-inc must be even numbers)
	=13: 10-node tetrahedra. (r-, s-, t-inc must be even numbers)
	=14: 20-node quadratic hexahedra. (r-, s-, t-inc must be even numbers)
	=15: 11-node quadratic tetrahedra. (r-, s-, t-inc must be even numbers)
	=17: 14-node quadratic tetrahedra. (r-, s-, t-inc must be even numbers)
	=18: 15-node quadratic tetrahedra. (r-, s-, t-inc must be even numbers)

Table 5.3: `BLOCk` element type specification using `b-type`

When using the `BLOCk` command one may enter zero for the *Node1* and *Elmt1* parameters. Values for the node and element numbers will then be automatically generated in the sequence data is input. Restrictions apply when mixing `BLOCk` or `BLENd` options with the `ELEM` option where numbers are required.

While polar coordinates may be used directly as input for the block master coordinates using the `POLAr` option, the actual nodal coordinates generated will be converted automatically from polar to Cartesian coordinates using the current `SHIFt` command

Type	n_e	g_type	Description
LINE	0 or 2	-	2-node line element
	3	-	3-node line element
TRIA	0 or 3	1	3-node triangular element (ll to ur)
	0 or 3	2	3-node triangular element (ul to lr)
		n	(b-type same as above)
	6	1	6-node triangular element (ll to ur)
	6	2	6-node triangular element (ul to lr)
	7	1	7-node triangular element (ll to ur)
	7	2	7-node triangular element (ul to lr)
	10	-	10-node triangular element (ll to ur)
QUAD	0 or 4	-	4-node quadrilaterals
	8	-	8-node quadrilaterals
	9	-	9-node quadrilaterals
	16	-	16-node quadrilaterals
TETR	0 or 4	-	4-node tetrahedral elements
	10	-	10-node tetrahedral elements
	11	-	11-node tetrahedral elements
	14	-	14-node tetrahedral elements
	15	-	15-node tetrahedral elements
BRIC	8	-	8-node hexahedral elements
	20	-	20-node hexahedral elements
	27	-	27-node hexahedral elements
	64	-	64-node hexahedral elements

Table 5.4: BLOCk element type specification using LINE, TRIA, QUAD, TETR or BRIC subcommand.

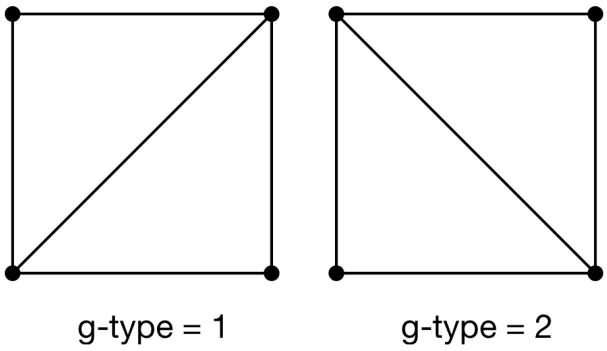
values for x_0, y_0, and z_0 (see Section 5.3).

With this option it also is not necessary to know the numbers for the generated nodes, as was required to use the COORdinate and POLAr commands. For three dimensional problems both the POLAr and CYLIdrical options becomes a cylindrical coordinate transformation. For three dimensional problems, it is also possible to use a *spherical* coordinate transformation using the SPHErical option in place of the CARTesian or POLAr forms.

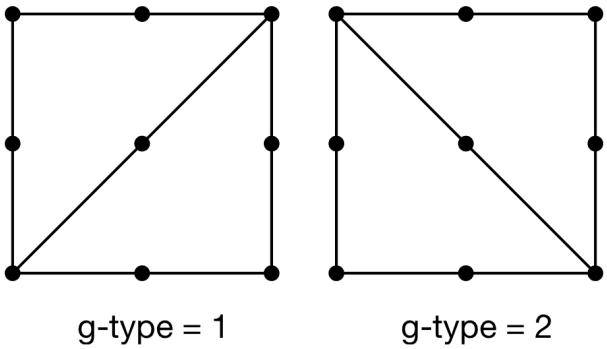
5.2.4 TRIBlock command for nodes & elements

Instead of a rectangular block of elements in two-dimensions, it is possible to generate a triangular region of triangular elements using the command set:

```
TRIBlock
  ctype,r_inc,<node1>,<elmt1>,<mat>,<b_type>
  <TRIAngle [3,6,7]>
  <MATErial mat>
  1,X_1,Y_1,Z_1
     ...
  N,X_N,Y_N,Z_N
```



Linear 3-node triangular meshes



Quadratic 6- and 7-node triangular meshes

Figure 5.4: Two-dimensional triangular mesh orientations

```
                ! Termination record
```

where `ctype` may be `cart, pola, cyli, sphe` and `btype` is `1, 2, 3` for 3-node, 6-node, or 7-node triangles. An example to generate 6 node triangle is

```
TRIBlock
  cart 10
  triangle 6
  material 2
    1  x1 y1
    2  x2 y2
    3  x3 y3
          !  Termination record
```

Warning: Node numbering for 7-node elements creates a poor profile for the default direct solver. It is recommended that the profile optimizer be used by issuing the commands

```
BATCh
   OPTImize profile
END
```

This usually is not necessary when iterative, sparse or user solvers are employed.

5.2.5 BLENd command for nodes & elements

A block of nodes and elements also may be generated using a blending function approach (e.g., see ,[19] pp 226 or ,[20] pp 181). In *FEAP* the blending function meshes are created from a set of control points (called *super-nodes*) input using the `SNODe` command, edges input using the `SIDE` command and a description of the region using the `BLENd` command. Meshes may be created as `SURFaces` in two and three dimensions or as `SOLIds` in three dimensions.

Super-nodes: SNODe command

The coordinates for super-nodes *always* are given in Cartesian form. The input form is given as:

```
SNODes
  N  X_N  Y_N  Z_N
   ...
            ! Blank termination record
```

where `N` is the super-node number and is sequenced from 1 to the maximum number needed to describe all blending functions. No generation is available for super-node input.

If loops are used to construct a mesh all `SNODe` definitions should be placed outside any `LOOP-NEXT` pairs (see Section 9.3 for more information on use of loops).

Sides of blending function regions: SIDE command

The sides of any surface and the edges of any solid to be generated by the blend command must be prescribed. Only sides for non-straight or non-uniformly spaced increments need be given. *FEAP* will automatically add all straight uniformly spaced sides not given as input data. The specification of sides using the `SIDE` command is given by the general form:

```
SIDE
  Type  V1,V2,V3,....,V14
```

where `Type` is the geometric type for the side, and `Vi` are a list of values. Sides are one of three different `Type`s:

- `Type = CARTesian`: Lagrange interpolation in Cartesian coordinates. The `Vi` values are numbers of super-nodes used for the interpolation

$$\mathbf{x}(\xi) \;=\; \sum_i L_i(\xi)\mathbf{x}_{Vi}$$

  where $L_i(\xi)$ are Lagrange interpolation polynomials in the natural coordinate ξ. End points for a Cartesian side are `V1` and `V2` with interior points numbered from `V3` to `Vn` starting from the `V1` end.

- `Type = POLAr`: Lagrange interpolation in polar (or cylindrical) coordinates. The interpolations are given as:

$$r(\xi) \;=\; \sum_i L_i(\xi) r_{Vi}$$
$$\theta(\xi) \;=\; \sum_i L_i(\xi) \theta_{Vi}$$

  where the radii r_{Vi} use the last specified super-node number in the list for `Vi` for the location of their origin. Other super-node numbers are given as for the Cartesian type.

- `Type = SEGMent`: Multiple straight segments with uniform increments on each segment. In this form the odd entries `V1, V3, V5, ...` are super-node numbers and the even entries `V2, V4, V6, ...` are the number of increments between the adjacent super-nodes.

If loops are used to construct a mesh all SIDE definitions should be placed outside any LOOP-NEXT pairs (see Section 9.3).

BLENd command

For two-dimensional blended meshes the SURFace option is used and four vertex super-nodes specify the orientation of the region. The super-nodes must be given as an anti-clockwise sequence (right hand rule). The input is given as:

```
BLENd
  SURFace inc_1 inc_2 <node1> <elem1> <mat1> <etype>
  <LAYEr dir_b>
    < ma_1 ma_2 ....>
  <[TRIAngle QUADrilateral TETRahedron BRICk] Etype>
  <MATErial Mat1>
    s1 s2 s3 s4
```

where the parameters are defined in Table 5.5.

The optional records for LAYEr or MATE permit specification of different material properties or uniform material type, respectively. LAYEr and MATErial may not both be given. If the LAYEr option is used the parameter dir_b defines the direction in the block for the various material numbers and the record must be immediately followed by a set of records (of length 16 items or less) defining the material set number for each layer.

Type	- Blend type (SURFace).
inc_1	- Number of nodal increments to be generated along 1-2 edge.
inc_2	- Number of nodal increments to be generated along 2-3 edge.
node1	- Number to be assigned to first generated node in patch (default = automatic). First node is located at same location as master node 1.
elmt1	- Number to be assigned to first element generated in patch; if negative no elements are generated (default = automatic)
matl	- Material identifier to be assigned to all generated elements in patch (default = 1)
etype	- Element type (same as b_type in BLOCk command (default is 4-node quadrilateral elements)

Table 5.5: Surface Blend Parameters

The two dimensional blended mesh shown in Figure 5.5 has three straight sides and one circular arc side. The spacing along each side is uniform, thus only end points are required to specify the control points. For non-uniform spacing additional control points may be given for edges. To construct this mesh the coordinates for the five super-nodes, the one arc edge, and the vertices for the blend region must be specified as shown in Figure 5.6.

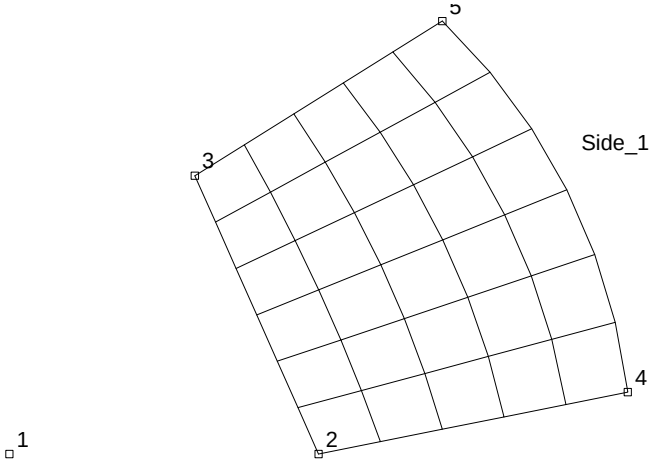


Figure 5.5: Two-dimensional Blended Mesh

```
SNODes
  1  0   0
  2  5   0
  3  3   4.5
  4 10   1
  5  7   7
            ! Blank termination record
SIDE
  POLAr  4  5  1
            ! Blank termination record
BLENd
  SURFace  5  6
    QUAD 4
    MATE 1
      2  4  5  3
            ! Blank termination record
```

Figure 5.6: Two-dimensional blended mesh data

For three-dimensional blended meshes either the SURFace or the SOLId option may be used to generate the mesh region. For the SURFace option the ordering is any

contiguous four super-node sequence and input is identical to that shown above (except super-nodes must have z-coordinate values). For the SOLId option the vertex order is identical to that for the 8-node BLOCk command: That is, number the super-nodes by right hand rule with the first four nodes on the *bottom* face and the last four on the *top* face. The input is given as:

```
BLENd
  SOLID inc_1 inc_2 inc_3 <node1> <elem1> <mat1> <etype>
  <LAYEr dir_b>
    < ma_1 ma_2 ....>
   <[TETRahedron BRICk] Etype>
   <MATErial Mat1>
     s1 s2 s3 s4 s5 s6 s7 s8
```

where the parameters are defined in Table 5.6.

The optional records for LAYEr or MATE permit specification of different material properties or uniform material type, respectively. LAYEr and MATErial may not both be given. If the LAYEr option is used the parameter dir-b defines the direction in the block for the various material numbers and the record must be immediately followed by a set of records (of length 16 items or less) defining the material set number for each layer.

Type	- Blend type SOLId.
inc_1	- Number of nodal increments to be generated along 1-2 edge.
inc_2	- Number of nodal increments to be generated along 2-3 edge.
inc_3	- Number of nodal increments to be generated along 1-5 edge.
node1	- Number to be assigned to first generated node in patch (default = automatic). First node is located at same location as master node 1.
elmt1	- Number to be assigned to first element generated in patch; if negative no elements are generated (default = automatic)
matl	- Material identifier to be assigned to all generated elements elements in patch (default = 1)
etype	- Element type (same as b_type in BLOCk command (default is 8-node brick elements)

Table 5.6: Three-dimensional Solid Blend Parameters

A blended region for a three dimensional mesh is shown in Figure 5.7 and generated using the data shown in Figure 5.8.

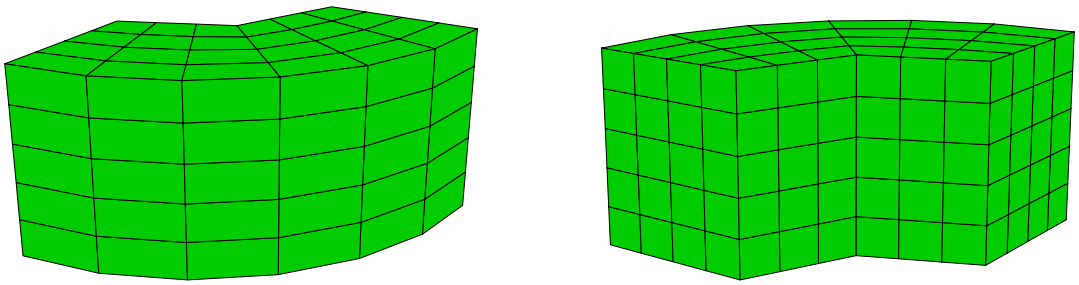


Figure 5.7: Three-dimensional Blended Mesh

```
SNODes
  1    0   0  0
  2   10   0  0
  3    0  10  0
  4    5   0  0
  5  3.5 3.5  0
  6    0   5  0
  7    0   0  6
  8   10   0  6
  9    0  10  6
 10    5   0  6
 11  3.5 3.5  6
 12    0   5  6
           ! Blank termination record
SIDEs
  POLA   2  3  1
  SEGM   4  3  5  3  6
  POLA   8  9  7
  SEGM  10  3 11  3 12
           ! Blank termination record
BLEND
  SOLID  6  4  5
    BRICk 8
    MATE  1
      2  3  6  4  8  9 12 10
           ! Blank termination record
```

Figure 5.8: Three-dimensional blended mesh data

Nodes and elements may be generated using a combination of the above schemes. Thus, it is possible to mix the BLOCk and BLENd options with the COORdinate and ELEMent commands to generate the mesh. Furthermore, the mesh may be described using any

of the coordinate systems as inputs and subsequently (or in the case of the BLOCk and BLENd options simultaneously) converting the input and/or generated coordinates to Cartesian coordinate values using the POLAr or SPHErical commands.

5.3 Coordinate and transformation systems

The mesh coordinates in *FEAP* must all be given in a Cartesian system. Input data, however, may be specified in *Cartesian*, *polar* or *cylindrical* or *spherical* coordinate systems.

5.3.1 POLAr, CYLIndrical, SPHErical and SHIFt commands

When polar, cylindrical or spherical coordinates are used to define the nodal data using the COORdinate command, they may be transformed to the required Cartesian form using the POLAr or SPHErical commands, respectively. Nodal coordinates generated with polar, cylindrical or spherical options in the BLOCk command do not require transformation. The data for a polar command is:

```
POLAr
  NODE n1 n2 inc
```

where n1 and n2 define a range of nodes and inc is the increment to be added to n1 for each step to n2. Alternatively, all currently defined nodes may be transformed using the command

```
POLAr
  ALL
```

The transformation is given by

$$x_1 = x_0 + r \cos\theta$$
$$x_2 = y_0 + r \sin\theta$$

and for 3-dimensional problems in cylindrical coordinates

$$x_3 = z_0 + z$$

where x_i are the Cartesian coordinates, r, θ, z are the polar (cylindrical) inputs, and x_0, y_0, z_0 are shifts defined by the SHIFt command given as

```
SHIFt
  X_0,Y_0,Z_0
```

By default x_0, y_0, z_0 are zero.

The SPHErical command is similar to the POLAr command. The input records are specified as:

```
COORdinate
  N NG R THETA PHI
```

Transformations use the relations

$$x_1 = x_0 + r \cos\theta \sin\phi$$
$$x_2 = y_0 + r \sin\theta \sin\phi$$

and

$$x_3 = z_0 + r \cos\phi$$

This is followed by use of the command set

```
SPHErical
  NODE n1 n2 inc
```

which is interpreted in a manner identical to the POLAr command. The SHIFt command may also used to locate the origin for spherical transformations.

5.3.2 Coordinate transformation

Cartesian systems may be translated, stretched, reflected and/or rotated using the TRANsform command. Any coordinates input after this command are transformed using

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

where $\hat{x}_i$ are the input values and the transformation parameters are defined by the command sequence

```
TRANsform
  T_11  T_12  T_13
  T_21  T_22  T_23
  T_31  T_32  T_33
  X_0   Y_0   Z_0
```

which must appear before any coordinates (i.e., the $\hat{x}_i$) are specified.

The TRANsform command may be used as many times as needed. In particular, it may be used with a portion of a mesh (substructure) in an include file to replicate repeated parts of meshes (see Chapter 9). When a reflection is performed, *FEAP* notes the coordinate transformation does not have a positive determinant and (if possible) resequences the node numbers on elements to maintain a positive Jacobian (provided the original data is correct in its local Cartesian basis - $\hat{x}_i$). However, users should always use CHECK and PLOT MESH commands to ensure the mesh is correct.

5.4 Extruding meshes: SWEEp command

The `SWEEp` mesh command may be use do extrude a two dimensional mesh into a 3-dimensional mesh. The input for for the command is given by the sequence

```
SWEEp
 <filename for 2-d mesh> ! Must be a flat file with all data
 sweep_type nseg ndir
 CENTer x_0 y_0 z_0
   1 x_1 y_1 z_1
   2 x_2 y_2 z_2
    ...
   n x_n y_n z_n
```

The parameter `sweep_type` must be either `CARTesian` or `CYLIndrical`. The parameters `(x_n, y_n, z_n)` define the `super-nodes` (viz. Sect. 5.2.5) for the `sweep_type` interpolation

$$\mathbf{x}'(\xi_i) = \sum_{a=1}^{n} N_a(\xi_i)\,\tilde{\mathbf{x}}^a \quad ; \quad \xi_i = 1, ..., nseg + 1$$

The *flat file* for the 2-d mesh may be generated by executing the solution command

```
OUTM
```

5.4.1 Cartesian type coordinates

The nodal coordinates in the 3-d mesh are computed from those in the 2-d mesh using

$$\hat{\mathbf{x}}^{(3d)} = \mathbf{x}_0 + \mathbf{x}'(\xi_i) + \mathbf{x}^{(2d)}$$

where $\mathbf{x}^{(2d)} = (x^{(2d)}, y^{(2d)}, 0)$. Finally, the $\hat{\mathbf{x}}^{(3d)}$ are transformed to the final values using

$$\mathbf{x}^{(3d)} = \mathbf{T}\,\hat{\mathbf{x}}^{(3d)} + \bar{\mathbf{x}}$$

where $\mathbf{T}$ and $\bar{\mathbf{x}}$ are defined by the coordinate `TRANsform` command data. The total number of nodes generated will be $numnp = numn2 * (nseg + 1)$, where $numn2$ is the number of nodes in the 2-d mesh.

5.4.2 Cylindrical type coordinates

A circular cylindrical sweep of the mesh may be generated by either a rotation about the $x_1 = x^{(2e)}$, $x_2 = y^{(2e)}$ or normal (i.e., x_3) axis of the 2-d mesh where the parameter *ndir* defines the 1, 2, 3 or 4 direction.

The *ndir* cylindrical interpolations are computed from inputs with `x_n = r_n`, `y_n = theta_n` and `z_n = z_n`. Thus $\mathbf{x}' = (r', \theta', z')$.

For rotation about the x_1 axis the transformations are computed as

$$\begin{Bmatrix} \hat{x}^{(3d)} \\ \hat{y}^{(3d)} \\ \hat{z}^{(3d)} \end{Bmatrix} = \begin{Bmatrix} x_0 \\ y_0 \\ z_0 \end{Bmatrix} + \begin{Bmatrix} (z' + x^{(2d)}) \\ (r' + y^{(2d)})\ \cos(\pi\,\theta'/180) \\ (r' + y^{(2d)})\ \sin(\pi\,\theta'/180) \end{Bmatrix}$$

For rotation about the x_2 axis the transformation is given by

$$\begin{Bmatrix} \hat{x}^{(3d)} \\ \hat{y}^{(3d)} \\ \hat{z}^{(3d)} \end{Bmatrix} = \begin{Bmatrix} x_0 \\ y_0 \\ z_0 \end{Bmatrix} + \begin{Bmatrix} (r' + x^{(2d)})\ \cos(\pi\,\theta'/180) \\ -(r' + x^{(2d)})\ \sin(\pi\,\theta'/180) \\ (z' + y^{(2d)}) \end{Bmatrix}$$

The rotation about the x_3 axis yields a 3-d mesh in which the 2-d mesh can vary in size and be twisted along the x_3 axis. The radius (`r_n` $= r'$) scales the 2-d mesh to vary along the z-axis. To perform the transformation a radius and angle from $\mathbf{x}_0$ is computed as

$$\begin{aligned} r^{(2d)} &= r'\ \left[(x^{(2d)} - x_0)^2 + (y^{(2d)} - y_0)^2\right]^{1/2} \\ \theta^{(2d)} &= \tan^{-1}(y^{(2d)} - y_0, x^{(2d)} - x_0) + \pi\,\theta'/180 \end{aligned}$$

and used to compute the final coordinates as

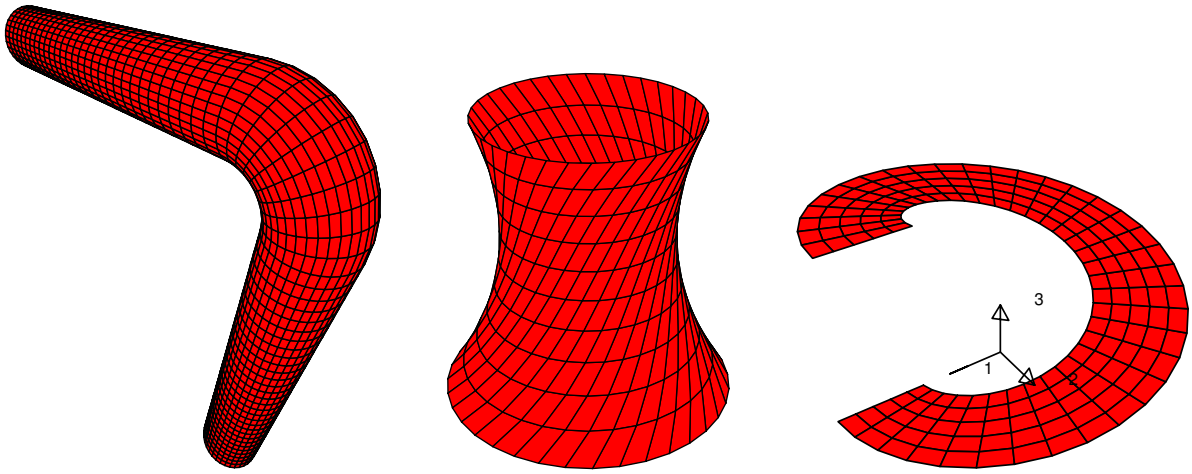
$$\begin{Bmatrix} \hat{x}^{(3d)} \\ \hat{y}^{(3d)} \\ \hat{z}^{(3d)} \end{Bmatrix} = \begin{Bmatrix} x_0 \\ y_0 \\ z_0 \end{Bmatrix} + \begin{Bmatrix} r^{(2d)}\cos(\theta^{(2d)}) \\ r^{(2d)}\sin(\theta^{(2d)}) \\ z' \end{Bmatrix}$$

The $\hat{\mathbf{x}}^{(3d)}$ are transformed to $\mathbf{x}^{(3d)}$ using $\mathbf{T}$ and $\bar{\mathbf{x}}$ in an identical manner as that for the Cartesian form.

The `ndir` 4 cylindrical interpolation rotates the 2-d mesh about the $y^{(2d)}$ axis. The sweep coordinates are given by $x_n = \theta'$ and $y_n = z'$. The coordinates of the 3-d mesh are computed from

$$\begin{Bmatrix} \hat{x}^{(3d)} \\ \hat{y}^{(3d)} \\ \hat{z}^{(3d)} \end{Bmatrix} = \begin{Bmatrix} x_0 \\ y_0 \\ z_0 \end{Bmatrix} + \begin{Bmatrix} x^{(2d)}\cos(\pi\,\theta'/180) \\ x^{(2d)}\sin(\pi\,\theta'/180) \\ y^{(2d)} + z' \end{Bmatrix}$$

All the $\hat{\mathbf{x}}^{(3d)}$ may be transformed to $\mathbf{x}^{(3d)}$ using $\mathbf{T}$ and $\bar{\mathbf{x}}$ in an identical manner as that for the Cartesian form.



(a) Pipe bend. (b) Cylinder twist. (c) Spiral.

Figure 5.9: Examples of 2-d meshes extruded to form 3-d meshes.

5.4.3 Elements types in the 3-d mesh

At present, all elements in the 2-d mesh must be of linear order. The elements in the 3-d mesh will be generated from the 2-d elements with a linear increment in the sweep direction, thus, the number of elements in the 3-d mesh will be $numel = nume2 * nseg$. With this option, point elements in the 2-d mesh become 2-node line elements, 2-node line elements become 4-node quadrilateral elements, and 4-node quadrilateral elements become 8-node brick elements.

Example 1:

As an example, the result shown in Fig. 5.9(a) is generated using the command sets

```
SWEEp
  Icirc.rev
  CART 30 0
  CENT  0 0 0
     1 0 0 -100
     2   0 0  0
       ! Termination record
SWEEp
  Icirc.rev
  CYLI 10 1
  CENT  0 -20 0
     1 20 0 0 0
     2 20 90 0
       ! Termination record
```

```
TRAN
  -1  0  0
   0  0  1
   0  1  0
   0 -20 20
       ! Termination record
SWEEp
  Icirc.rev
  CART 30 0
  CENT 0 0  0
     1 0 0 -100
     2 0 0   -0
       ! Termination record
```

The file `Icirc.rev` defined the circular cross-section as 2-node elements (the extender `rev`) results from use of an `OUTMesh` command to produce the flat data file.

Example 2:

As a second example, the mesh shown in Fig. 5.9(b) was produced using the command set

```
SWEEp
  Icirc.rev
  CYLI 10 3
  CENT  0 0 0
     1 1.0    0  0
     2 0.75  90 20
     3 0.6   45 10
       ! Termination record
```

Example 3:

As a third example, the mesh shown in Fig. 5.9(c) was produced using the command set

```
SWEEp
  Icc.rev
  CYLI 36 4
  CENT  0 0 0
    1   0.0  0.0
    2  360.0 10.0
       ! Termination record
```

The mesh `Icc.rev` was produced using the 2-d block generation

```
BLOCk
  CART 5
    LINE 2
      1  5.0 0.0
      2 10.0 0.0
       ! Termination record
```

and output using a `OUTM` solution command.

5.5 Nodal boundary conditions

The basic *FEAP* boundary condition quantities are values for non-zero nodal *forces* and nodal *displacements*. For problems in solid mechanics these terms have physical meaning; however, for general classes of problems forces and displacements are interpreted in a *generalized* sense - e.g., as flux and dependent variable pairs. Non-zero values for forces and displacements may be input at each node. It is not necessary to input conditions for any node where all components are zero. The actual condition to be imposed (i.e., force or displacement) is determined by the active values of the *boundary restraint condition*. A non-zero value of a boundary restraint condition for a degree-of-freedom implies that the value of the specified nodal displacement is to be imposed; whereas, a zero value implies that the value of the specified nodal force (flux) is to be applied. Generally, these quantities are specified by components associated with directions in the global Cartesian mesh coordinates. It is possible, however, to specify components which are associated with directions different than the global mesh coordinate. Options exist to specify the orientation by an angle in the $x-y$ plane (see command `ANGLe` below), by Euler angles (see command `EULEr` below) or by a general triad of direction cosines (see command `TRIAd` below). At present a set of coordinates which are described by a rotation angle (in degrees) about the x_3 axis with respect to the x_1 axis. The input of boundary condition quantities associated with nodes may be specified based on: (a) Node numbers; (b) Nodal coordinate values; or (c) Edge coordinate values.

5.5.1 Nodal input form.

The options to input a nodal quantity associated with boundary conditions are shown in Table 11.1. The use of a nodal form (i.e., `BOUNdary, FORCe, DISPlacement, ANGLe, EULEr, TRIAd`) implies a specification using a *node number*. Other options do not require node numbers and are preferred when possible. These forms also apply to the *real* part of complex form with the *imaginary* part input using the command forms `IFORce` and `IDISpl`.

Type	Boundary	Forces	Displacements	Angle
Nodal real	BOUNdary	FORCe	DISPlacement	ANGLe, EULEr, TRIAd
Nodal imag		IFORce	IDISplacement	
Edge	EBOUndary	EFORce	EDISplacement	EANGle
Coordinate	CBOUndary	CFORce	CDISplacement	CANGle
Material	MBOUndary			

Table 5.7: Nodal Boundary Condition Quantity Inputs

An example of the use of the nodal option for input of a force in the 1- and 2-direction on node 19 is given by:

```
FORCe
  19  0    5.0     10.0
                      ! Termination record
```

Here the 1-component has a value of 5.0 and the 2-component a value of 10.0, respectively. For a problem in complex arithmetic the default mode is the *real* part of the force. To add the imaginary part the form

```
IFORce
  19  0   -1.0      0.0
                      ! Termination record
```

may be used. A similar form applies to specification of DISPlacement values.[2] The input records for nodal FORCe, DISPlacement, BOUNdary condition, ANGLe, EULEr and TRIAd commands are similar to those for COORinates with the node and generation increment in the first two fields and the list of values for each degree-of-freedom in the remaining field. Thus, a nodal record is:

```
N_node G_inc V_1 V_2 ... V_n
```

where N_node is a node number G_inc the increment to the next node and V_i values of the list. The values of all arrays are set to zero at the start of each problem, hence only non-zero values need be specified for forces, displacements, boundary conditions and angles.

Similarly, the specification of the real part of a non-zero displacement at a node may be given using the command

```
DISPlacement
  19  0  0.0  -0.1
```

This sets the 1-component to zero and assigns the 2-component a value of -0.1 units. A similar form may be used for the imaginary part using the command IDISplacement.

[2]The forms EFOR, EDIS, CFOR, CDIS, CSUR currently do not allow the input of imaginary parts.

The value of a force or displacement will be selected based on the *boundary restraint code* value active at the time of execution. Non-zero boundary restraint codes imply a specified displacement and zero values a specified load. The boundary restraint codes may be set using the command

```
BOUNdary codes
  19  0  0  1
```

which states the first degree-of-freedom is a specified force (zero by default) and the second a specified displacement (again zero by default). Thus, if both of the above force and displacement commands are included both a non-zero force and a non-zero displacement will be used. During execution it is possible to change the boundary restraint codes and use different force/displacement combinations.

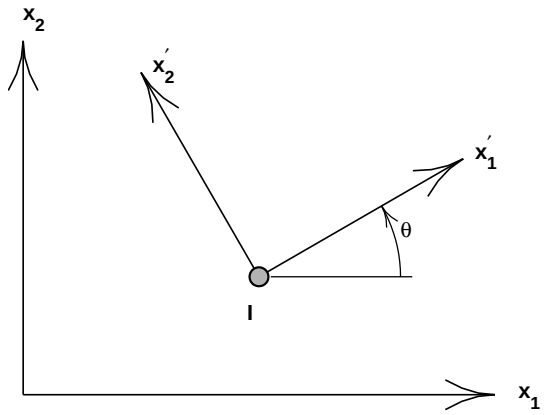


Figure 5.10: Angle boundary condition specification.

It is possible to set all boundary conditions to a fixed condition using the command

```
BOUNdary FIX
```

or

```
TBOUndary
```

This may be useful in checking consistency of an input model or to impose *Taylor* type boundary conditions on representative volume elements (RVE's).

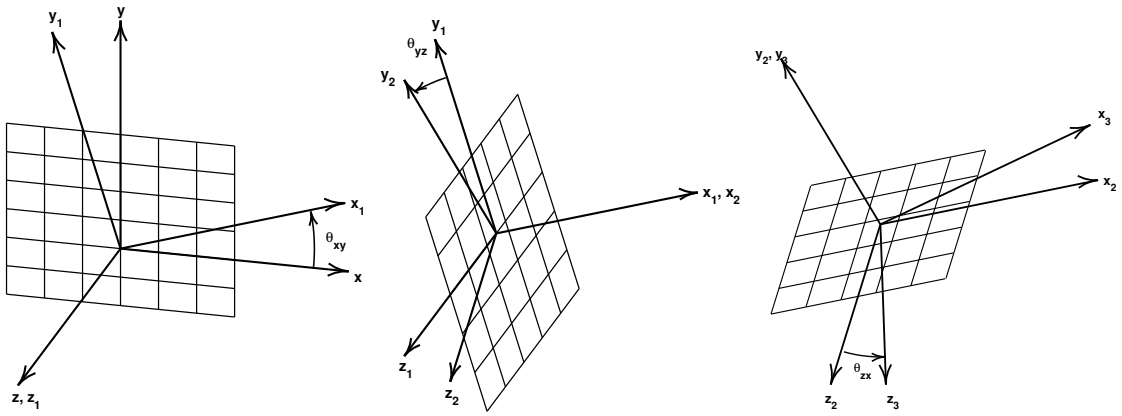
The specification of angle boundary conditions are given by specifying the angle in degrees the axis makes with the x_1 axis as shown in Fig. 5.10. The data is input using the commands

```
ANGLe
  19  0  30.0
        ...
```

where `19` denotes the node and `30.0` the angle. The angle boundary condition may be used in either two or three dimensional problems, however, the orientation is restricted to a rotation about the x_3 axis for three dimensional problems. An alternative for three dimensional problems that permits general rotations is the use of Euler angles to describe the orientation of the rotated axes. Figure 5.11 shows the sequence of rotations to be applied. The input is given using the command sequence

```
EULEr angles
  19  0  30.0 60.0 35.0
            ...
```

where $\theta_{xy} = 30$, $\theta_{yz} = 60$ and $\theta_{zx} = 35$, all angles being in degrees.



(a) Rotation θ_{xy} (b) Rotation θ_{yz} (c) Rotation θ_{zx}

Figure 5.11: Euler angle rotations for nodes.

The `TRIAd` option for inputs is given using the commands

```
TRIAds
   N_node G_inc t_1x t_1y t_1z t_2x t_2y t_2z t_3x t_3y t_3z
     ...
```

The three vectors $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_3$ form an orthonormal set of unit vectors to represent the nodal displacement components as (see Fig. 5.12)

$$\mathbf{u} = \sum_{j=1}^{3} \widehat{u}_j \, \mathbf{t}_j$$

For each transformed node the nodal boundary conditions are associated with the components $\widehat{u}_j$ instead of the Cartesian components u_j.

To use the nodal input option it is a users responsibility to determine the correct number for each node - often for two-dimensional mesh problems the graphics capability of *FEAP* can assist in determining the correct node numbers; however, for a very large number of forces this is a tedious method. Accordingly, there are two other options available to input nodal values.

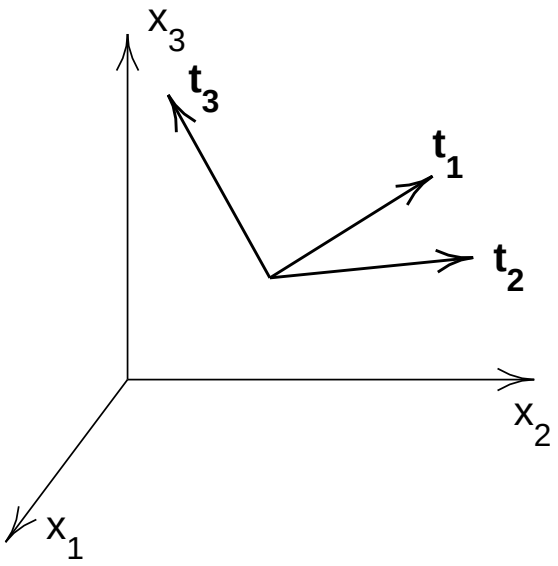


Figure 5.12: Triad description for 3-d displacement rotation of nodes.

5.5.2 Edge input form.

The second option available to specify the nodal quantities is based on coordinates and is used to apply a common value to all nodes located at some constant coordinate location called the *edge* value. The options `EBOUndary, EFORce, EDISplacement, EANGle` are used for this purpose. For example, if it is required to impose a zero displacement for the first degree of freedom of all nodes located at $y = 0.5$. The edge boundary conditions may be set using

```
EBOUndary
  2    0.5  1   0
                        ! Termination record
```

In the above the 2 indicates the second coordinate direction (i.e., x_2 or y for Cartesian coordinates) and 0.5 is the value of the x_2 or y coordinate to be used to find the nodes. The last two fields are the boundary condition pattern to apply to all the nodes located. That is, in the above we are indicating the first degree-of-freedom is to have specified nodal displacements and the second is to have specified nodal forces. *FEAP* locates all nodes which are within a small tolerance of the specified coordinate *after the mesh input is completed* (i.e., after the mesh `END` command is encountered).

By default the edge options will be appended to any previously defined data at a node by the pattern specified. If it is desired to *replace* the other set conditions, edge options are specified as:

```
EBOUndary SET
```

```
1    0.5  1   0
2    0.5  0   1
                    ! Termination record
```

By default, where no option is set, or with the inclusion of the `ADD` parameter the boundary restraint code at a node located at (0.5, 0.5) will be fully restrained (i.e., have both directions with a unit (1) restrained value). With the `SET` option as shown above the node located at (0.5, 0.5) would have only its second degree-of-freedom restrained.

5.5.3 Coordinate input form.

Using the options `CBOUndary, CFORce, CDISplacement, CANGle` indicates that the quantities are to be input based on the coordinates of a node. An example to specify a 10 unit force on the 2-component for a two-dimensional problem node located at $x = 4.0$ and $y = 5.0$ is given by:

```
CFORCe
  NODE 4.0  5.0    0.0     10.0
                       ! Termination record
```

This method will place the force on the node nearest the specified point. If two nodes have the same or equally close coordinate only one will have the force applied. While much easier, this method is still somewhat tedious if a large number of forces need to be applied. Options exist to generate the forces automatically for some distributed loading types (e.g., see Section 5.6).

Coordinate generated data will replace previously generated values unless the `ADD` parameter is added. Thus the final outcome of the above `CFORce` command would be to have a force value for the first degree-of-freedom of 10.0.

5.5.4 Material input form.

The boundary conditions may be set for a single degree of freedom for all nodes in a material set using the commands:

```
MBOUndary
  ma dof
```

where `ma` is the material set and `dof` the degree of freedom to fix. This command is useful for problems using the linear shell element in which the elements all lie in a plane (flat condition). For this case the degree of freedom associated with the rotation component normal to the plane has no stiffness and must be restrained.

5.5.5 Hierarchy of input forms.

The input of the nodal boundary data is performed by *FEAP* in a specific order. Data input in the nodal form is interpreted immediately after the data records are read. Values assigned by the nodal input replace any previously specified values - they are not accumulated.

The material option is processed first. Data input by the edge option is interpreted next and any coordinate specified data is then processed. By default the data is added to any previously specified information; however, if the data is specified in a `Exxx,SET` option the information is replaced. Multiple edge sets may be input and are interpreted later in the order they were encountered in the input file. Thus, use of the sequence of commands

```
EBOUndary,SET
  1 10.0  1 0
                        ! Termination record
EBOUndary,ADD (or blank)
  1  0.0  1 0
  2  0.0  0 1
                        ! Termination record
```

defines two data sets. The first will replace the boundary code definition for any node which has x_1 equal to 10.0 by a restrained first dof and an unrestrained second dof. Subsequently, the second set will restrain all the first dof at any node with x_1 equal to zero and also restrain the second dof at any node with x_2 equal to zero. Thus, if there is a node with (x_1, x_2) of (0.0, 0.0) the node will be fully restrained

After all edge data sets are processed the data defined by the coordinate option is processed. By default it is also interpreted in a `SET` mode unless the data set is defined by a `Cxxx,ADD` command.

When using the coordinate or edge options it is recommended that the graphics options in *FEAP* be used to check that all desired quantities are located. For the coordinate method other options are available to specify forces, displacements, and boundary conditions. These are described further in Appendix A.

5.5.6 Time dependent load functions

Each nodal force or displacement may be multiplied by a time dependent, *proportional* loading function. By default the sum of all proportional loads is used as the multiplying factor. Each load function is defined by the `PROP`ortional command during a solution phase. Each proportional loading record is defined by a number. Thus, the number for a proportional load varies from one (`1`) to a maximum (`NPLD`). Specific proportional

loading functions may be assigned to a nodal force or displacement using the FPROp, EPROp, and/or CPROp commands. These commands are processed in a set mode in the same nodal, edge, and coordinate sequence defined above for the other nodal boundary data. For example,

```
FPROportional
  m mg pm_1 pm_2 ... pm_ndf
  n  0 pn_1 pn_2 ... pn_ndf
                        ! Termination record
```

would generate a pattern of proportional loads between nodes m and n at increments of mg. The patterns pm_i pn_i should be identical to produce predictable results. Each pm_i refers to a specific proportional loading function (see section in command language chapter). If a pm_i is zero the forced quantity will be multiplied by the *sum* of all proportional loading functions active at a particular solution time.

As a second example, the command sequence

```
EPROportional
  1 10.0  1 0 3
                        ! Termination record
```

would assign the non-zero force or displacement quantities of all nodes where x_1 is 10.0 to have their first dof multiplied by proportional loading number 1 and the third dof by proportional loading number 3. Any second dof would be multiplied by the *sum* of all defined proportional loading functions. For this to work properly it is necessary to have at least three proportional loading functions defined during the solution phase.

Proportional loading functions may also be used to specify acceleration effects on lumped masses. The MPROp command is used to specify the mass loading function numbers on nodes which have discrete masses specified by the MASS mesh command. The MPROp command is input as:

```
MPROportional
  m mg  mp_1 mp_2 ... mp_ndf
  n ng  np_1 np_2 ... np_ndf
                        ! Termination record
```

and generation can be performed in a manner similar to the FPROp command.

In each momentum equation a time-dependent loading term associated with an MPRO command will be computed as:

$$\mathbf{F}_n = \boldsymbol{g}(\boldsymbol{x}_n) - \boldsymbol{M}_{nn}\,\ddot{\boldsymbol{x}}_n$$

where n is the node number and the components of $\boldsymbol{g}$ are defined as

$$g_i(\boldsymbol{x}_n) = f_i\, prop_k(t) \quad \text{where } k = np_i(n)\ .$$

The factors f_i are specified using the GROUp global command. If the g_i are associated with inertial loading, for example from an earthquake record, the product of f_i and $prop_k$ must be density times acceleration.

5.5.7 Periodic boundary conditions

Specified displacement conditions

For some problems in stress and/or thermal analysis of solids it is desirable to specify non-zero nodal displacements that result from applying a non-zero displacement and/or thermal gradient. For such situations the displacement of the node is computed from

$$\mathbf{u}_a = \mathbf{G}\,\mathbf{x}_a$$

where $\mathbf{u}_a$ and $\mathbf{x}_a$ are the displacement and coordinate of node a; and $\mathbf{G}$ is a *specified* displacement gradient computed from

$$\mathbf{G} = \begin{bmatrix} \dfrac{\partial u_1}{\partial x_1} & \dfrac{\partial u_1}{\partial x_2} & \dfrac{\partial u_1}{\partial x_3} \\ \dfrac{\partial u_2}{\partial x_1} & \dfrac{\partial u_2}{\partial x_2} & \dfrac{\partial u_2}{\partial x_3} \\ \dfrac{\partial u_3}{\partial x_1} & \dfrac{\partial u_3}{\partial x_2} & \dfrac{\partial u_3}{\partial x_3} \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix}$$

and the thermal gradient from

$$T_a = \mathbf{G}\,\mathbf{x}_a$$

where T_a are nodal temperatures and $\mathbf{G}$ is a *specified* thermal gradient computed from

$$\mathbf{G} = \begin{bmatrix} \dfrac{\partial T}{\partial x_1} & \dfrac{\partial T}{\partial x_2} & \dfrac{\partial T}{\partial x_3} \end{bmatrix}$$

Periodic specified displacements

Another useful application is to impose *periodic* boundary conditions in which the constraint on two *periodic* nodes is given as

$$u_i^+ = u_i^- + \sum_{j=1}^{d} G_i j (x_j^+ - x_j^-) \quad \text{with} \quad x_j^+ = x_j^- \quad j \neq i$$

where d is the mesh dimension. Such periodic cases are useful in evaluating the behavior of micro-scale models under specified strain histories. Generally, the constraint can only be used for meshes in which the geometry is rectangular and the boundary nodes satisfy the above constraint. The boundary constraints for the problem can then be given by specifying *fixed displacement conditions for the corner nodes* and `ELINk` conditions for the parallel boundaries (see Sec. 11.2).

A similar behavior may be imposed for thermal problems where temperature T has the condition

$$T^+ = T^- + \sum_{j=1}^{d} G_j(x_j^+ - x_j^-) \quad \text{with} \quad x_j^+ = x_j^- \quad j \neq i$$

in which $G_j = \partial T/\partial x_j$ is a constant thermal gradient, $T^+ = T(x_i^+)$ and $T^- = T(x_i^-)$

It is also possible to solve the problem using an incremental form in which temperature, strain and deformation gradient are defined by

$$\nabla T = \frac{\partial T}{\partial \mathbf{x}} + \bar{G}$$

$$\boldsymbol{\varepsilon} = \left(\frac{\partial \mathbf{u}}{\partial \mathbf{x}}\right)^{(s)} + \bar{\mathbf{G}}^{(s)}$$

and

$$\mathbf{F} = \frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \bar{\mathbf{G}}$$

respectively. In the above the barred quantities are the specified imposed gradients.

Data input for periodic conditions

FEAP provides for two types of treatment for periodic behavior on a representative volume element (RVE): (a) an energy balance based on Hill-Mandel theory;[17,21,22] and (b) an Irving-Kirkwood treatment.[23,24] For small strain problems the data input is given as

```
PERIODIC <HILL,IRVIng>
  <MECHanical,IMEChanical> <PROP n_u>
    eps_11 eps_12 eps_13
    eps_21 eps_22 eps_23
    eps_31 eps_32 eps_33
```

which are components of the *tensor* strains, i.e. $\epsilon_{ij} = \gamma_{ij}/2$; and for finite deformation by a *periodic displacement gradient* is specified by

```
PERIODIC <HILL,IRVIng>
  <MECHanical,IMEChanical> <PROP n_u>
    G_11 G_12 G_13
    G_21 G_22 G_23
    G_31 G_32 G_33
```

and for the thermal problem by

```
PERIODIC <HILL,IRVIng>
  <THERmal,ITHEmal> <PROP n_t>
    G_1 G_2 G_3
```

For a thermo-mechanical problem both parts may be specified by

```
PERIODIC <HILL,IRVIng>
  <MECHanical,IMEChanical> <PROP n_u>
    G_11 G_12 G_13
    G_21 G_22 G_23
    G_31 G_32 G_33
  <THERmal,ITHErmal> <PROP n_t>
    G_1 G_2 G_3
```

where n_u is the displacement proportional load and n_t the one for temperature. The specification of the mechanical and thermal gradients may be given in either order In the above the `PROP` permits subjecting the boundaries to a time dependent behavior, but at the same spatial gradient distribution.

In some instances it is desirable to specify a time varying gradient applied to an RVE. In such cases the gradients may be specified in a separate data file. The specification of the `PERIodic` command for file input is given as with the record

```
PERIodic <HILL,IRVIng>
  FILE filename
  <MECHanical,IMEChanical>
  <THERmal,ITHERMal>
```

where *filename* describes where the gradients are located. For mechanical problems the displacement gradients at each time t are given by

```
t G_11 G_21 G_31 G_12 G_22 G_32 G_13 G_23 G_33
```

for a thermal problem the data has the form

```
t G_1 G_2 G_3
```

and for a thermo-mechanical problem

```
t G_11 G_21 G_31 G_12 G_22 G_32 G_13 G_23 G_33   G_1 G_2 G_3
```

See Sect. 14.1.25 for details on solving the problem.

5.6 Surface loading

FEAP uses the `CSURface` command to specify distributed traction, displacement, or flux values (e.g., for thermal problems) on portions of two or three dimensional surfaces defined by interpolation patches.

5.6.1 Two dimensional problems

The two dimensional surface command may be used for any element that has a linear (2-node), a quadratic (3-node) or a cubic (4-node) edge. For two dimensional problems the command has the structure

```
CSURface
  type, data
  LINEar
  1,X_1,Y_1,P_1
  2,X_2,Y_2,P_2
                        ! blank termination record
```

or

```
CSURface
  type, data
  QUADratic
  1,X_1,Y_1,P_1
  2,X_2,Y_2,P_2
  3,X_3,Y_3,P_3
                        ! blank termination record
```

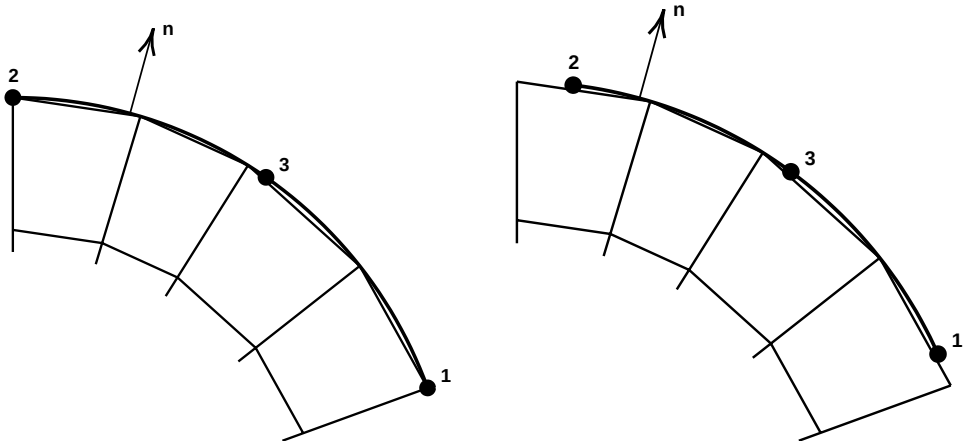
where `type` is an optional data type selected from: `AXIS`metric; `PLANe` (default); `CART`esian (default); `POLAr`; `GAP`; `MATE`; `FLUX` (e.g., thermal loading or other scalar types); `NORM`al traction; `TANG`ential traction; `DISP`lacement pattern (default is normal traction). More than one `type` record may be given. The `MATE` type may be used to restrict the surface search to elements that are associated with a specific material type. The command is given as

```
MATE data
```

where `data` is a number between 0 and `nummat`. If `data` is zero the elements for all materials are included.

If the data type is `DISPlacement` the parameter `data` specifies the coordinate direction for the specified values and the `P_i` the values to be interpolated. Similarly, for `FLUX` values the `data` specifies the degree of freedom to which the flux is applied (default is 1). Multiple records of `type` may exist before input of interpolation patches and patterns. An example of a `FLUX` input for a two-dimensional quadratic segment is:

```
CSURface
  FLUX 4
  QUADratic
     1 x_1 y_1
     2 x_2 y_2
     3 x_3 y_3
     ! Termination record
```



(a) Covering all nodes (b) Not covering nodes

Figure 5.13: Two-Dimensional Surface Loading

The parameters `LINE`ar or `QUAD`ratic define the order of the interpolation patch. The values of x_1,y_1 and x_2,y_2 define coordinate end points on the patch and, for quadratic surfaces, x_3,y_3 define the middle point coordinates for the patch. The parameters p_1, p_2, and p_3 define the values of the traction or the displacement at the corresponding coordinates on the patch.

FEAP will search for all nodes which are closer to the interpolation patch than `GAP` (default is $0.002 \times h$ where h is an estimated element size). Using the element boundary segments which have outward normals to the patch (*by right hand coordinate rule* as shown for a two-dimensional problem in Figure 5.13 – also see Fig. 5.2 for node numbering sequences) will be located and the values interpolated to nodes. For tractions the equivalent nodal loads will be computed. In two dimensions it is not necessary for the interpolation patch to exactly match the element boundary segments as shown in Fig 5.13(b). Correct nodal loading will be determined for the partial cover of the end segments for elements which have linear, quadratic and cubic isoparametric edge interpolation; however, for a Hermitian cubic beam interpolation the treatment will be treated as a linear interpolation edge.

Use of the `POLA`r option permits the coordinates x_1 and x_2 to be given as a radius and angle (in degrees) and internally converted to Cartesian form.

A common error is to have an incorrect sequence for the boundary segments so that the outward normal points in the wrong direction. When no loads are computed it is necessary to carefully check the normal direction to a patch. Also check that the value of the proportional loading factor is non-zero. If none of these errors are identified then the value of the search gap can be increased by inserting the command

```
GAP,value
```

before the interpolation patch data.

5.6.2 Three dimensional problems

For three dimensional problems the command has the structure

```
CSURface
  type, data
  SURFace
  1 X_1 Y_1 Z_1 P_1
  2 X_2 Y_2 Z_2 P_2
  3 X_3 Y_3 Z_3 P_3
  4 X_4 Y_4 Z_4 P_4
    ...
  9 X_9 Y_9 Z_9 P_9
                        ! blank termination record
```

where `type` is the data type selected from: `GAP`; `NORM`al traction, `TRACtion`, `DISP`lacement pattern or `FLUX`. No tangential option currently exists, however, the `TRACtion` type can be used to apply loading in a specified coordinate direction, where `data` is the coordinate direction for the computed traction component. Thus, if a surface exists for which all nodes have a constant coordinate value the loading may be made to be tangential to this surface using the `TRACtion` option. Also, only those element surface facets which lie on or within the interpolation patch are selected. No partial facets are permitted. The surface patch is given as a 4 to 9-node quadrilateral super-element. The first 4-nodes are the vertices with subsequent nodes the mid-side and central nodal locations. The vertex nodes are required, whereas the others are optional.

The surface option may be used for elements whose surface facets are 3 or 6-node triangle, 4, 8, 9, or 16-node quadrilateral, 4 or 10-node tetrahedron, 8, 20, 27 or 64-node hexahedron, 5-node pyramid, or 6-node wedge element types.

5.7 Load groups

The above input types for boundary and surface loads and displacements may be assigned to individual time dependent load groups using the input commands

```
LOAD <PROPld PNUM>
  ....  Force and/or Displacement Commands
LOAD  END
```

where `PROPld PNUM` indicates use of a proportional load with number `PNUM`. It is necessary to define the proportional load functions as described in Section 5.5.6 and, thus, the value of `PNUM` must be in the range *0* to the maximum number defined. If the option `PROPld PNUM` is omitted and the load group is specified as

```
LOAD
  ....  Force and/or Displacement Commands
LOAD  END
```

the sum of all defined proportional loads is used as the time function multiplying the forces and displacements defined in the load group (this is identical to using `LOAD PROP 0` to start a load group).

Note that *only* force or displacement commands should be placed within a load group. All boundary condition sets should be placed outside all load sets.

Multiple load sets may be input without the `LOAD END` statement if no inputs other than those belonging to the individual load sets exists. Thus, multiple sets may be input as

```
LOAD PROP 1
  ....  Force and/or Displacement Commands for first set
LOAD PROP 2
  ....  Force and/or Displacement Commands for second set
LOAD   ...
  ....  etc.
LOAD END
```

If the final `LOAD END` is omitted before the `END` mesh command, any input forces or displacements defined after the final `LOAD` set will be assigned to that set.

Using this option, it is possible to have individual nodal values of force or displacement vary by multiple time functions. As an example consider the input command set:

```
LOAD PROP 1
  FORCE
    10  0  0.0 -10.0

LOAD PROP 2
  FORCE
    10  0  2.0   5.0

LOAD END
```

will produce a nodal loading on node 10 given by:

$$\mathbf{F}(t) = \begin{Bmatrix} 0.0 \\ -10.0 \end{Bmatrix} p_1(t) + \begin{Bmatrix} 2.0 \\ 5.0 \end{Bmatrix} p_2(t)$$

where $p_1(t)$ and $p_2(t)$ denote time functions defined by the proportional loads 1 and 2, respectively.

5.8 Finite rotation boundary condition

In some instances it is necessary to rotate a boundary node through a finite angle. For example, subjecting a tubular test specimen to a twist requires one end to rotate with respect to the other. For finite angles of twist the displacements must move on circular arcs. This may be accomplished within a `LOAD` group (see Sect. 5.7) using the mesh `SPIN` command. The command is given using the following set of statements

```
LOAD PROPld ps

  SPIN
    CENTer   xc yc zc ! Point about which nodes rotate
    NORMal   nx ny nz ! Axis of rotation
    RATE     w        ! Rotation angle/time in degrees
    VELOcity vx vy vz ! Translateion velocity of center
    PROPld   pv       ! Velocity proportional load number
                      ! Termination record
 <DISP,EDIS,CDIS>     ! List of nodes to spin>
    ...
LOAD END
```

The proportional load `ps` of the load group controls the spin rate motion. The `VELOcity` and `PROPld` commands are not required if the center point does not translate. These options are, however, required if the `SPIN` command is used to model a rolling wheel.

In addition to the `SPIN` command, the nodes to be rotated must be defined using any of the displacement options (e.g., `DISP, EDISp, CDISp`). To force *FEAP* to place the node in the load group a non-zero displacement must be placed on at least one component of each defined displacement. For example using the edge option the commands may be given as

```
EDISplacement
  3  z  1.0
                    ! Termination record
```

where z is the value of the 3-coordinate to which the spin is to be given and the unit displacement value is to force all nodes with this coordinate to be in the load set. `N.B. The actual value of the displacement in the command is never used. The actual displacement is computed from the SPIN values.`

5.9 Filter command

When using commands based on coordinate or edge values there are often situations where the coordinate value also exists in parts of the domain where the input is to be

ignored. *FEAP* uses a `FILTer` command to associate the domain part where the value is to be considered. The domain part must be distinguished by either a material set or a region number. Material set numbers are the value set during input of `ELEMent`, `BLOCk`, or `BLENd` command sets.

The filtering applies to subsequent commands that are based on edge or coordinate values (e.g., `EFORce`, etc.).

To associate filtering based on a material set. the command is given by

```
FILTer MATErial 2

EBOUndary
  2 40.0  1 0

FILTer OFF
```

This set would look at all elements belonging to material set 2 and apply a boundary condition `(0, 1)` to any node in the set that had a coordinate `y = 40.0`. To associate filtering based on `REGIon` definitions the filtering command set is initiated by

```
FILTer REGIon r_num
```

where `r_num` is the number of a `REGion`.

5.10 Regions and element groups

The elements in *FEAP* may be assigned to different groups using the `REGIon` command. The command is given as

```
REGIon,number
```

where `number` is an integer constant of parameter defining the group number for the elements. Any elements which are input after a region command is given belong to the given group number. By default all elements are assigned to region zero.

The use of regions facilitates two aspects in *FEAP*. The first is for use in merging groups of elements (mesh manipulation command) whose nodes should be common but have different numbers (e.g., those defined using `BLOCk` commands). An illustration of this option is used in Example 4 in the *Example Manual*. The second use is to activate or deactivate elements to represent excavation or construction sequences. This option uses the `ACTIvate` or `DEACtivate` command language instructions (see Appendix D). Elements in Region zero may not be deactivated.

5.11 Global data

FEAP uses the `GLOBal` command to specify data which is common to *all elements*. In general a `GLOBal` command must appear before any data to which it is applied (*e.g*, before `MATErial` sets). The set of options for the `GLOBal` command are given in Table 5.8.

Command	Type	Parameters
EQUAtion		g_e, g_{part}
PLANe	STRAin	
PLANe	STREss	
AXISymmetric		
AXISymmetric	TORSion	
SMALl	deformation	
FINIte	deformation	
REFErence	AXIAL	
REFErence	NODE	x, y, z
REFErence	POLAr	
REFErence	VECTor	v_x, v_y, v_z
GROUp		$g_i \quad i = 1, \text{ndf}$
RAYLeigh		a_0, a_1
OMEGa		ω-radians/time
OMEGa	CYCLe	ω-cycles/time
OMEGa	NODE or COOR	x, y, z
OMEGa	VECTor	n_x, n_y, n_z
TEMPerature	DOF	`tdof`
QUADrature	NODE or NODAl	Use nodal quadrature
QUADrature	GAUSs	Use Gaussian quadrature
AUGMent	ON or OFF	Turn on/off augmented Lagrangian for solids
SCALe	COOR	`s_x` Factor for coordinates
SCALe	TIME	`s_t` Factor for time
SCALe	MASS	`s_m` Factor for mass
SCALe	DISP	`s_d(1:ndf)` Factor of each displacement dof
SCALe	ELEM	`n, s_e(1:n)` Factor of element parameters

Table 5.8: Global command options.

Global equations

In addition to nodal and element variables, *FEAP* can accommodate a small number of global equations. Global equations are those that can influence all or a very large

number, of nodes in the mesh. Global equations may only be applied to one partition in the problem. The command is given in the mesh data as

```
GLOBal
  EQUAtion,,no_geq,part_geq
                        ! blank termination record
```

where `no_geq` is the number of global equations and `part_geq` the partition number to which they apply. By default they apply to the default partition.

Two-dimensional geometry type

In two-dimensional applications it is possible to specify that all elements should select a plane stress, a plane strain, or an axisymmetric representation. If the problem is to be solved as a plane stress problem, the global data is specified as:

```
GLOBal
  PLANe STREss
                        ! blank termination record
```

Thus, by changing the record describing the type of two dimensional analysis the system elements will all use the same type of behavior. The other options are `PLANe STRAin`, `AXISymmetric` and `AXISymmetric TORSion`. All the two-dimensional forms for solid elements (except the torsion case) have 2-degrees of freedom at each node (u_x, u_y for plane cases and u_r, u_z for axisymmetry). The torsion case has a third degree of freedom. In the small deformation problem the degree of freedom is a displacement component u_θ, whereas, in the finite deformation case it is an angle ϕ that each node r, z point rotates in the circumferential direction. **N.B. The GLOBal command must appear before the MATErial data to which it applies.**

If it is desired, for some modeling reason, to have one type of element use a different formulation the global data can be ignored by specifying the particular type of analysis needed as part of the `MATErial` property data.

Finite or small deformation analysis

A problem in solid mechanics may be designated as *small* or *finite* deformation using global commands

```
GLOBal
  SMALl
                        ! blank termination record
```

or

```
GLOBal
  FINIte
                        ! blank termination record
```

respectively.

Proportional body loading

Problems for which *ground accelerations* are specified as proportional load tables may be solved using a specified pattern of amplification factors, f_i, for each degree of freedom. These factors are applied to a discrete mass input using the `MASS` command using the command

```
GLOBal
  GROUp factors f_1 f_2 ... f_ndf
                        ! blank termination record
```

The loading is applied as

$$\mathbf{F}_n = \boldsymbol{g}(\boldsymbol{x}_n) - \boldsymbol{M}_{nn}\,\ddot{\boldsymbol{x}}_n$$

where n is the node number and the components of $\boldsymbol{g}$ are defined as

$$g_i(\boldsymbol{x}_n) = f_i\, prop_k(t) \quad \text{where } k = np_i(n)\ .$$

The factors f_i are specified using the `GROUp` global command. If the g_i are associated with an earthquake record expressed as ground acceleration the f_i must include density scaling so that the product of f_i and $prop_k$ yields density times acceleration.

Rayleigh damping parameters

For small deformation transient analysis damping effects may be introduced for use by the solid and structural elements as Rayleigh damping. Each material may have different Rayleigh damping parameters (see 7.15). Alternatively, the Rayleigh parameters may be assigned as global values using the commands

```
GLOBal
  RAYLeigh damping a0 a1
                        ! blank termination record
```

where the parameters `a0` and `a1` are defined in Section 7.15. The global damping value may also be used for modal solutions as described in Section 14.5.1.

Reference vector for 3-d frames

The cross sectional properties for a frame element are described in terms of an orthonormal triad $\mathbf{t}_1$, $\mathbf{t}_2$ and $\mathbf{t}_3$ as shown in Fig. 5.14. The vector $\mathbf{t}_1$ is aligned with the beam axis and $\mathbf{t}_2$, $\mathbf{t}_3$ describe the cross-section plane. The orientation of the triad is described from a *reference vector* that may be given either as part of a `GLOBal` data set or as part of the `MATErial` data set (viz. Sect. 6.5).

For frame elements in which the cross section is a square or circular shape the orientation is not critical and the reference vector may be specified to be in the *axial* direction of each element. In such case a unique set of vectors will be constructed automatically. The data is given as

```
GLOBal
  REFErence AXIAl
                    ! blank termination record
```

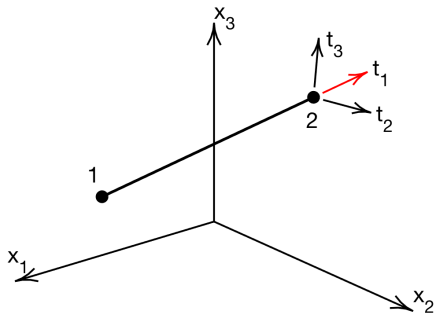
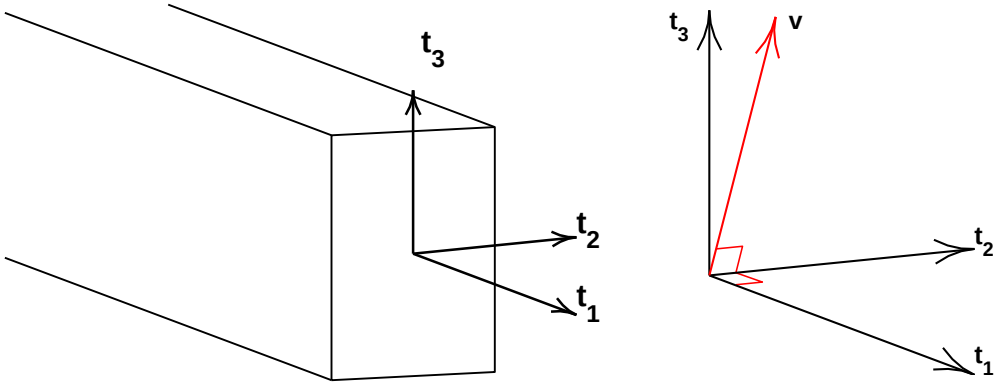


Figure 5.14: Axes for a 3-d frame elements.



(a) Cross-section vectors. (b) Reference vector.

Figure 5.15: Reference vectors for a 3-d frame cross section.

In other cases it is necessary to define the frame axes as shown in Fig. 5.15(a). This may be performed by specifying a vector which is *not parallel to the axis for any member in the set*. Three options exist to specify the orientation.

- The `VECTor` option is used to describe the orientation for the axis of each element. If the vector is given as $\mathbf{v}$ [viz. Fig. 5.15(b)] a set of axis are constructed using a unit vector in the axial direction $\mathbf{t}_1$ as

$$\mathbf{T}_2 = \mathbf{v} \times \mathbf{t}_1 \quad ; \quad \mathbf{t}_2 = \frac{\mathbf{T}_2}{|\mathbf{T}_2|}$$

  and

$$\mathbf{t}_3 = \mathbf{t}_1 \times \mathbf{t}_2 \quad .$$

  where $|\mathbf{T}| = (\mathbf{T} \cdot \mathbf{T})^{1/2}$. The input is given by

```
GLOBal
  REFErence VECTor v_1 v_2 v_3
                        ! blank termination record
```

  where v_i are the components of the $\mathbf{v}$ vector.

- If the input is given as

```
GLOBal
  REFErence NODE x y z
                        ! blank termination record
```

  then the same computation is used with $\mathbf{v} = \mathbf{x} - \mathbf{x}_1$ where $\mathbf{x}_1$ is the nodal position of the first node of the element and `x, y, z` the components of the $\mathbf{x}$ vector..

- If the input is given as

```
GLOBal
  REFErence POLAr
                        ! blank termination record
```

  the vector $\mathbf{v} = (v_x,\ v_y,\ 0)$ where $v_x = \frac{1}{2}(x_1 + x_2)$ and $v_y = \frac{1}{2}(y_1 + y_2)$ where x_a and y_a are positions of the 2 element nodes.

Rotational body force

Three dimensional bodies which are rotating at a constant angular velocity about a *fixed axis* may be analyzed. The body force is computed from

$$\mathbf{b} = -\,\rho\,\widehat{\boldsymbol{\omega}}\,\widehat{\boldsymbol{\omega}}\,(\mathbf{x} - \mathbf{r})$$

where for finite deformation applications ρ is the mass density in the *deformed* configuration, $\boldsymbol{\omega}$ is the *constant* angular velocity vector in radians/time unit (which has

the skew symmetric matrix form $\widehat{\boldsymbol{\omega}}$), and $(\mathbf{x} - \mathbf{r})$ is the deformed position relative to a fixed point $\mathbf{r}$. In a finite element setting the nodal force is most easily computed by transforming to the reference body as

$$\mathbf{F}_A = -\int_{\Omega_0} \rho_0 \, N_A \, \widehat{\boldsymbol{\omega}} \, \widehat{\boldsymbol{\omega}} \, N_B \, \mathrm{d}V \, \mathbf{x}_A$$

where Ω_0 is the reference configuration volume, ρ_0 the mass density in the reference configuration and N_A, N_B are shape functions for the element with $\mathbf{x}_B$ the nodal coordinates in the deformed position. A constant angular rotation may be expressed as

$$\boldsymbol{\omega} = \omega \, \mathbf{n}$$

where ω is the rotational velocity and $\mathbf{n}$ a unit vector describing the axis. To carry out the computations, it is necessary to describe the position $\mathbf{r}$ and direction of a fixed axis $\mathbf{n}$ about which the rotation takes place. This is given by specifying a point (`NODE` or `COORdinate`) and a vector `VECTor` along the axis. The input vector is converted to a unit vector. Finally, the angular velocity may be given in either *radians/time* or *cycles/time*. A typical data set is given by the commands

```
GLOBal
  OMEGA radian omega
  OMEGA COORd  x   y   z
  OMEGA VECTor n_x n_y n_z
                    ! blank termination record
```

Thermal degree of freedom

The variable to be used for temperature in a coupled thermo-mechanical analysis be defined globally as

```
GLOBal
  THERmal dof n_tdof
                    ! blank termination record
```

Scaling factors for equations

Some problems require scaling the governing equations into a non-dimensional form. *FEAP* provides global commands to scale coordinates, time, mass, each degree-of-freedom parameter and other element parameters by a factor.[3]

The scaling factors for coordinates, time and mass are given by the command set

[3] Factors are passed to elements using by including `pglob1.h`.

```
GLOBal
  SCALe COOR s_x
  SCALe TIME s_t
  SCALe MASS s_m
                        ! blank termination record
```

The scaling factors for each degree-of-freedom are given by the command set

```
GLOBal
  SCALE DISP s_d(1:ndf)
                        ! blank termination record
```

where `ndf` is the number of degrees-of-freedom specified on the control record and `s_d` is the scaling factor for each degree-of-freedom. Finally, if additional element parameters are used the parameters are specified by

```
GLOBal
  SCALe ELEMent n s_e(1:n)
                        ! blank termination record
```

where `s_e` is the scaling factor for each component.

N.B: In the current release none of the standard elements use scaling factors.

Chapter 6

Element library

FEAP contains a library of standard elements and material models which may be employed to solve a wide range of problems in heat transfer, solid mechanics and structural mechanics analyses. In addition, users may program and add new elements to the program. In this chapter we discuss the general features for each element type included in *FEAP* along with some specific data commands which are used to input material parameters. In Chapter 7 we present in more detail features for each material type. Additional descriptions on formulation details for elements are included in the *FEAP* Theory Manual.[25]

The type of element to be employed in an analysis is specified as part of each `MATErial` data set. The first record of each set also contains the material property number. Each material property number is a unique integer ranging from one (1) to the maximum number of material sets in the problem (the maximum number is generally set automatically by `FEAP` but also may be specified on the control data record, see Sect. 5.1). The second record of a material set data defines the *type* of element to be used for the set. The actual element routine to be selected will also depend on the spatial dimension of the problem being solved (i.e., a 1, 2, or 3-dimensional geometry for the mesh). The library of standard elements includes the following types:

1. THERmal - A thermal element is used to compute temperatures in the mesh. The degree of freedom for each *node* is temperature, T, and, by default, is placed in the first position in the unknowns (i.e., first degree of freedom). This class of elements solves the Fourier heat conduction equation. The degree of freedom at each node is temperature, T.

2. CONVection - A thermal convection element is used to impose linear and non-linear surface flux conditions. The degree of freedom for each *node* is temperature, T, and, by default, is placed in the first position in the unknowns (i.e., first degree

of freedom). This class of element is used with the Fourier heat conduction element for surface boundary conditions.

3. SOLId - The solid element is used to solve continuum problems with either small or large deformations. Options exist to use finite elements based on displacement, mixed, and enhanced strain formulations. The material behavior for solid elements may be modeled by elastic, viscoelastic, elasto-plastic, or user developed constitutive equations. The degrees of freedom at each node are displacements, u_i.

4. FRAMe - The frame element is used to model structural members which include axial, bending, and (optionally) shearing deformations only. The model is formulated in terms of force resultants which are computed by integration of stress components over the cross-sectional area of the member. Each element has 2-nodes and may be used in a two or three dimensional problem. The degrees of freedom on each node are: Displacements, u_i, in the coordinate directions and; A rotation, θ_z, about the z-axis for two dimensions and rotations, θ_i, about all axes for three dimensions. The degrees of freedom are ordered as: 2-D u_x, u_y, θ_z; 3-D $u_x, u_y, u_z, \theta_x, \theta_y, \theta_z$;

5. TRUSs - The truss element is used to model structural members which include axial deformations and forces only. The axial force resultant is computed by integration of the axial stress component over the cross-sectional area of the member. Each element has 2-nodes and degrees of freedom at each node are displacements, u_i, in each coordinate direction; thus, the number is the same as the spatial dimension of the problem. The degrees of freedom are ordered as: u_x, u_y, u_z

6. PLATe - The plate element is used to model structural behavior of planar bodies which have one dimension small compared to the two other dimensions. The element may be used for small deformation analyses only and includes bending and transverse shearing deformations. Provisions are also included to permit modeling of plates for which the transverse shearing deformations are ignored. The model is formulated in terms of force resultants which are computed by integration of stress components over the thickness of the plate. Each element may be a triangle with 3-nodes or a quadrilateral with 4-nodes and is used in a two dimensional problem. The degrees of freedom at each node are: The transverse displacement, $u_3 = w$, and rotations θ_x and θ_y about the coordinate axes. The degrees of freedom are ordered as: w, θ_x, θ_y;

7. SHELl - The shell element is used to model structural behavior of curved bodies which have one dimension small (a thickness normal to the remaining surface coordinates) compared to the other dimensions of the surface. The shell for small deformations includes bending and in-plane deformations only (no transverse

shearing strains). The model is formulated in terms of force resultants which are computed by integration of stress components over the cross-sectional thickness of the shell. Each element is a quadrilateral with 4-nodes and may only be used in a three dimensional problem. The degrees of freedom on each node are: Displacements, u_i, and rotations, θ_i, about the coordinate axes. The degrees of freedom are ordered as: $u_x, u_y, u_z, \theta_x, \theta_y, \theta_z$ (6-dof); The large displacement shell element includes in-plane, bending, and shearing deformations in a 5 degree-of-freedom form. This element is formulated in an energy-momentum conserving form and may not converge quadratically in general applications.

8. MEMBrane - The membrane element is used to model structural behavior of curved bodies which are thin and carry loading by in-plane loading only. These elements can be unstable unless attached to a contiguous solid or otherwise restrained. The model is formulated in terms of the in-plane force resultants and a cross-sectional thickness. Each element is a quadrilateral with 4-nodes and may be used in a three dimensional problem. The degrees of freedom at each node are: Displacements, u_i. The degrees of freedom are ordered as: u_x, u_y, u_z;

9. INTErface element - The interface element is used to define behavior on the surface between two solid, thermal, shell, membrane and/or user elements of the same kind. The degrees of freedom of the element are those of the adjoining element pairs. In general, users are required to program the physics to be modeled on the interface, however, using linking options some degrees of freedom on the two faces can share the same values (see Section 11.2 for details on linking specific degrees of freedom).

10. POINt element - The point element may consist of a mass, linear dash pot, and/or linear spring. The dash pot and spring are fixed at one end and added to the degrees-of-freedom specified by the nodes of a 2-node element. The dash pot and spring are oriented using a specified direction vector. The element may be used in any one, two or three dimensional problem. The degrees of freedom are ordered as: u_x, u_y, u_z (ndm-dof);

11. PRESsure loading - The pressure loading element is used to apply normal loading to the surface of two or three dimensional objects. The loading is associated with *elements* which define the surface on which to apply the load. Loads may be applied with respect to the normals in the reference configuration (*dead loads*) or with respect to the current configuration (*follower loads*). For follower loads an *unsymmetric tangent matrix* results and thus, only use of unsymmetric equation solvers can result in quadratic rates of convergence. Indeed, convergence may not be obtained at all when a symmetric solver is used.

12. GAP - The gap element is used to model a restraint between nodes which permits only compressive loads to be transmitted. The restraint must be in one of the

coordinate directions. This element may be used in one, two, or three dimensional problems. General problems involving intermittent contact may also be solved using the contact option (see Chapter 12).

13. ACOUstic element - The acoustic element solves the scalar wave equation in terms of pressure for 1, 2, and 3 dimensional problems. the degree of freedom is given as p. In addition the element contains interface conditions to solve the multi-physics problems with transient small deformation solid response, small surface waves, and also radiation and surface absorption conditions at boundaries.

14. LABC element - The Lysmer Absorbing Boundary Condition is used to absorb outgoing waves from a region to prevent boundary reflections. The condition is implemented for 1-, 2- and 3-dimensional regions and is most effective for elastic wave absorption.

15. ROBIn element - The Robin element is provided to support boundaries by an elastic, viscous, or combined elastic-viscous behavior. The boundary may be defined by either a line or a surface and either may be used with 2- or 3-dimensional problems. The behavior is linear and, thus, for large displacement problems provides equal resistance in the reference coordinate directions based on reference line or surface measures.

16. WINKler element - The Winkler element may be used to provide elastic support in the normal direction to boundary surfaces of 2- or 3-dimensional problems. The behavior is linear and, thus, for large displacement problems provides resistance in the normal to the reference coordinate geometry.

17. USER - Each user element must be developed and added to the program. Provisions are included which permit the addition of up to 50 additional element modules to the program. The shape of the element, the number of degrees of freedom at each node, and other parameters may be set by the user. See the *FEAP* Programmers Manual[12] for information on adding a user element.

6.1 Material sets

The association of elements to materials is controlled by the *material set* records. The first two records of the `MATE`rial set must be:

```
MATErial ma <text label>
  type unum mset (doflist(i),i=1,ndf)
```

where

- `ma` is the material set number,
- `text label` is an optional text descriptor added to the output,
- `type` is one of the above element types (e.g., solid, truss, user, etc.),
  - for `user` type, `unum` is the user element number,
- `mset` is the material set number given for each element (if omitted it is set to the material number - this option permits two material types to access the same element connection list),
- `doflist` is the list of global degree-of-freedoms to assign the internal element order (by default this is the order 1,2,3,...,ndf). For example, in a 2-d thermo-mechanical problem a thermal element which normally places the temperature at the first degree of freedom can have it moved to the third using

```
MATErial 1
  SOLId
    ....
MATErial 2
  THERmal,,1,3
    ....
```

  where the `mset` is the element set with material 1 and `unum` is not needed.

For the standard elements contained in *FEAP* one needs only the `type` parameter unless degrees-of-freedom are to be relocated (e.g., for thermal analysis as noted above).

Each element requires additional input data as described below.

6.2 Thermal elements

The thermal element solves the Fourier heat conduction equation in one, two, or three dimensional domains. The equation is described by

$$-\boldsymbol{\nabla}^T \mathbf{q} + Q = \rho\, C\, \frac{\partial T}{\partial t}$$
$$\mathbf{q} = -\mathbf{K}\, \boldsymbol{\nabla} T$$

where $\mathbf{q}$ is the thermal flux, $\mathbf{K}$ are (constant) thermal conductivity values, $\boldsymbol{\nabla}$ is the gradient operator, Q is a heat source added per volume, ρ is mass density and C is specific heat. The degree of freedom for each node α is a temperature, $\tilde{T}^{\alpha}$, and, by default, is assigned to the first global degree of freedom. The provided material

behavior for thermal analysi is a *linear* Fourier model as given above. The thermal flux relation may also be replace by a user model.

For one-dimensional problems each element may be a 2 node, a 3 node or a 4 node line element (see Section 3.1). Two dimensional problems are modeled using a surface element and each element may be a 3 or 6 node triangle or a 4, 8, 9 or 16 node quadrilateral (see Section 3.2). Two dimensional analyses may be performed for plain or axisymmetric geometry (for a one radian segment in the circumferential θ direction). Three dimensional problems are modeled using volume elements and each element is a tetrahedron with 4 or 10 nodes or a hexahedron (brick) with 8, 27 or 64 nodes (see Section 3.3).

The *thermal elements* are all based on a standard displacement (Galerkin) formulation. The material behavior for a Fourier model may be specified by either an isotropic or an orthotropic model. For orthotropy the thermal conductivity has the form

$$\mathbf{K} = \begin{bmatrix} K_1 & 0 & 0 \\ 0 & K_2 & 0 \\ 0 & 0 & K_3 \end{bmatrix}$$

where K_i are the principal conductivities. For isotropy $K_1 = K_2 = K_3 = K$.

The thermal element is included using the commands:

```
MATErial 1
  THERmal
    FOURier isotropic K   C
    DENSity mass      rho
    BODY    HEAT      Q
                            ! blank termination record
```

for isotropic behavior or for orthotropic materials

```
MATErial 1
  THERmal
    FOURier ORTHotropic K_1 K_2 K_3 C
    DENSity mass      rho
    BODY    HEAT      Q
                            ! blank termination record
```

Input of the material set is terminated with a blank record.

For an axisymmetric analysis in a one or two dimensional domain, it is necessary to add the command `AXISymmetric` to the material data, thus for an isotropic material the data is given as

```
MATErial 1
  THERmal
```

```
FOURier isotropic K   c
DENSity mass      rho
BODY    HEAT      Q
AXISymmetric
                          ! blank termination record
```

Recall also that all axisymmetric formulations, including the thermal problem, are developed on a one radian sector in the circumferential θ direction; hence the factor $2\,\pi$ is not included in any applied flux terms, etc.

Data in the set is not dependent on order except for the `MATE` and `THERmal` commands which *must* be the first and second data records, respectively. Only data needed for the analysis type to be performed must be included. Thus, if an analysis is steady state (static) the `DENSity` record may be omitted. Similarly if $Q = 0$ the `BODY` record may be omitted.

FEAP will pick the quadrature order depending of the element type and order, however, users may specify the quadrature order as data using the command

```
QUADrature order q_1
```

where `q_1` is the order used to compute arrays and for element outputs.

Similarly, the mass matrix type is *consistent* by default. The command

```
MASS CONSistent
```

explicitly gives this option or alternatively a command

```
MASS DIAGonal
```

or

```
MASS LUMP
```

may be used to specify a diagonal (lumped) mass. Indeed, an interpolation between a consistent and a diagonal mass may be specified using the command

```
MASS type a
```

in which a mass is computed using

$$\mathbf{M} = (1 - a)\,\mathbf{M}_{diag} + a\,\mathbf{M}_{cons}$$

In some cases a better or smoother answer may be obtained using an interpolated form (e.g., see pp. 476 in Zienkiewicz & Taylor[19] and in Taylor & Iding[26]). Indeed, when small time steps are used along with a consistent approximation oscillations result and may be better controlled using small or zero values of `a`.

6.3 Convection thermal elements

The convection element is used with thermal elements to impose boundary flux defined by:

$$q_n = q_0 * p_1(t) + H\left[T^n - (T_0 * p_2(t))^n\right]$$

where q_0 is a constant prescribed flux; $p_i(t)$ are a proportional loading values; H is a surface parameter; n is an integer exponent for convection/radiation and T_0 is a free-stream temperature. Setting $n = 1$ gives a *linear* convection condition whereas setting $n = 4$ gives the Stefan-Boltzman radiation approximation rendering the analysis *non-linear*.

The input data for the convection boundary condition is given by:

```
MATErial
  CONVection
    SURFace  H T_0 p_2
    FLUX     q_0 p_1
    EXPOnent n            ! Default is 1
    PLANe or AXISymmetrix ! Default is PLANe
```

where `H` $= H$; `T_0` $= T_0$; `q_0` $= q_0$ and `p_1` and `p_2` are proportional loading numbers. The `PLANe` or `AXISymmetric` options are used to set the geometry type of the boundary. If the proportional loading number is zero, $p_i(t) = 1$.

The form for a plane, linear convection condition is given as

```
MATErial
  CONVection
    SURFace  H T_0 p_2
```

and for the Stefan-Boltzman radiation condition by

```
MATErial
  CONVection
    SURFace  H T_0 p_2
    EXPOnent 4
```

N.B. The global setting of `AXISymmetric` is not recognized by the convection boundary element, the geometry type must be set by the element.

Convection boundary elements may be used with 1-, 2- or 3-dimensional elements. The element type has a spatial dimension one less than that for the thermal element; thus, it is a point, 1-2 line, or 2-d surface. For 3-dimensional problems the option for `PLANE/AXISymmtric` is not specified.

6.4 Solid elements

Solid elements are used to analyze problems in solid mechanics which are modeled as plane stress, plane strain, axisymmetric (with or without torsion) deformations in one or two dimensions or as fully three dimensional deformations. This analysis type is designated within a *material set* by the type `SOLId`. For one dimensional problems each element is a *line* with 2 to 4 nodes. For two dimensional problems each element may be a triangle with 3, 6, or 10 nodes or a quadrilateral with 4, 8, 9, or 16 nodes (enhanced formulation permits only 4-node quadrilaterals). In three dimensional problems Each element may be a 4 or 10 node tetrahedron, a 5 node pyramid, a 6-node wedge or an 8, 20, 27 or 64 node hexahedron (brick) with (enhanced element may only be an 8 node hexahedron).

The degrees of freedom on each node are displacements, u_i, in the coordinate directions. The degrees of freedom are ordered as:

- 1-D Plane problem, u_x, coordinate is x;
- 1-D Axisymmetric problem without torsion, u_r, coordinate is r;
- 1-D Spherical coordinate problem, u_r, coordinate is r;
- 2-D Plane problems, u_x, u_y, coordinates are x, y;
- 2-D Axisymmetric problems without torsion, u_r, u_z, coordinates are r, z;
- 2-D Axisymmetric problems with torsion, u_r, u_z, u_θ, coordinates are r, z;
- 3-D problems, displacements u_x, u_y, u_z, coordinates are x, y, z.

All axisymmetric formulations included in *FEAP* are developed on a one radian sector in the circumferential θ direction; hence the factor $2\,\pi$ is not included in any applied loads, etc.

6.4.1 Small deformation analysis

By default all analyses for solid elements are performed using a small deformation assumption where strains $\boldsymbol{\epsilon}$ in Cartesian coordinates are expressed in terms of displacements as

$$\boldsymbol{\epsilon} = \frac{1}{2}\left[\boldsymbol{\nabla} u + (\boldsymbol{\nabla} u)^T\right]$$

The strains are ordered as;

$$\boldsymbol{\epsilon} = \left[\ \epsilon_{xx}\,,\quad \epsilon_{yy}\,,\quad \epsilon_{zz}\,,\quad \gamma_{xy}\,,\quad \gamma_{yz}\,,\quad \gamma_{zx}\ \right]^T$$

where engineering shear strains defined by $\gamma_{ij} = 2\,\epsilon_{ij}$ are used. For axisymmetric coordinates the strains are ordered

$$\boldsymbol{\epsilon} = \left[\begin{array}{cccccc} \epsilon_{rr}\,, & \epsilon_{zz}\,, & \epsilon_{\theta\theta}\,, & \gamma_{rz}\,, & \gamma_{z\theta}\,, & \gamma_{\theta r} \end{array} \right]^T$$

with the strain displacement relations given by

$$\boldsymbol{\epsilon} = \left[\begin{array}{c} \partial u_r/\partial r \\ \partial u_z/\partial z \\ u_r/r \\ \partial u_r/\partial z + \partial u_z/\partial r \\ \partial u_\theta/\partial z \\ \partial u_\theta/\partial r - u_\theta/r \end{array} \right]$$

For problems without torsion the last two components are ignored and only two degrees of freedom are used at each node.

For small deformation analyses the material behavior may be linear elastic, linear viscoelastic, isotropic/anisotropic elasto-plastic or generalized isotropic plasticity models. Additional user material models may be added as described in the *FEAP* Programmer Manual.[12] Finally, *FEAP* has an option to define material behavior using an FE^2 formulation with representative volume elements (RVE).

Linear elastic models

The constitutive behavior for linear elastic behavior is given by Hooke's law expressed as

$$\boldsymbol{\sigma} = \mathbf{D}\left[\boldsymbol{\epsilon} - \boldsymbol{\alpha}(T - T_0)\right]$$

where $\boldsymbol{\sigma}$ are stresses, $\mathbf{D}$ are elastic moduli, α is coefficient of linear thermal expansion, and T_0 is a stress free temperature. The temperatures T may be either specified at nodes (see Appendix A, command `TEMP`) or computed by a thermal analysis.

For an isotropic material in which the independent material properties are given as E, ν, α and ρ the input data is given by the command set

```
MATErial 1
  SOLId
    ELAStic ISOTropic E  nu
    DENSity mass      rho
    THERmal ISOTropic alpha T-0
                           ! blank termination record
```

The property `ELAStic` is required for all types of `SOLId` elements. After the `SOLId` specification, commands may be given in any order. Input of the material set is terminated with a blank record.

In addition to an isotropic model *FEAP* permits analyses for transversely isotropic, orthotropic and general anisotropic models. For a transversely isotropic material the elastic properties are input as E_1, E_2, ν_{12}, ν_{31} and $G_{23} = G_{31}$ as

```
ELAStic TRANsverse E_1 E_2 nu_12 nu_31 G_23
```

with coordinate direction 2 (y) is normal to the plane of isotropy. The direction of the normal may be changed only by a rotation about the 3 (z) axis using the command

```
ANGLe   axis        psi
```

where ψ describes the angle in degrees.

For an orthotropic material the properties are given as:

```
ELAStic ORTHotropic E_1 E_2 E_3 nu_12 nu_23 nu_31 G_12 G_23 G_31
```

and by default are described with respect to the 1 (x), 2 (y), and 3 (z) axes. Again a rotation may be specified about the 3 axis using the `ANGLe` command.

For general anisotropic behavior the properties may be input as *moduli* or as *compliances*. The input data is specified as:

```
ELAStic COMP (or MODU)  n
 C_11 C_12 ... C_1n
 C_21 C_22 ... C_2n
 ...
 C_n1 C_n2 ... C_nn
```

Although the full array must be input *FEAP* assumes symmetry and retains only a triangular part. Users should be especially careful that the input compliances or moduli define a *positive definite* matrix which has a unique inverse. For compliance inputs, *FEAP* performs the inverse and saves the properties as moduli.

Currently, the most general input for the thermal expansion array is for orthotropic behavior where input is given as

```
THERmal ORTHotropic alpha_1 alpha_2 alpha_3 T_0
```

for the principal directions. These are transformed also when a nonzero angle is given by the `ANGLe` commend.

The default element formulation type is a displacement model. *FEAP* also contains a mixed model and an enhanced strain model as options (see theory in Zienkiewicz, Taylor & Zhu[16]). To designate another formulation type the commands are given as:

```
MATErial ma
  SOLId
    ELAStic ....
   <MIXEd, ENHAnced, DISPlacement, INCOmpressible>
     ....
```

When an effective Poisson ratio is high (e.g., greater than 0.4) *shear locking* may be avoided using either the mixed formulation or an enhanced strain formulation (see Zienkiewicz, Taylor & Zhu[16]).

There are six solid element types: (1) Displacement model; (2) Mixed $\mathbf{u} - p - \theta$ model; (3) Enhanced strain model; (4) Mixed-Enhanced model; (5) Energy-Momentum conserving model; and (6) Incompressible, isotropic model. Types currently available in each analysis form are discussed next.

6.4.2 One dimensional formulations

The input options for one dimensional analysis are:

- `PLANe STRAin, PLANe STREss, AXISymmetric` or `SPHErical`;
- `SMALl` deformation;
  - `DISPlacement`, `MIXEd` or `ENHAnced`;
- `FINIte` deformation;
  - `DISPlacement`, `MIXEd`; or `CONServing`;
- `MASS LUMPed`, `MASS CONSistent` or `MASS OFF`;

Type: `SOLId`	Command Name	Option Name	Dimen. `NDM`	Node/Elm. `NEL`
Displacement	DISP		1	2-3
Mixed	MIXE		1	2-3
Enhanced	ENHA		1	2
Displacement	DISP		2	3-16
Mixed	MIXE		2	3-16
Enhanced	ENHA		2	4
Mixed-Enhanced	MIXE	ENHA	2	4
Incompressible	INCO		2	4-16
Displacement	DISP		3	4-64
Mixed	MIXE		3	4-64
Enhanced	ENHA		3	8
Incompressible	INCO		3	8-64

Table 6.1: Options for Small Deformation Solid Elements

Type: SOLId	Command Name	Dimen. NDM	Node/Elm. NEL
Displacement	DISP	1	2-3
Mixed	MIXE	1	2-3
Energy-Momentum	CONS	1	2
Displacement	DISP	2	3-16
Mixed	MIXE	2	3-16
Enhanced	ENHA	2	4
Incompressible	INCO	2	4-16
Energy-Momentum	CONS	2	4
Displacement	DISP	3	4-64
Mixed	MIXE	3	4-64
Enhanced	ENHA	3	8
Incompressible	INCO	2	8-64
Energy-Momentum	CONS	3	8

Table 6.2: Options for Finite Deformation Solid Elements

In one dimensional applications the displacement and the mixed formulation may be described by line elements with two (2), three (3) or four (4) nodes. The elements are shown in Fig. 3.1. The enhanced strain element is limited to two (2) node lines only. Recall also that all axisymmetric formulations included in *FEAP* are developed on a one radian sector in the circumferential θ direction; hence the factor $2\,\pi$ is not included in any applied loads, etc.

6.4.3 Two dimensional formulations

The input options for two dimensional analysis are:

- `PLANe STRAin, PLANe STREss, AXISymmetric` or `AXISymmetric TORSion`;
- `SMALl` deformation;
  - `DISPlacement`, `MIXEd`; `ENHAnced`; `MIXEd ENHAnced`; `UNIForm` or `INCOmpressible`;
- `FINIte` deformation;
  - `DISPlacement`, `MIXEd`; `ENHAnced`; `UNIForm`; `CONServing` or `INCOmpressible`;
- `MASS LUMPed`, `MASS CONSistent` or `MASS OFF`;

In two dimensional applications the displacement and the mixed formulation may be described by elements with between four (4) and sixteen (16) nodes. The elements described by this range are shown in Figs. 3.2 and 3.3. The enhanced strain and mixed-enhanced elements are limited to four (4) node quadrilaterals only. A three node triangular element may be formed for the displacement model by repeating the number of any node or by specifying only three nodes on an element. Recall also that all axisymmetric formulations included in *FEAP* are developed on a one radian sector in the circumferential θ direction; hence the factor $2\,\pi$ is not included in any applied loads, etc.

6.4.4 Three dimensional formulations

The input options for three dimensional analysis are:

- `SMALl` deformation;
  - `DISPlacement`, `MIXEd`; `ENHAnced`; `UNIForm` or `INCOmpressible`;
- `FINIte` deformation;
  - `DISPlacement`, `MIXEd`; `ENHAnced`; `UNIForm`; `CONServing` or `INCOmpressible`;
- `MASS LUMPed`, `MASS CONSistent` or `MASS OFF`;

In three dimensional applications the displacement and the mixed formulation may be described by elements with between eight (8) and twenty seven (64) nodes. The elements described by this range are shown in Fig. 3.4. The enhanced strain element is limited to eight (8) node hexahedral bricks only. A four node tetrahedral element may be formed for the displacement model by specifying only four nodes on an element. The displacement model may also describe a wedge or pyramid shape by coalescing appropriate nodes of a hexahedron.

The options available for use with the solid elements are summarized in Tables 6.1 and 6.2 and in Chapter 7.

6.5 Frame elements

Frame elements are modeled by line elements that include axial, bending and in some transverse shear deformations. The elements can treat small or large displacement problems in two or three dimensional geometries. A frame element is included using the commands:

```
MATErial ma
  FRAMe
      ....
```

All frame elements have two nodes. Material data must be specified in addition to the `FRAMe` command and *must* include at least `ELAStic` constitutive data and the cross-section specification. Options for cross section specifications are defined in Sect. 7.16. In addition to material and cross-section data, element loading types also may be added (see Sect. 7.16.3 for details). For 3-D problems, a description of the orientation of the cross-section with respect to the frame axis must be provided (see Sect. 5.11). The available element type options are described next.

6.5.1 Two-dimensional elements

Four types of elements are provided in two dimensions:

- A small displacement element based on Euler-Bernoulli theory (no shear deformation);
- A small displacement element based on Timoshenko theory (with shear deformation);
- A large displacement, small rotation element (2nd order theory, no shear deformation);
- A large displacement, large rotation element with shear deformation.

All element types can consider linear elastic behavior and the elements with shearing deformation can consider inelastic behavior for bending and axial effects but retain linear elastic response in the transverse shear terms. Selection of each type of element is described in Table 6.3. *Note in particular the interpolation order used for the bending behavior of each element type.* Some are linear and some are cubic. This can strongly affect the accuracy, especially if only one element is used for each structural member.

6.5.2 Three-dimensional elements

Two types of elements are provided for three dimensions:

- A small deformation element based on Euler-Bernoulli theory;

Type: FRAMe	Command Names	Interp. Order	Material Models			
			Elastic	Viscoelastic	Plastic	User
Euler-Bernoulli	SMALl SHEAr OFF	Cubic	Yes	Yes	Yes	Yes
Timoshenko	SMALl SHEAr ON	Linear	Yes	Yes	Yes	Yes
Timoshenko (linear strains)	SMALl SHEAr ON ENHAnce	Cubic	Yes	Yes	Yes	Yes
Euler-Bernoulli (2nd order)	FINIte SHEAr OFF	Cubic	Yes	Yes	Yes	Yes
Timoshenko	FINIte SHEAr ON	Linear	Yes	Yes	Yes	Yes

Table 6.3: Options and Material Models for 2-D Frame Elements

Type: FRAMe	Command Names	Interp. Order	Material Models			
			Elastic	Viscoelastic	Plastic	User
Euler-Bernoulli	SMALl SHEAr OFF	Cubic	Yes	Yes	Yes	Yes
Timoshenko	SMALl SHEAr ON	Linear	Yes	Yes	Yes	Yes
Euler-Bernoulli (Second order)	FINIte SHEAr OFF	Cubic	Yes	Yes	Yes	Yes
Timoshenko (Simo-Vu Quoc)	FINIte SHEAr ON DISPlace	Linear	Yes	Yes	Yes	Yes
Timoshenko (Ibrahimbegovic)	FINIte SHEAr ON MIXEd	Linear	Yes	Yes	Yes	Yes
Timoshenko (Co-rotational)	FINIte SHEAr ON COROtate	Linear	Yes	Yes	Yes	Yes
Timoshenko (Energy-momentum conserving)	FINIte SHEAr ON CONServe	Linear	Yes	No	No	No

Table 6.4: Options and Material Models for 3-D Frame Elements

- A large displacement, large rotation elements based on exact the kinematic for-

mulation of Simo *et al.*,[27,28] a co-rotational form based on the formulation by Crisfield and Jelinić,[29,30] the incremental form of Ibrahimbegovic[31] and the energy conserving form of Simo *et al.*[32]

Selection of the various element types is defined in Table 6.4. The large displacement element includes elastic resultant and one dimensional one-dimensional models with inelastic behavior based on integration over the cross section. Cross sectional shapes are included for thin tubes, solid circular shape, rectangles, angles, channels, and wide flange shapes.

For elastic behavior only the large displacement elements may be used in transient analyses based on the energy-momentum conserving algorithm of Simo.[32]

6.5.3 Material set data: Reference axes

The required material set data for frame elements is the material model, cross section data, any distributed loading, and for three dimensional frames geometric information to orient the coordinate axes of the cross section. To define the orientation of the cross section for a three dimensional analysis it is necessary to define a `REFE`rence `VECT`or, `DIRE`ction, or `NODE`.

```
MATErial ma
  FRAMe
    REFErence VECTor or NODE or AXIAl
      ....
```

6.5.4 Constitutive models

Speification of an `ELAStic` model is required with inelastic models supplemented by plastic or viscoelastic properties. The cross section data is given either as `CROSs` section properties or `SECTion` types. The geometric data for orienting cross section axes is given by `REFErence VECTor` or `REFErence NODE` options.

For example, a visco-elastic model (see Section 7.4 for details on how viscoelastic models are described) is given by

```
MATErial ma
  FRAMe
    ELAStic ISOTropic E nu
    CROSs   SECTion   A  Ixx Iyy Ixy J ...
    VISCoelastic    1 mu_1 lambda_1
      ....
```

where `E` is a linear elastic modulus, `nu` is Poisson's ratio, `mu_1, lambda_1` are relaxation parameters and `A, Ixx, Iyy,` etc. are cross section data.

Similarly, a plasticity model for *Mises* type plasticity with no hardening (see Section 7.5 for details on specifying plasticity models) is given by

```
MATErial ma
  FRAMe
    ELAStic ISOTropic E nu
    CROSs   SECTion   A  Ixx Iyy Ixy J ...
    PLAStic MISEs     Y
      ....
```

where `E` is a linear elastic modulus, `nu` is Poisson's ratio, `Y` is the uniaxial yield stress and `A, Ixx, Iyy,` etc. are cross section data.

Inertial effects are included in all frame elements. These are activated by specifying a density and mass type

```
DENSITY mass rho
MASS <LUMP, DIAGonal, CONSistent, other> a, r, rot
```

For a diagonal (lumped) mass the factor 'a' is zero, for a consistent mass the factor 'a' is unity. It is possible to interpolate between the two by using an `other` type with the factor: $0 < a < 1$. For diagonal mass, it is possible also to set a rotational factor '`r`' (for explicit time solutions). Some 3-d elements also include rotatory inertia effects which may be activated by setting '`rot`' to any non-zero value.

6.6 Truss elements

The *truss elements* include small and large deformation formulations. The elements have two nodes and include a number of one dimensional constitutive models as indicated in the next chapter. The truss element is included using the commands:

```
MATErial ma
  TRUSs
      ....
```

Required data is material model (e.g., typically `ELAStic`) and cross section `CROSs` giving the area of the section.

6.7 Plate elements

The *plate element* is restricted to small deformation applications in which only the bending response of flat slabs is included. The problem is treated as a two-dimensional

problem for the mesh (in the x_1-x_2 coordinate plane). At present only linear thermo-mechanical response is included for the material models. Each element may be a three node triangle or a four node quadrilateral. The plate element is included using the commands:

```
MATErial ma
  PLATe
      ....
```

Required data is the material model (e.g., `ELAStic`) and the thickness given by the `THICk` option.

6.8 Shell elements

The *shell elements* are capable of both small and finite deformation analysis. Both resultant and through thickness integration forms are available for the small displacement formulation. For the through thickness formulation all the constitutive forms available for the two-dimensional small deformation analyses are also available for the shell. The resultant formulation is restricted to elastic behavior. The large displacement element is also currently restricted to an elastic resultant formulation. The small deformation model includes bending and membrane strains only - no transverse shearing deformation is included - thus restricting application to thin shell problems only. The finite deformation shell is based on the energy-momentum conserving development of Simo et. al.[33] and includes exact large displacement deformations for membrane, bending and shearing strains. Since the formulation is based on the energy-momentum algorithm it is necessary to specify a `TRAN`sient analysis with either the `STAT`ic or `CONS`erving options (see chapter on transient solutions). Also, the tangent matrix is non-symmetric, thus to achieve quadratic rates of convergence the `UTAN`gent solution command must also be employed. The shell element is included using the commands:

```
MATErial ma
  SHELl
      ....
```

Required data is the material model (e.g., `ELAStic`) and the thickness given by the `THICk` option specified as

```
  THICk,,h,kappa,q
```

where h is the thickness, κ the shear factor (for finite deformation only), and q the number of quadrature points through the thickness (small deformation only). If $q = 0$ a resultant form of linear elastic model is used.

For situations where the shell element set has a flat geometry the rotation parameter normal to the shell (often called a "drill" rotation) has no associated stiffness. If the

elements all lie in a coordinate plane the degree of freedom associated to the drill may be eliminated using a restrained boundary condition (e.g., by setting the appropriate parameter using a `BOUN`, `EBOU`, `CBOU` or `MBOU` mesh set as describe elsewhere in this manual or in Appendix A).

6.8.1 Stress and deformation outputs

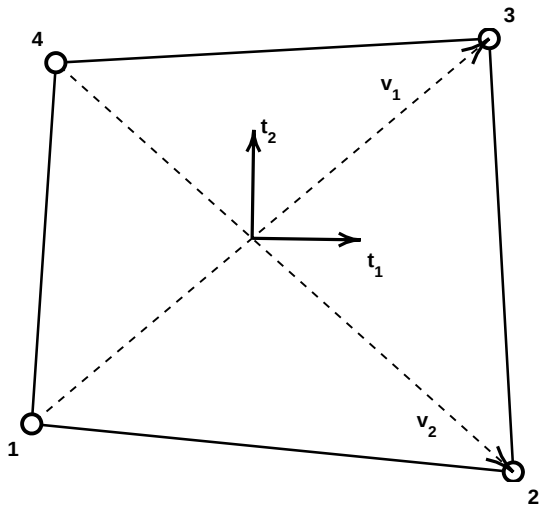


Figure 6.1: Local axes for shell stress computation.

The output and projection of the stress and deformation measures to nodes is carried out for shell elements. Stress output is given with respect to local axes as shown in Fig. 6.1. Local coordinate bases $(\mathbf{t}_1, \mathbf{t}_2)$ are computed from the nodal coordinates using

$$\begin{aligned} \mathbf{v}_1 &= \frac{\mathbf{x}_3 - \mathbf{x}_1}{|\mathbf{x}_3 - \mathbf{x}_1|} \quad ; \quad \mathbf{v}_2 = \frac{\mathbf{x}_2 - \mathbf{x}_4}{|\mathbf{x}_2 - \mathbf{x}_4|} \\ \mathbf{t}_1 &= \frac{\mathbf{v}_1 + \mathbf{v}_2}{|\mathbf{v}_1 + \mathbf{v}_2|} \quad ; \quad \mathbf{t}_2 = \frac{\mathbf{v}_1 - \mathbf{v}_2}{|\mathbf{v}_1 - \mathbf{v}_2|} \end{aligned} \tag{6.1}$$

The average element shell normal is then computed using

$$\mathbf{n} = \mathbf{t}_1 \times \mathbf{t}_2 = \mathbf{v}_1 \times \mathbf{v}_2 \tag{6.2}$$

For the general small deformation three dimensional shell elements the stresses are projected as indicated in Table 6.5. Only the first sixteen (16) components are available when the integrated through the thickness form is used. For the finite deformation shell element components for the resultants in each shell are output as indicated in Table 6.6.

The directions for the ij-components in shells are computed in each shell element individually. As projected, it is assumed that when the local numbers of the shell are given by the order 1-2-3-4 the 1-2 edge is adjacent to the 4-3 edge of an adjacent element; the 2-3 edge is adjacent to the 4-1 edge of an adjacent element; the 3-4 edge is adjacent to the 2-1 edge of an adjacent element; and the 4-1 edge is adjacent to the 3-2 edge of an adjacent element. Thus, if meshes are generated with element numbering given in any different way between elements the nodal projections may not be an accurate representation of the stress state. Principal values should be a good measure however.

Component Number	Value Name	Description
1	N_{11}	Membrane normal force/unit length
2	N_{22}	Membrane normal force/unit length
3	N_{12}	Membrane shear force/unit length
4	N_1	Membrane 1-principal force/unit length
5	N_2	Membrane 2-principal force/unit length
6	M_{11}	Normal bending moment/unit length
7	M_{22}	Normal bending moment/unit length
8	M_{12}	Shear twist moment/unit length
9	M_1	1-Principal bending moment/unit length
10	M_2	2-Principal bending moment/unit length
11	ϵ_{11}	Normal membrane strain
12	ϵ_{22}	Normal membrane strain
13	ϵ_{12}	Shear membrane strain
14	χ_{11}	Normal bending curvature
15	χ_{22}	Normal bending curvature
16	χ_{12}	Shear bending twist
17	σ_{11}	Normal stress at top
18	σ_{22}	Normal stress at top
19	σ_{12}	Shear stress at top
20	σ_{11}	Normal stress at bottom
21	σ_{22}	Normal stress at bottom
22	σ_{12}	Shear stress at bottom
23	$\lvert\sigma_1\rvert$	Maximum 1-principal stress top & bottom
24	$\lvert\sigma_2\rvert$	Minimum 2-principal stress top & bottom

Table 6.5: Small deformation shell stress projections

Component Number	Value Name	Description
1	N_{11}	Membrane normal force/unit length
2	N_{22}	Membrane normal force/unit length
4	N_{12}	Membrane shear force/unit length
5	Q_1	Transverse 1-shear force/unit length
6	Q_2	Transverse 2-shear force/unit length
7	M_{11}	Normal bending moment/unit length
8	M_{22}	Normal bending moment/unit length
10	M_{12}	Shear twist moment/unit length
11	N_1	Principal membrane force/unit length
12	N_2	Principal membrane force/unit length
13	M_1	Principal bending moment/unit length
14	M_2	Principal bending moment/unit length

Table 6.6: Finite deformation shell stress projections

6.9 Membrane elements

The *membrane elements* are derived from the shell elements by deleting the bending and shearing deformations, thus leaving only the in-plane strain deformation terms. Elements for small and large displacements are included but are restricted in the current release to elastic behavior. The membrane element is included using the commands:

```
MATErial ma
  MEMBrane
      ....
```

Required data is the material model (e.g., `ELAStic`) and the thickness given by the `THICk` option.

6.10 Interface element

Interface elements are provided to model surface behavior between two different sets of elements, usually with different material behavior. An interface element consists of nodal pairs that describe the surface topology of each element. Each pair of nodes have the same coordinates (no thickness to a surface element) associated with the two surfaces. It is anticipated that some degrees of freedom of each nodal pair are linked so that the solution variable is the same, while others have different values depending on the behavior of the surface model.

The material set data for an interface set begins with the three commands:

```
MATErial ma
  INTErface
    TYPE  <1, 2, 3, 4, 5>
      ....
```

In general the remaining data will depend on the models that are programmed in the modules as described in Section 7.12.

6.11 Point element

The *point elements* are restricted to linear elastic behavior with linear dash pot and point mass. The point element material set is included using the commands:

```
MATErial ma
  POINt
    MASS   m
    DAMPer c
    SPRIng k
    ORIEnt v_1,v_2,v_3 (ndm values)
```

The `ORIEnt` vector is used to describe the direction cosines for the orientation of the dash pot and spring. The input order for `MASS, DAMPer, SPRIng` and `ORIEnt` is arbitrary. Unspecified terms are assumed zero. The `ORIEnt` command is required if a damper or spring is specified.

The point element is assigned to a 2-node element and may be used to specify a linear stiffness, damper, or mass between the two nodes. If an effect is wanted on a single node see Sect. 8.

6.12 Pressure: Follower loads

The *pressure load element* is specified by material set records:

```
MATErial ma
  PRESsure
   LOAD,<NODAl> p prop-ld
   PLOT,<ON,OFF>    !  PLOT surface on/off: Default OFF
   <PLANe,AXISym>   ! 2-d types: Default PLANe
   <DEAD,FOLLower>  ! Default DEAD
   QUAD,<NODA,GAUS> ! Default GAUSs
   <LINE,SURFace>   ! Default LINE in 2-d; SURFace in 3-d
      ...
```

Loading is specified by options `LOAD` or `NODAl` and, for follower loads by `FINIte` or `FOLLower`. Loading intensity may be associated with the proportional loading number `prop-ld`. If the `NODAl` option is used, the values for each node must be input using the `TEMP` mesh command (see Appendix A), or users must program a module that stores them in the global array.

6.13 Gap element

The *gap element* requires very little data to use. Two options for use are provided. In the first form, the material record is given as:

```
MATErial ma
  GAP
    DIREction,x_dir
    DEGRee,n_dof
    <LAGRange,PENAlty>,pen_value
                              ! blank termination record
```

where `x_dir` is an integer ranging from 1 to ndm; `n_dof` is the degree-of-freedom to which the gap condition is applied and `pen_value` is a penalty parameter used to enforce the constraint. The gap element is used with a two node element where, if `x_dir` is positive, the first to second node indicate a positive direction to enforce the constraint and if `x_dir` is negative the first to second node are taken in a negative coordinate sense. If `n_dof` has the same value as the absolute value of `x_dir` the gap is treated in a physical sense. However, if it is different, a 'gap' condition between the displacements of the two nodes is used. Thus, for the equal sense and a positive `x_dir` a movement of the second node in a positive `x_dir` relative to the first node *opens* the gap without restraint or reduces the restraint force until an opening takes place. A negative motion of the second node relative to the first closes the gap, and when the distance between the two is negative or zero a penalty restraint is inserted. If `x_dir` is negative an opposite interpretation to the above is used. If the penalty is too small an overlap of the regions will exist and if it is considered to be excessive either the penalty parameter value should be increased or an augmented Lagrangian solution should be performed.

A fully Lagrange multiplier form of the gap element may also be used by replacing the solution option `PENAlty` with `LAGRange` multiplier. Solution values for the Lagrange multiplier are stored separately from the nodal values, and thus may not be output easily. Special care must be used when using any Lagrange multiplier solution method as no diagonal results in the tangent solution matrix for this equation. To avoid solution difficulties it is usually required to use a direct solution method – this is the default solver and alternatively may be specified using either of the commands:

```
DIREct          ! In-core solver
DIREct BLOCk    ! Out-of-core blocked solver
```

while in `BATCh` or `INTEractive` solution mode.

The second form for input of a gap element is given by

```
MATErial ma
  GAP
    NORMal n_1 n_2 n_3
    <LAGRange,PENAlty>,pen_value
                              ! blank termination record
```

The gap is constrained only in the normal direction specified by the `n_i` values. For 2-d problems the `n_3` is omitted. this form permits constraint on constant sloping surfaces.

6.14 Acoustic element

The *acoustic elements* are restricted to linear behavior of the Helmholtz wave equation in 1, 2 or 3 dimensional problems. The basic input for the material set is given by

```
MATErial ma
  ACOUstic
    FLUID velocity c
    MASS <LUMP,CONS,    >
    BODY weight (Q_i, i=1,ndm)
    QUAD <NODAl,GAUSs,    >
    GROUnd accel  (Q_i, P_i, i=1,ndm)
    <AXISymmetric, PLANe>
       ! Blank termination record
```

where `c` is the acoustic wave velocity in the medium; `Q_i` are components of loading in the `i`-coordinate direction; `Q_i` are proportional load numbers. The parameters `MASS`, `BODY`, `QUAD` and `GROUnd` are optional. Also the parameter `AXIS` or `PLANe` is required only for 1 or 2 dimensional problems. Instead of the acoustic wave velocity, the elastic bulk modulus and material density may be specified using

```
MATErial ma
  ACOUstic
    FLUID
      BULK modulus K
      DENSity mass rho
        ...
```

The acoustic element also permits specification of interface conditions between solids, rigid base or free surface, as well as, a radiation condition replacing an infinite direction.

These options use elements with one spatial dimension less than that of the acoustic fluid. The interface between an acoustic fluid and a small-deformation solid is given as

```
MATErial ma
  ACOUstic
    INTErface density rho
```

The radiation condition is specified as

```
MATErial ma
  ACOUstic
    RADIation velocity c
```

or

```
MATErial ma
  ACOUstic
    RADIation
    BULK modulus K
    DENSity mass rho
```

A free surface boundary is specified as

```
MATErial ma
  ACOUstic
  SURFace free g
```

where `g` is the fluid weight per unit volume.

6.15 User elements

The specification of *user elements* must contain a number of an element module which has been added to *FEAP*. Each user developed element module is designated as `subroutine elmtnn(...)`, where `nn` ranges from `01` to `50`. The basic input is

```
MATErial ma
  USER unum mset (doflist(i),i=1,ndf)
  xxxxxx                    ! Additional data records
```

where `unum` is a number between 1 and 50 and `mset` is the element material set number, which if omitted is set to `ma`. The `doflist(i)` is again the location where element degree of freedoms are to be assigned to the global values. If the list is omitted it is set as `1` to `ndf`. Accordingly, a typical set of data for a user element `elmt12` is given as:

```
MATErial ma
  USER     12               ! Use elmt12(...) module
  xxxxxx                    ! Additional data records
                            ! blank termination record
```

Options exist for replacing `USER` by a programmer selected name. The name selected by a programmer may be between 4 and 15 characters, but may not conflict with the names of standard program elements (e.g., the name `SOLI` should not be used since then all standard solid elements may no longer be accessed).[1] Longer names are permitted, thus, `SOLID_SHELL` would be properly interpreted. When using the user selected name the specification of the user element number is not required (however, the input of `USER 12` also still may be used).

Details on implementing a user element may be found in the *FEAP* Programmer Manual.[12]

[1] Users could select `SOLID` and it would select the user element, however, this is not recommended as often the data input uses this even though only the first 4 characters are intended to be interpreted.

Chapter 7

Material models

The data input for each of the current material options is summarized below. Tables are included to indicate which elements types can use each type of data option. As much as possible a common format and notation is used for all the element types.

7.1 Heat conduction material models

For thermal analysis a linear heat conduction capability is included in *FEAP*. The constitutive equation is given by a linear Fourier model in which the heat flux $\mathbf{q}$ is related to the thermal gradient $\mathbf{h} = \boldsymbol{\nabla} T$ by the relation[1]

$$\mathbf{q} \;=\; -\,\mathbf{K}\,\mathbf{h} \tag{7.1}$$

where, in the principal directions,

$$\hat{\mathbf{K}} \;=\; \begin{bmatrix} K_1 & 0 & 0 \\ 0 & K_2 & 0 \\ 0 & 0 & K_3 \end{bmatrix} \;. \tag{7.2}$$

The principal conductivities $\hat{K}$ may be transformed to a global set using

$$\mathbf{K} = \mathbf{R}^T\,\hat{\mathbf{K}}\,\mathbf{R} \tag{7.3}$$

Currently the transformation array may only be specified as a rotation ψ about the global 3-axis (z-axis) which gives

$$\mathbf{R} \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The rotation angle in degrees is given by the `ANGLe` command (see Table 7.1).

The values for K_i and, for transient problems, the specific heat, c, are specified using the command `FOURier`,`ORTHotropic` or for the case where all are equal using `FOURier`,`ISOTropic` as indicated in Table 7.1. The mass density is given by the `DENSity` command. An example material set for an isotropic thermal material is

```
MATErial ma
  THERmal
    FOURier ISOTropic k c
    DENSity mass      rho
      !  Blank record to stop inputs
```

Other available model inputs are described in Table 7.1.

Command	Type	Parameters
FOURier	ISOTropic	K, c
FOURier	ORTHotropic	K_1, K_2, K_3, c
ANGLe		ψ
DENSity		ρ

Table 7.1: Heat Conduction Material Model Data Inputs

7.2 Linear thermo-elastic models

Linear thermo-elasticity constitutive equations are expressed by algebraic relations between the stress tensor σ_{ij} and the strain tensor ϵ_{ij}. In Cartesian tensor notation the relationship may be expressed by:

$$\epsilon_{ij} = c_{ijkl}\,\sigma_{kl} + \epsilon^{th}_{ij} \tag{7.4}$$

where c_{ijkl} are components of the fourth-order elastic compliance tensor and ϵ^{th}_{ij} those of the thermal expansion strain. Traditionally, finite element developments have been developed in a matrix form where the components of strain and stress are expressed using a *Voigt notation*. In *FEAP* the ordering is taken as

$$\boldsymbol{\sigma} = \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{Bmatrix} \quad \text{and} \quad \boldsymbol{\varepsilon} = \begin{Bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ 2\,\epsilon_{12} \\ 2\,\epsilon_{23} \\ 2\,\epsilon_{31} \end{Bmatrix} = \begin{Bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ \gamma_{12} \\ \gamma_{23} \\ \gamma_{31} \end{Bmatrix} \tag{7.5}$$

where $\gamma_{ij} = 2\,\epsilon_{ij}$ are *engineering shear strains*. Users should note that this ordering may be different than that used in other sources. For example in the original paper by Voigt the ordering was taken as[34,35]

$$\boldsymbol{\sigma} = \begin{bmatrix}\sigma_{11} & \sigma_{22} & \sigma_{33} & \sigma_{23} & \sigma_{13} & \sigma_{12}\end{bmatrix}^T \tag{7.6}$$

and in Abaqus the ordering is[36]

$$\boldsymbol{\sigma} = \begin{bmatrix}\sigma_{11} & \sigma_{22} & \sigma_{33} & \sigma_{12} & \sigma_{13} & \sigma_{23}\end{bmatrix}^T \tag{7.7}$$

This implies users must be very careful in specifying elastic properties for any non-isotropic model.

Using Voigt notation a linear elastic material model in *FEAP* may be given by[16]

$$\boldsymbol{\epsilon} = \mathbb{C}\,\boldsymbol{\sigma} + \boldsymbol{\epsilon}^{th} \tag{7.8}$$

where $\boldsymbol{\epsilon}$ and $\boldsymbol{\sigma}$ are the stress and strain arrays and $\mathbb{C}$ is the elastic compliance array which has the form

$$\begin{aligned}\mathbb{C} &= \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \\ &= \begin{bmatrix} c_{1111} & c_{1122} & c_{1133} & 2\,c_{1112} & 2\,c_{1123} & 2\,c_{1131} \\ c_{2211} & c_{2222} & c_{2233} & 2\,c_{2212} & 2\,c_{2223} & 2\,c_{2231} \\ c_{3311} & c_{3322} & c_{3333} & 2\,c_{3312} & 2\,c_{3323} & 2\,c_{3331} \\ 2\,c_{1211} & 2\,c_{1222} & 2\,c_{1233} & 4\,c_{1212} & 4\,c_{1223} & 4\,c_{1231} \\ 2\,c_{2311} & 2\,c_{2322} & 2\,c_{2333} & 4\,c_{2312} & 4\,c_{2323} & 4\,c_{2331} \\ 2\,c_{3111} & 2\,c_{3122} & 2\,c_{3133} & 4\,c_{3112} & 4\,c_{3123} & 4\,c_{3131} \end{bmatrix} \end{aligned} \tag{7.9}$$

For analysis purposes the model in inverted to:

$$\boldsymbol{\sigma} = \mathbb{D}\,\left[\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^{th}\right] = \mathbb{D}\,\boldsymbol{\epsilon} + \boldsymbol{\beta}^{th} \tag{7.10}$$

where $\mathbb{D}$ is the elastic modulus array defined as:

$$\mathbb{D} = \mathbb{C}^{-1} \tag{7.11}$$

and

$$\boldsymbol{\beta}^{th} = -\,\mathbb{D}\,\boldsymbol{\epsilon}^{th} \quad . \tag{7.12}$$

FEAP permits use of either *isotropic*, *transversely isotropic*, *orthotropic* or general *anisotropic* linear elastic models which include both mechanical and thermal effects.

7.2.1 Isotropic linear elastic models

An isotropic model is defined by two independent elastic parameters for $\mathbb{C}$ and two parameters for $\boldsymbol{\beta}^{th}$.

The elastic parameters used to define $\mathbb{C}$ are taken as Young's modulus, E, and Poisson's ratio, ν. The elastic compliance array is defined as:

$$\mathbb{C} = \begin{bmatrix} \frac{1}{E} & -\frac{\nu}{E} & -\frac{\nu}{E} & 0 & 0 & 0 \\ -\frac{\nu}{E} & \frac{1}{E} & -\frac{\nu}{E} & 0 & 0 & 0 \\ -\frac{\nu}{E} & -\frac{\nu}{E} & \frac{1}{E} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G} \end{bmatrix} \tag{7.13}$$

with the shear modulus G related through

$$G = \frac{E}{2\,(1+\nu)} \quad . \tag{7.14}$$

For isotropic materials the thermal strain is given by

$$\boldsymbol{\epsilon}^{th} = \begin{bmatrix} \alpha \\ \alpha \\ \alpha \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta T \tag{7.15}$$

where α is the coefficient of linear thermal expansion and $\Delta T = T - T_0$, where T_0 is the temperature where thermal strains vanish.

The data input for the thermo-mechanical isotropic model for material set 1 is given as:

```
MATErial ma
  SOLId
  ELAStic ISOTropic E     nu
  THERmal ISOTropic alpha T0
                           ! blank termination record
```

Additional data options and parameters are defined in Table 7.2 and Table 7.3 describes the added options which may be added to some of the models.

For problems in which no thermal effects are included it is not necessary to specify values for α and T_0.

Command	Type	Parameters
ELAStic	ISOTropic	E, ν
ELAStic	ORTHotropic	$E_1, E_2, E_3, \nu_{12}, \nu_{23}, \nu_{31}, G_{12}, G_{23}, G_{31}$
ELAStic	TRANsverse	$E_1, E_2, \nu_{12}, \nu_{31}, G_{31}$
DAMPing	RAYLeigh	a0 , a1
PLAStic	MISEs	Y_0, Y_∞, β
PLAStic	DRUCker	σ_t, σ_c
PLAStic	LODE	σ_t, σ_c
PLAStic	YIELd	$Y_0, Y_\infty, \beta, H_{iso}$
PLAStic	SWIFt	$K, \epsilon_0, n, H_{iso}$
PLAStic	HILL	$R_{11}, R_{22}, R_{33}, R_{12}, R_{23}, R_{31}$
PLAStic	KINEmatic	$H_2, H_3, H_4, H_5, H_6, J_1$
PLAStic	GENEralized	Y_0, Y_∞, β
PLAStic	HARDening	H_{iso}, H_{kin}
VISCoelastic		μ_i, τ_i
DAMAge		g_{limit}, r_{rate}
THERmal	ISOTropic	α, T_0
THERmal	ORTHotropic	$\alpha_1, \alpha_2, \alpha_3, T_0$
FOURier	ISOTropic	K, c
FOURier	ORTHotropic	K_1, K_2, K_3, c
FOURier	REFErence	T_{ref}
TEMPerature		$tdof$ - degree of freedom number
ANGLe		ψ
DENSity		ρ
STARt	ELAStic	First iteration always elastic (default)
STARt	INELastic	First iteration can be inelastic.

Table 7.2: Material Model Data Inputs

The elastic moduli for all cases except plane stress is given by

$$\mathbb{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\nu) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-2\nu)/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & (1-2\nu)/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & (1-2\nu)/2 \end{bmatrix} . \tag{7.16}$$

For plane stress the condition $\sigma_{33} = 0$ is enforced to give

$$\epsilon_{33} = -\frac{\nu}{E}\,(\sigma_{11} + \sigma_{22}) + \alpha\,\Delta T \tag{7.17}$$

and

$$\mathbb{D} \;=\; \frac{E}{(1-\nu^2)} \begin{bmatrix} (1-\nu) & \nu & 0 & 0 & 0 & 0 \\ \nu & (1-\nu) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-\nu)/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{7.18}$$

In general, all constitutive models in *FEAP* are defined in terms of all possible stress/strain components. For plane stress or strain and for axisymmetric deformation without torsion the components ϵ_{23} and ϵ_{31} are zero and thus also give σ_{23} and σ_{31} as zero.

7.2.2 Orthotropic linear elastic models

The linear orthotropic elastic material model in *FEAP* is expressed in the *principal material directions* as

$$\hat{\boldsymbol{\epsilon}} \;=\; \hat{\mathbb{C}}\,\hat{\boldsymbol{\sigma}} \;+\; \hat{\boldsymbol{\epsilon}}^{th} \tag{7.19}$$

where $\hat{\boldsymbol{\epsilon}}$ and $\hat{\boldsymbol{\sigma}}$ are the stress and strain arrays in the principal material directions and the elastic compliance array $\hat{\mathbb{C}}$ in principal material directions is given by:

$$\hat{\mathbb{C}} \;=\; \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{12}}{E_1} & -\frac{\nu_{13}}{E_1} & 0 & 0 & 0 \\ -\frac{\nu_{21}}{E_2} & \frac{1}{E_2} & -\frac{\nu_{23}}{E_2} & 0 & 0 & 0 \\ -\frac{\nu_{31}}{E_3} & -\frac{\nu_{32}}{E_3} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{12}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{23}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{31}} \end{bmatrix} \tag{7.20}$$

where E_i are Young's moduli in principal directions, ν_{ij} are Poisson ratios for strains measured in principal directions and G_{ij} are shear moduli for the principal directions.

Elastic Input Option	Type	Thermal	Visco-Elastic	Plastic	Gen. Plastic	Complex Moduli
`COMP`liances	Anisotropic	X				
`MODU`li	Anisotropic	X				
`ORTH`otropic	Orthotropic	X		X		
`TRAN`sverse	Transverse	X				
	Isotropic	X				
`ISOT`ropic	Isotropic	X	X	X	X	X

Table 7.3: Small deformation models for solid elements

The above sign convention corresponds to

$$C_{ii} = \frac{1}{E_i} \quad \text{and} \quad C_{ij} = -\frac{\nu_{ij}}{E_i} \quad \text{for} \quad i,j = 1,2,3 \tag{7.21}$$

and the definition of terms is identical to that given by Christensen[37] (except for shear modulus terms). **Users should be careful that the above form agrees with the way that elastic properties for moduli and Poisson ratios are defined in other sources and also recall the form of the Voigt notation used in FEAP**.

The thermal strain is given by:

$$\hat{\boldsymbol{\epsilon}}^{th} \;=\; \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta T \;=\; \hat{\boldsymbol{\alpha}}\, \Delta T \tag{7.22}$$

where

$$\Delta T \;=\; T \,-\, T_0\,, \tag{7.23}$$

α_i are coefficients of linear thermal expansion and T_0 is a specified reference temperature.

The orthotropic material parameters are input as shown in Table 7.2 using the commands `ELAStic,ORTHotropic` and `THERmal,ORTHotropic`. For 2-dimensional analyses the values of G_{23} and G_{31} are not used and may be omitted. The angle the principal directions makes with the x_1 (or x) axis for plane stress and plane strain analyses or the r axis for axisymmetric analysis may be specified using the material `ANGLe` command as shown in Table 7.11. Using this angle *FEAP* transforms the input material compliances to

$$\mathbb{C} = \mathbf{R}^T\, \hat{\mathbb{C}}\, \mathbf{R} \tag{7.24}$$

and converts the constitutive equation to the form given in Eqs. 7.10 to 7.12.

Material data for this option is given by the command set:

```
MATErial ma
  SOLId
  ELAStic ORTHotropic e1 e2 e3 nu12 nu23 nu31 g12 g23 g31
  THERmal ORTHotropic a1 a2 a3 t0
  ANGLe   axis-1      psi
                            ! blank termination record
```

The `ANGLe` command describes the angle in degrees which the principal material axis 1 makes with the x_1 axis. For the transformation defining $\mathbf{R}$ it is assumed that the principal material axis 3 coincides with the direction of the x_3 axis.

Principal axis orientation

A more general option exists to describe the orientation of the principal material axes. This option is described by two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ which lie in the plane of the principal material axes 1 and 2. For this case it is necessary to include the statement

```
VECTor ORTHotropic v11 v12 v13 v21 v22 v23
```

where `v1i, v2i` are components of vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. These vectors are converted to two orthogonal unit vectors $\bar{\mathbf{v}}_1$ and $\bar{\mathbf{v}}_2$ using

$$\begin{aligned}
\bar{\mathbf{v}}_1 &= \frac{\mathbf{v}_1}{(\mathbf{v}_1 \cdot \mathbf{v}_1)^{1/2}} \\
\mathbf{N} &= \mathbf{v}_1 \times \mathbf{v}_2 \quad ; \quad \mathbf{v}_2 = \mathbf{N} \times \mathbf{v}_1 \\
\bar{\mathbf{v}}_2 &= \frac{\mathbf{v}_2}{(\mathbf{v}_2 \cdot \mathbf{v}_2)^{1/2}}
\end{aligned}$$

Thus, the final axes are always oriented in the direction of the input vector $\mathbf{v}_1$.

Material data for this option is given by the command set:

```
MATErial ma
  SOLId
  ELAStic ORTHotropic e1 e2 e3 nu12 nu23 nu31 g12 g23 g31
  THERmal ORTHotropic a1 a2 a3 t0
  VECTo   ORTHotropic v11 v12 v13 v21 v22 v23
                           ! blank termination record
```

If anisotropic plasticity is added to the model the orthotropic vectors are assumed to coincide with both the elastic and plastic principal axes.

7.2.3 Transversely isotropic linear elastic models

The transversely isotropic linear elastic material model in *FEAP* is expressed in the *principal material directions* in the same way as for the orthotropic model. It is assumed that the principal material directions 1 and 3 define a plane of isotropy; 5 elastic constants are necessary to define the compliance array $\hat{\mathbb{C}}$ in the principal material directions. Using the definition of the compliance array given in Eq. (7.20) the parameters defining the transversely isotropic model satisfy the constraints:

$$E_1 = E_3 \quad ; \quad \nu_{12} = \nu_{13} \quad ; \quad G_{31} = \frac{E_1}{2(1+\nu_{31})} \quad \text{and} \quad G_{12} = G_{23} \tag{7.25}$$

Thus, only E_1, E_2, ν_{12}, ν_{23} and G_{12} need be specified for the model.

The coefficients of thermal expansion for the thermal strain are specified using the orthotropic model with

$$\alpha_1 = \alpha_3 \tag{7.26}$$

The compliance matrix for the above transversely isotropic material becomes:

$$\hat{\mathbb{C}} = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{12}}{E_1} & -\frac{\nu_{12}}{E_1} & 0 & 0 & 0 \\ -\frac{\nu_{21}}{E_2} & \frac{1}{E_2} & -\frac{\nu_{23}}{E_2} & 0 & 0 & 0 \\ -\frac{\nu_{12}}{E_1} & -\frac{\nu_{32}}{E_3} & \frac{1}{E_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{12}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{23}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{31}} \end{bmatrix} \tag{7.27}$$

The transversely isotropic material parameters are input as shown in Table 7.2 using the commands `ELAStic,TRANsverse` and `THERmal,ORTHotropic`. Material data is specified by the command set:

```
MATErial ma
  SOLId
  ELAStic TRANsverse  e1 e2 nu12 nu31 g12
  THERmal ORTHotropic a1 a2 a1 t0
  ANGLe   axis-1      psi
                            ! blank termination record
```

The angle the principal directions makes with the x_1 (or x) axis for plane stress and plane strain analyses or the r axis for axisymmetric analysis may be specified using the material `ANGLe` command as shown in Table 7.11. The `ANGLe` command describes the angle in degrees which the principal material axis 1 makes with the x_1 axis. For the transformation defining $\mathbf{R}$ it is assumed that the principal material axis 3 coincides with the direction of the x_3 axis.

7.2.4 Anisotropic linear elastic models

A linear anisotropic elastic material model in *FEAP* may be input as either *compliances* or as *moduli*. In compliance form the input is given as:

```
MATErial ma
  SOLId
  ELAStic COMPliance n
     C11 C12  ... C1n
      ...
     Cn1 Cn2  ... Cnn
```

This form permits inputs for two-dimensional analyses as a size 4 array and for three-dimensional analyses as a full size 6 array. The compliance form assumes the constitutive model as

$$\boldsymbol{\epsilon} = \mathbb{C}\,\boldsymbol{\sigma} \tag{7.28}$$

which is inverted to the form needed by elements as form assumes the constitutive model as

$$\boldsymbol{\sigma} = \mathbb{D}\,\boldsymbol{\epsilon} \tag{7.29}$$

where $\mathbb{D} = \mathbb{C}^{-1}$.

To input moduli directly, the input sequence

```
MATErial ma
  SOLId
  ELAStic MODUli n
     D11 D12  ... D1n
      ...
     Dn1 Dn2  ... Dnn
```

is used.

If a plane stress option is to be performed, the above moduli must be reduced prior to input. Use of the `PLANe STREss` option will not perform the required modifications.

Additional data options to describe materials and their parameters are defined in Table 7.2.

The types of elements for which elastic material models may be specified is indicated in Table 7.4.

Command	Solid	Truss	Frame	Plate	Shell	Membrane	Thermal
ELAStic	X	X	X	S	X	X	-
PLAStic	X	X	F	-	S	-	-
VISCoelastic	X	X	F	-	-	-	-
THERmal	X	X	X	X	-	X	-
FOURier	X	X	-	-	-	-	X
ANGLe	X	-	-	X	X	X	X
DENSity	X	X	X	X	X	X	X

Table 7.4: Material Commands vs. Element Types. X=all, F=finite, S=small.

7.2.5 Thermo-elastic models

Then coupled thermo-mechanical behavior is analyzed the properties for both mechanical and thermal models may be specified in a single material set. Accordingly, for an isotropic thermal model the input is specified by

```
MATErial ma
  SOLID
    ELAStic ISOTropic E     nu
    THERmal ISOTropic alpha T0
    FOURier ISOTropic k     c
    DENSity mass      rho
    ! Blank record
```

Solid elements will assign the temperature to the degree of freedom that is one more than the mesh space dimension. If a different degree of freedom is used the material set command

```
    TEMPerature ,, nt
```

is added with `nt` the value.

When significant rate effects are present, such as in wave propagation problems, the balance equation for temperature becomes

$$\rho\, c\, \frac{\partial T}{\partial t} = -\mathrm{div}\mathbf{q} - T_{ref}\, \boldsymbol{\beta}^T \dot{\boldsymbol{\epsilon}} + Q \tag{7.30}$$

where T_{ref} is an absolute temperature about which the linear equation is solved. Thus, in addition to the elastic moduli, thermal expansion coefficients it is necessary to also provide the value of the absolute temperature using the command

```
FOURier REFErence Tref
```

where `Tref` specifies the value.

7.3 Finite deformation models

Finite deformation hyper-elastic models are provided in *FEAP* for several stored energy functions which are written in terms of deformation measures.

Deformation measures may be defined in terms of positions in the reference configuration, denoted by $\mathbf{X}$, and positions in the current configuration, denoted by $\mathbf{x}$. The motion of a point from the reference to the current configuration at time t is expressed as

$$\mathbf{x} = \boldsymbol{\varphi}(\mathbf{X}, t) \tag{7.31}$$

The deformation gradient is then defined as

$$\mathbf{F} = \frac{\partial \boldsymbol{\varphi}}{\partial \mathbf{X}} \ . \tag{7.32}$$

Additional measures of deformation are given by the right Cauchy-Green deformation tensor

$$\mathbf{C} = \mathbf{F}^T \mathbf{F} \tag{7.33}$$

and the left Cauchy-Green deformation tensor

$$\mathbf{b} = \mathbf{F}\,\mathbf{F}^T \tag{7.34}$$

A measure of strain is provided by the Green strain

$$\mathbf{E} = \tfrac{1}{2}\,(\mathbf{C} - \mathbf{1}) \tag{7.35}$$

The types of material models which may be used in the finite deformation solid elements is summarized in Table 7.5. In general all models are elastic; however some include capabilities to include viscoelastic or plastic parts. In the next subsections each of the elastic models is summarized. In Section 7.4 a description of the viscoelastic model is described and in Section 7.5 that for plasticity is given.

Model Name	Input command	Type	Viscoelastic	Plastic
St.Venant-Kirchhoff	`STVK` or `STVE`	Orthotropic		
Energy Conserving	`CONS`	Orthotropic		
Fung model	`FUNG`	Orthotropic		
Orthotropic	`ORTH`	Orthotropic		X
Neohookean	`NEOH`	Isotropic		
Modified Neohookean	`MNEO`	Isotropic	X	
Mooney-Rivlin	`MOON`	Isotropic		
Modified Mooney-Rivlin	`MMOO`	Isotropic		
Ogden	`OGDE`	Isotropic	X	
Logarithmic stretch	`ISOT`	Isotropic	X	X
Arruda-Boyce	`ARRU`	Isotropic		
Yeoh	`YEOH`	Isotropic		

Table 7.5: Finite deformation elastic models for solid elements

7.3.1 Elastic models

The hyper-elastic model expressed in terms of the strain energy function as a function of $\mathbf{C}$ is given by[2,38]

$$\mathbf{S} = 2\,\frac{\partial W(\mathbf{C})}{\partial \mathbf{C}} \tag{7.36}$$

where W is a *stored energy* function. Stress in the current configuration may be deduced by transformation (pushing) the stress. Accordingly

$$\sigma = \frac{1}{J}\,\mathbf{F}\,\mathbf{S}\,\mathbf{F}^T \ . \tag{7.37}$$

Isotropic models may be expressed in terms of the invariants of the deformation tensor. Accordingly, the three principal invariants given by

$$I_C = \operatorname{tr}\mathbf{C} \tag{7.38}$$

$$II_C = \tfrac{1}{2}\left(I_C^2 - \operatorname{tr}\mathbf{C}^2\right) \tag{7.39}$$

and

$$III_C = \det\mathbf{C} = J^2 \tag{7.40}$$

where J is $\det\mathbf{F}$ may be used to write the stored energy function.

The deformation tensor may also be expressed in terms of principal stretches, λ_A, and their associated eigenvectors, $\mathbf{N}_A$. Accordingly, one may write

$$\mathbf{C} = \sum_{A=1}^{3} \lambda_A^2\,\mathbf{N}_A \otimes \mathbf{N}_A \ . \tag{7.41}$$

The invariants are then given by

$$I_C = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \tag{7.42}$$

$$II_C = \lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2 \tag{7.43}$$

and

$$III_C = \lambda_1^2\lambda_2^2\lambda_3^2 \ . \tag{7.44}$$

The three principal stretches may be used directly to write the stored energy function. Both forms are used in *FEAP*.

Other alternative forms may be used to write the stored energy function. One which is often used splits the deformation gradient into *volumetric* and *deviatoric* (i.e., isochoric) parts as

$$\mathbf{F} = \mathbf{F}_{vol}\,\mathbf{F}_{dev} \tag{7.45}$$

where

$$\mathbf{F}_{vol} = J^{1/3}\mathbf{1} \tag{7.46}$$

and

$$\mathbf{F}_{dev} = J^{-1/3}\mathbf{F} \tag{7.47}$$

The determinant of $\mathbf{F}_{dev}$ is unity (which is the definition of an isochoric motion).

Using the split the isochoric part of the deformation tensor becomes

$$\mathbf{C}_{dev} = J^{-2/3}\mathbf{C} = \mathbf{F}_{dev}^T\mathbf{F}_{dev} \quad . \tag{7.48}$$

and again has unit volume change. The stored energy function may then be written in terms of the invariants

$$\bar{I}_C = \operatorname{tr}\mathbf{C}_{dev} = J^{-2/3}\, I_C \tag{7.49}$$

$$\begin{aligned} \bar{I}I_C &= \tfrac{1}{2}\left(\bar{I}_C^2 - \operatorname{tr}\mathbf{C}_{dev}^2\right) \\ &= \tfrac{1}{2}\, J^{-4/3}\left(I_C^2 - \operatorname{tr}\mathbf{C}^2\right) \end{aligned} \tag{7.50}$$

and J.

In *FEAP* the isotropic elastic moduli are defined to match results from the small strain isotropic elastic models. Accordingly, they generally require specification of the elastic modulus, E, and Poisson ratio, ν (or equivalently the bulk modulus, K, and shear modulus, G). Some models require additional parameters to define behavior in the finite deformation range.

7.3.2 St. Venant-Kirchhoff and energy conserving model

The simplest model is a St. Venant-Kirchhoff model given by:[39,40]

$$\mathbf{S} = \mathbb{D}\,\mathbf{E} \tag{7.51}$$

where $\mathbf{S}$ is the second Piola-Kirchhoff stress, $\mathbf{E}$ is the Green strain, and $\mathbb{D}$ are the elastic moduli. This model may be deduced from the stored energy function

$$W = \tfrac{1}{2}\,\mathbf{E}^T\mathbb{D}\,\mathbf{E} \; . \tag{7.52}$$

For isotropy the model may be written in terms of the invariants of $\mathbf{E}$; however, the $\mathbb{D}$ will have the same structure as in an isotropic linear elastic material (see Section 7.2.1).

The material data set for the St. Venant-Kirchhoff model is given as

```
MATErial ma
  SOLId
  ELAStic STVE(or STVK) E nu
                          ! blank termination record
```

for an isotropic material where a `FINIte` statement is optional; or by

```
MATErial ma
  SOLId
  ELAStic ORTHotropic e1 e2 e3 nu12 nu23 nu31 g12 g23 g31
  FINIte
                              ! blank termination record
```

for an orthotropic material where the `FINIte` is required to distinguish from the small deformation case.

Principal axis orientation

If the material axes do not coincide with the principal elastic axes it is necessary to include the statement

```
  VECTor ORTHotropic v11 v12 v13 v21 v22 v23
```

where `v1i, v2i` are components of vectors in the principal material axes 1 and 2. These vectors are converted to two orthogonal unit vectors $\bar{\mathbf{v}}_1$ and $\bar{\mathbf{v}}_2$ using

$$\bar{\mathbf{v}}_1 = \frac{\mathbf{v}_1}{(\mathbf{v}_1 \cdot \mathbf{v}_1)^{1/2}}$$
$$\mathbf{N} = \mathbf{v}_1 \times \mathbf{v}_2 \quad ; \quad \mathbf{v}_2 = \mathbf{N} \times \mathbf{v}_1$$
$$\bar{\mathbf{v}}_2 = \frac{\mathbf{v}_2}{(\mathbf{v}_2 \cdot \mathbf{v}_2)^{1/2}}$$

Energy conserving model

The same constitution is used to implement an energy-momentum algorithm for finite deformation analyses. The data to perform an energy-momentum conserving form is given as

```
MATErial ma
  SOLId
  FINIte
  ELAStic CONServing E nu
                              ! blank termination record
```

Recall that the location of the `FINIte` command is not order dependent so that the commands can also be.

```
MATErial ma
  SOLId
  ELAStic CONServing E nu
  FINIte
                              ! blank termination record
```

Optionally the finite deformation designation also may be given for all elements as GLOBal data as:

```
GLOBAl
  FINIte
                                ! blank termination record
```

: WARNING: The energy conserving form may **not** be combined with any inelastic model (e.g., viscoelastic or elasto-plastic).

Fung model

A variant of the St. Venant-Kirchhoff model which is used in some biomechanics applications is the Fung model expressed by the stored energy function as[38]

$$W(\mathbf{E}) = C\, \exp(\mathbf{E}^T \mathbf{A}\, \mathbf{E}) \; . \tag{7.53}$$

Here the array $\mathbf{A}$ has identical structure to an orthotropic elastic tensor, but is dimensionless, and the parameter C has dimensions of modulus. The input for the Fung model is given as:

```
MATErial ma
  SOLId
  ELAStic FUNG C A_11 A_22 A_33 A_12 A_23 A_31 A_44 A_55 A_66
  ANGLe   axis-1      psi
  FINIte
                                ! blank termination record
```

The St. Venant-Kirchhoff and Energy Conserving models should not be used for problems where large compressive deformations are expected. For the parameters selected, these models give identical results to the small deformation isotropic model if deformations are truly infinitesimal. It is also an acceptable model to use if the displacements are large, but strains remain small. For situations where large elastic deformations are involved the NEOHookean, MNEOhookean, or OGDEn models discussed next should be used. All the available isotropic models and their required inputs are summarized in Table 7.6.

7.3.3 Neo-Hookean and modified neo-Hookean models

The stored energy functions for finite deformation hyper-elastic models are split into two parts. The first part defines the behavior associated with volume changes and the second the behavior for other deformation states. The volumetric deformation part is defined by a function $U(J)$, where J is the determinant of the deformation gradient $\mathbf{F}$,

Command	Type	Parameters
ELAStic	NEOHookean	E, ν
ELAStic	MNEOhookean	E, ν
ELAStic	MOONey-Rivlin	E, ν, c
ELAStic	MMOOney-Rivlin	E, ν, c
ELAStic	OGDEn	$K, C_1, a_1, C_2, a_2, C_3, a_3$
ELAStic	STVK	E, ν
ELAStic	STVE	E, ν
ELAStic	CONServe	E, ν
ELAStic	ARRUda-Boyce	$E, \nu, m(1/n)$
ELAStic	YEOH	E, ν, k_1, k_2

Table 7.6: Isotropic Finite Deformation Elastic Material Models and Inputs

multiplied by a material parameter. The volumetric function in *FEAP* is taken from one of the following:

$$\begin{aligned} 1: &\quad U(J) = \tfrac{1}{4}\left(J^2 - 1 - 2\,\ln J\right) \\ 2: &\quad U(J) = \tfrac{1}{2}\left(J-1\right)^2 \\ 3: &\quad U(J) = \tfrac{1}{2}\left(\ln J\right)^2 \\ 4: &\quad U(J) = [J - 1 - \ln(J)] \end{aligned} \tag{7.54}$$

where $ln(J)$ is the natural logarithm of J. Users may select which of the above models to use during data inputs. By default model number 1 is used.

The neo-Hookean hyper-elastic model is deduced from the stored energy function[2,38,41]

$$W = \left(K - \tfrac{2}{3}G\right)\, U(J) + \tfrac{1}{2}\,G\,\left(I_C - 3 - 2\,\ln(J)\right) \tag{7.55}$$

where I_C is defined in Eq. (7.38). The parameters K and G are equivalent to the small strain bulk and shear moduli, respectively. Input data for the model is specified in terms of the equivalent small strain modulus (E) and Poisson ratio (ν) such that the K and G are given by

$$K = \frac{E}{3\,(1-2\nu)} \quad ; \quad G = \frac{E}{2\,(1+\nu)}\ . \tag{7.56}$$

The data set to use this form is given by

```
MATErial ma
  SOLId
  FINIte
  ELAStic NEOHook E nu
                              ! blank termination record
```

The particular volume model to use may be set as an option on the `FINIte` command as:

```
FINIte VOLUme n
```

where `n` is the model number from 1 to 3 according to those defined by Eq. (7.54).

A modified form to the neo-Hookean model is also available. The modified form defines the stored energy function in terms of the volumetric/deviatoric split of the deformation gradient as described in Eqs (7.45) to (7.47). Accordingly, the stored energy function is given as

$$\begin{aligned} W &= K\,U(J) + \tfrac{1}{2}\,G\,\left(J^{-2/3} I_C - 3\right) \\ &= K\,U(J) + \tfrac{1}{2}\,G\,\left(\bar{I}_C - 3\right) \end{aligned} \tag{7.57}$$

where $\bar{I}_C$ is defined by Eq. (7.49).

The parameters K and G are again specified by their small strain equivalent E and ν defined in Eq. 7.56.

The data set to use the modified form is given by

```
MATErial ma
  SOLId
  FINIte <VOLUme n>
  ELAStic MNEOHook E nu
                              ! blank termination record
```

A quantity within pointed brackets denotes an option; here it denotes the volume model to use.

7.3.4 Mooney-Rivlin model

A Mooney-Rivlin material model is implemented with the stored energy function given by[2,38,41]

$$W = (K - \tfrac{2}{3}G)\,U(J) + \tfrac{1}{2}\,G\,[(1-c)\,(I_C - 3 - 2\,\ln J) + c\,\,(II_C - 3 - 4\,\ln J)] \tag{7.58}$$

where I_C and II_C are defined in (7.38) and (7.39), respectively; K and G are the small strain moduli; and c is the coefficient for the second invariant term. Setting c to zero gives the neo-Hookean model. The volumetric behavior is again given by $U(J)$ as described in Eq. 7.54.

A modified form of the Mooney-Rivlin model is given as

$$W = K\,U(J) + \tfrac{1}{2}\,G\,\left[(1-c)\,\left(\bar{I}_C - 3\right) + c\,\,\left(\bar{II}_C - 3\right)\right] \tag{7.59}$$

where $\bar{I}_C$ and $\bar{II}_C$ are defined in (7.49) and (7.50), respectively. The input data for the Mooney-Rivlin model is given by

```
MATErial ma
  SOLId
  FINIte <VOLUme n>
  ELAStic MOONey E nu c
                              ! blank termination record
```

for the standard model and

```
MATErial ma
  SOLId
  FINIte <VOLUme n>
  ELAStic MMOOney E nu c
                              ! blank termination record
```

for the modified model. The appropriate values for the bulk and shear modulus are computed from Eq. 7.56.

7.3.5 Ogden model

FEAP also contains a model for hyper-elastic behavior which is expressed directly in terms of *deviatoric* principal stretches,[42,43] $\tilde{\lambda}_A$. This model has a stored energy function expressed in the form:

$$W = K\,U(J) + \sum_{A=1}^{3} w(\tilde{\lambda}_A, J) \tag{7.60}$$

and is based on the *Valanis-Landel hypothesis*.[43,44] The deviatoric principal stretches are defined as

$$\tilde{\lambda}_A = J^{-1/3}\lambda_A \tag{7.61}$$

and used to write the scalar stored energy functions as

$$w(\tilde{\lambda}_A) = \sum_j \frac{C_j}{a_j}\left(\tilde{\lambda}_A^{a_j} - 1\right) \tag{7.62}$$

where, in *FEAP*, j can range from 1 to 3 terms. The data input for the Ogden model is given as

```
MATErial ma
  SOLId
  FINIte <VOLUme n>
  ELAStic OGDEn K C_1 a_1 C_2 a_2 C_3 a_3
                              ! blank termination record
```

7.3.6 Arruda-Boyce hyperelastic model

An isotropic hyperelastic model given by Arruda and Boyce[45] is defined by the stored energy function:

$$W(J,\bar{\mathbf{b}}) = K\,U(J) + \tfrac{1}{2}\,\mu\,[(\bar{I}_1 - 3) + \tfrac{1}{10}\,m\,(\bar{I}_1^2 - 9) + \tfrac{11}{525}\,m^2\,(\bar{I}_1^3 - 27)] \tag{7.63}$$

where $U(J)$ is one of the functions defined in (7.54), $\bar{\mathbf{b}}$ is the modified left Cauchy-Green tensor given by

$$\bar{\mathbf{b}} = J^{-2/3}\mathbf{F}\,\mathbf{F}^T$$

and $\bar{I}_1 = \mathrm{tr}\,\bar{\mathbf{b}}$ with tr denoting the trace. The bulk modulus K and shear modulus μ are computed from the elastic modulus E and Poisson ration ν in an identical way as for linear elasticity.

The input data for the model is given as

```
MATErial 1
  SOLID
  FINIte <VOLUme n>
  ELAStic ARRUda  E  nu  m
                             ! blank termination record
```

7.3.7 Yeoh hyperelastic model

An isotropic hyperelastic model given by Yeoh[46] is defined by the stored energy function:

$$W(J,\bar{\mathbf{b}}) = K\,U(J) + \tfrac{1}{2}\,\mu\,[(\bar{I}_1 - 3) + k_1\,(\bar{I}_1 - 3)^2 + k_2\,(\bar{I}_1 - 3)^3] \tag{7.64}$$

where $U(J)$ is one of the functions defined in (7.54), $\bar{\mathbf{b}}$ is the modified left Cauchy Green tensor and $\bar{I}_1 = \mathrm{tr}\,\bar{\mathbf{b}}$ with tr denoting the trace. The bulk modulus K and shear modulus μ are computed from the elastic modulus E and Poisson ration ν in an identical way as for linear elasticity.

The input data for the model is given as

```
MATErial 1
  SOLID
  FINIte <VOLUme n>
  ELAStic YEOH  E  nu  k_1  k_2
                             ! blank termination record
```

7.3.8 Logarithmic stretch model

An alternative principal stretch model is defined by strains expressed as

$$\epsilon_A = \log \lambda_A \qquad . \tag{7.65}$$

The stored energy function for this form is identical to the small strain isotropic model expressed in principal strains. Accordingly,

$$W(\lambda_A) = \tfrac{1}{2}\,\left(K - \tfrac{2}{3}G\right)\left(\sum_{A=1}^{3}\epsilon_A\right)^2 + G\sum_{A=1}^{3}\epsilon_A^2\ . \tag{7.66}$$

The stress-strain behavior for principal stresses σ_A and principal strains ϵ_A is given by

$$\left\{\begin{array}{c}\sigma_1\\ \sigma_2\\ \sigma_3\end{array}\right\} = \frac{E}{(1+\nu)(1-2\,\nu)}\left[\begin{array}{ccc}(1-\nu) & \nu & \nu\\ \nu & (1-\nu) & \nu\\ \nu & \nu & (1-\nu)\end{array}\right]\left\{\begin{array}{c}\epsilon_1\\ \epsilon_2\\ \epsilon_3\end{array}\right\}\ . \tag{7.67}$$

The transformation to the global stresses is carried out as defined in Ogden.[43]

This form of the finite strain implementations in *FEAP* is the only one which may be used in elastic-plastic analyses. It is not recommended for situations involving hyperelastic behavior at large strains. The data input for the logarithmic stretch model is given as

```
MATErial 1
  SOLId
  FINIte
  ELAStic log E nu
                              ! blank termination record
```

Note that the descriptor `log` is placed to fill the second field, it is not used explicitly by *FEAP*, indeed any word except `STVK`, `STVE`, `ORTH`, `NEOH`, `MNEO`, `MOON`, `MMOO`, `OGDE` or `CONS` may be used here. One choice is to use `ISOT` since then the `FINIte` command may be removed to test the mesh in a small deformation environment (which converges more quickly than the finite one and thus may be used to find mesh errors more easily).

7.4 Viscoelastic models

Materials which behave in a time dependent manner require extensions of the elastic models cited above. One model is given by viscoelasticity where stress may be related to strain through either differential or integral constitutive models (e.g., see *FEAP* Theory Manual). At present, the implementation in *FEAP* is restricted to isotropic

viscoelasticity in which time effects are included for the deviatoric stress components only. If we split the stress as:

$$\boldsymbol{\sigma} \;=\; \sigma_{vol}\,\mathbf{1} + \boldsymbol{\sigma}_{dev} \tag{7.68}$$

where σ_{vol} represents the spherical part given by $\frac{1}{3}\sigma_{kk}$ and $\boldsymbol{\sigma}_{dev}$ is the deviatoric stress part. Similarly the strain may be split as

$$\boldsymbol{\epsilon} \;=\; \tfrac{1}{3}\,\theta\,\mathbf{1} + \boldsymbol{\epsilon}_{dev} \tag{7.69}$$

where θ is the trace of the strain (ϵ_{kk}) and $\boldsymbol{\epsilon}_{dev}$ is the deviatoric part.

The constitutive equation may now be written as

$$\boldsymbol{\sigma}_{dev} \;=\; 2\,G\int_{-\infty}^{t} \mu(t-\tau)\,\frac{d\boldsymbol{\epsilon}_{dev}}{d\tau}\,d\tau \tag{7.70}$$

where $\mu(t)$ is a relaxation function. The term $G\,\mu(t)$ is called the relaxation modulus function. In *FEAP* the relaxation function is represented by a Prony series (in exponential terms)

$$\mu(t) \;=\; \mu_0 + \sum_i \mu_i\,\exp -t/\tau_i\;. \tag{7.71}$$

The τ_i are time parameters defining the relaxation times for the material and the μ_i are constant terms. Currently, *FEAP* limits the representation to three (3) exponential terms. The value of μ_0 is computed from

$$\mu_0 \;=\; 1 - \sum_i \mu_i\;. \tag{7.72}$$

Thus, the elastic modulus G represents the instantaneous elastic response and $G\,\mu_0$ the equilibrium, or long time, elastic modulus. Only positive μ_i are permitted and care must be taken in defining the μ_i to ensure that μ_0 is positive or zero. If μ_0 is zero the response can have steady creep and never reach an equilibrium configuration.

Input data for a one term model is given by the following data set:

```
MATErial ma
  SOLId
  ELAStic ISOTropic 30e+06 0.3
  VISCoelastic term1 0.7   10.0
                           ! blank termination record
```

Here μ_1 is 0.7 giving a μ_0 of 0.3. The relaxation time is 10 time units.

After defining the response by the above exponential representation, the constitutive equations are integrated in time by assuming the strain rate is constant over each time step. The method for integration uses exact integration over each time step and leads

to a simple recursion for each exponential term (e.g., see[47]). Additional details are also given in the *FEAP* Theory manual.

For finite deformation problems the viscoelastic parameters are related to the second Piola-Kirchhoff stress and Green strain.[48] The only finite elastic models to which viscoelasticity may be added are the Ogden form and the modified neo-Hookean model. The data set to use viscoelasticity with the modified form is given by

```
MATErial ma
  SOLId
  FINIte
  ELAStic MNEOHook E nu
  VISCoelastic term1 0.7   10.0
                           ! blank termination record
```

Replacing the elastic model with the Ogden form uses elastic data as described in Section 7.3.5. The added damage model is described by Simo[48] and is added to the model with the record

```
  DAMAge values limit rate
```

where `limit` is the fraction of damage to permit ($0 < limit < 1$) and `rate` is the rate at which damage is approached.

7.4.1 Frequency based solutions

Linear viscoelastic problems may also be formulated in a form dependent on steady state cyclic loading at a frequency ω. In this form the response quantities must be expressed in complex arithmetic, with a real response defining amplitude and an imaginary one phase change. We represent the complex stress as $\boldsymbol{\sigma}^\star$ and the complex strain as $\boldsymbol{\epsilon}^\star$ in which

$$\begin{aligned} \boldsymbol{\sigma}^\star &= \boldsymbol{\sigma}_\Re + i\,\boldsymbol{\sigma}_\Im \\ \boldsymbol{\epsilon}^\star &= \boldsymbol{\epsilon}_\Re + i\,\boldsymbol{\epsilon}_\Im \end{aligned} \tag{7.73}$$

with $i = \sqrt{-1}$. With this representation we may then write the viscoelastic material response as

$$\boldsymbol{\sigma}^\star(\omega) = \mathbb{D}^\star(\omega)\,\boldsymbol{\epsilon}^\star(\omega) \tag{7.74}$$

where $\mathbb{D}^\star(\omega)$ are frequency dependent complex moduli.

If we split the stress into volumetric and deviatoric components as

$$\begin{aligned} \boldsymbol{\sigma}^\star &= \sigma^\star_{vol}\,\mathbf{1} + \boldsymbol{\sigma}^\star_{dev} \\ \boldsymbol{\epsilon}^\star &= \theta^\star\,\mathbf{e} + \boldsymbol{\epsilon}^\star_{dev} \end{aligned} \tag{7.75}$$

and consider isotropic materials only we can write the response in terms of two complex modulus functions as

$$\sigma^\star_{vol} = K^\star\,\theta^\star \quad \text{and} \quad \boldsymbol{\sigma}^\star_{dev} = 2\,G^\star\,\boldsymbol{\epsilon}^\star_{dev} \tag{7.76}$$

where $K^\star$ and $G^\star$ are the complex bulk and shear moduli, respectively. The bulk and shear modulus functions have the real and imaginary parts

$$\begin{aligned} K^\star &= K_\Re + i\,K_\Im \\ G^\star &= G_\Re + i\,G_\Im \quad . \end{aligned} \tag{7.77}$$

In the sequel we shall assume that the volumetric response is purely elastic so that $K_\Im = 0$ at all values of ω.

In *FEAP* the viscoelastic relaxation (time)(time) function is represented by a series of exponential terms and written as

$$G(t) = G\left[\mu_0 + \sum_i^n \mu_i\,\exp -t/\tau_i\right] \quad . \tag{7.78}$$

In this form G is the elastic modulus of elasticity, τ_i are *relaxation times* and $\mu_i; i = 0, 1, , n$ are dimensionless parameters which again satisfy

$$\mu_0 + \sum_i^n \mu_i = 1 \quad \text{with} \mu_0\,\mu_i > 0 \quad . \tag{7.79}$$

The complex shear modulus for this representation has real and imaginary parts given by

$$\begin{aligned} G_\Re &= G\left[\mu_0 + \sum_i^n \mu_i \left(\frac{\omega^2\tau_i^2}{1+\omega^2\tau_i^2}\right)\right] \\ G_\Im &= G\sum_i^n \mu_i \left(\frac{\omega\tau_i}{1+\omega^2\tau_i^2}\right) \quad . \end{aligned} \tag{7.80}$$

The input form for input of the viscoelastic parameters is described in Section 7.4. When a problem form is given as:

```
*COMplex    !  Requests complex storage/solution
FEAP * * title record
  . . . .

MATErial  ...
  SOLId
    ELAStic   ISOTropic E    nu
    VISCoelastic term_i mu_i tau_i
       ....
END
```

the problem will be considered to be *frequency dependent*. In this case the solution command sequence given by

```
DT,,Domega
LOOP frequency nn
  TIME              ! omega <- omega + Domega
  TANG,,1           ! performs complex solution
   ...              ! including a FORM and SOLVe.
NEXT
```

defines the solution process for uniformly space ω steps. Changing the value in the command `DT` changes the frequency interval. Note that solutions in the frequency domain must be *linear*; thus, no iterations are required (if iteration is specified the residual should be zero for the second and any subsequent iterations). Note that omission of the `*COMplex` statement *before* the `FEAP` start record will result in the program performing all operations in *real arithmetic*.

Remark 1: Currently, only the *solid, displacement model* elements can treat complex materials.

Remark 2: Omission of the viscoelastic terms results in a material with all imaginary moduli set to zero. Linear elastic and viscoelastic materials may be used in the same analysis.

7.5 Plasticity models

Classical elasto-plastic material models are included in *FEAP* for small and finite deformation problems. One finite deformation model is based on logarithmic principal stretches and product split of the deformation gradient. This leads to a form which is similar to that for small strains, however, the finite deformation form does not allow use of kinematic hardening.[1] Accordingly, here we limit our discussion to the small strain problem.

The stress for an elasto-plastic material may be computed by assuming an additive split of the strain as

$$\epsilon = \epsilon^{el} + \epsilon^{pl} \ . \tag{7.81}$$

For most models, an associative flow rule is assumed so that the plastic strain rate may be computed from a gradient of the *yield function*, F, as

$$\dot{\epsilon}^{pl} = \dot{\gamma} \, \frac{\partial F}{\partial \sigma} \ . \tag{7.82}$$

[1] For cases requiring use of kinematic hardening one may use the orthotropic `HILL` yield model set for isotropy.

The relation is integrated in time using a backward Euler (implicit) time integration to compute a discrete form of the problem.[2,41]

Isotropic and kinematic hardening are also added to the model. The kinematic hardening may be given by a linear form where it is assumed that

$$\alpha = H_{kin}\,\epsilon^{pl} \tag{7.83}$$

where α is the back stress and H_{kin} is the kinematic hardening modulus. The isotropic hardening is taken in a linear and saturation form as

$$Y(e^{pl}) = Y_\infty + (Y_0 - Y_\infty)\exp(-\beta\, e^{pl}) + H_{iso}\, e^{pl} \tag{7.84}$$

where Y_0 is the initial uniaxial yield stress, Y_∞ a stress at large values of strain, β a delay constant, and H_{iso} is a linear isotropic hardening modulus. The accumulated plastic strain is computed from

$$e^{pl} = \int_0^t \dot{\gamma}\, d\tau \ . \tag{7.85}$$

In *FEAP* the discrete problem is solved using a closest point return map algorithm (e.g., see[2,41,49,50]).

7.5.1 Isotropic plasticity

Input properties for the `J_2` (`MISEs`) model are given by linear isotropic hardening is given by:

```
MATErial ma
  SOLId
  ELAStic ISOTropic  E      nu
  PLAStic MISEs      Y_0   Y_inf beta
  PLAStic HARDening  H_iso <H_kin>
                           ! blank termination record
```

N.B. In isotropic plasticity `H_kin` is allowed in small deformation only. The orthotropic model described below allows for both isotropic and kinematic hardening.

By default, the first iteration of each step for an inelastic solution is always assumed to be *elastic*. For solutions computed either by a static or an implicit dynamic solution this should be used. However, for any explicit solution method of an inelastic material this will result in a purely elastic solution and the statement

```
STARt INELastic
```

must be added to the material data. The command may be used for other cases if no unloading is expected, however, if significant unloading does occur use of the inelastic option generally results in a failure the Newton iteration to converge.

In the finite deformation form it is also possible to use additional types of yield functions. A Drucker-Prager yield form is input using the command

```
PLAStic DRUCker sig_t  sig_c
```

where `sig_t` and `sig_c` are the uniaxial yield stress in tension and compression. The standard parameters are computed from these. This yield type should be used for cases where limited pure hydrostatic tension exist as no multi-surface cap parameter is included. Finally, a third yield function based on the Prager-Lode form is available with the input parameters given as

```
PLAStic LODE sig_t  sig_c
```

These two yield function forms should be used with great caution.

7.5.2 Orthotropic plasticity

The orthotropic plasticity behavior is based on the Hill model[51] in which the yield function is specified as:

$$f = \left[F\,(\sigma_{22}-\sigma_{33})^2 + G\,(\sigma_{33}-\sigma_{11})^2 + H\,(\sigma_{11}-\sigma_{22})^2 \right.$$
$$\left. +2\,L\,\sigma_{23}^2 + 2\,M\,\sigma_{31}^2 + 2\,N\,\sigma_{12}^2\right]^{1/2} - \sigma_Y(e^p)$$

where F, G, H, L, M, N control the ratio of yield in different directions. In *FEAP* the parameters are defined from uniaxial stress states as

$$Y_{ij} = R_{ij}\,\sigma_Y(e^p)$$

in which, for example Y_{11} is the yield value for a uniaxial test using only σ_{11}. In this form σ_Y may be defined as one of the uniaxial yield values and the corresponding R-value is set to unity.

The relation between the Hill parameters and the ratios is given by

$$G+H = \frac{1}{R_{11}^2}\ , \quad 2F = \frac{1}{R_{22}^2} + \frac{1}{R_{33}^2} - \frac{1}{R_{11}^2}$$
$$H+F = \frac{1}{R_{22}^2}\ , \quad 2G = \frac{1}{R_{33}^2} + \frac{1}{R_{11}^2} - \frac{1}{R_{22}^2}$$
$$F+G = \frac{1}{R_{33}^2}\ , \quad 2H = \frac{1}{R_{11}^2} + \frac{1}{R_{22}^2} - \frac{1}{R_{33}^2}$$

and

$$2L = \frac{1}{R_{23}^2}\ , \quad 2M = \frac{1}{R_{31}^2} \ \text{ and } \ 2N = \frac{1}{R_{12}^2}\,.$$

Note that the definitions for F, G, H, L, N and M here are scaled by σ_Y^2.

The *FEAP* input data is given as:

```
MATE ma
  SOLID
    ELAStic <ISOT, ORTH>  ..... ! add correct data
    PLAStic HILL R_11 R_22 R_33 R_12 R_23 R_31
    PLAStic <YIELd,SWIFt> ..... ! add correct data
      ... Some other data can also be added
```

A Hill model may also be used to represent isotropic behavior by using the form

```
PLAStic HILL 1.0 1.0 1.0 1/sqrt(3) 1/sqrt(3) 1/sqrt(3)
```

The behavior for the yield may be given in a saturation form:

$$\sigma_Y = Y_0 + (Y_\infty - Y_0)(1 - \exp \beta e^p) + H_{iso}\, e^p$$

with input specified as

```
\vspace{-12pt}
PLAStic YIELd Y_0 Y_inf beta H_iso
```

Alternatively, in a Swift power law form as

$$\sigma_Y = K\,(e^p + e_0^p)^n + H_{iso}\, e^p$$

with input specified as[2]

```
PLAStic SWIFt K ep_0 n H_iso
```

Note the initial yield for a swift behavior is given by $\sigma_Y(0) = K\,(e_0^p)^n$. Kinematic hardening is specified using the command

```
PLAStic KINEmatic H_2 H_3 H_4 H_5 H_6 J_1
```

(see paper by Papadopoulos & Lu[52] for details on model). For an isotropic material all the `H_i` are set to the *kinematic hardening modulus* H_{kin} and `J_1` is set to zero.

Orthotropic plasticity axes coincide with the principal elastic axes. If the axes differ from the material coordinate axes a `VECTor ORTHotropic` record should be specified as described in Section 7.3.2.

[2]The determination of the `SWIFt` parameters from test data is given in ASTM Standard E646.

7.6 Generalized plasticity models

For both small and finite strain problems, *FEAP* includes a *generalized plasticity formulation*.[2,53,54] The generalized plasticity formulation has advantages when cyclic loading is involved. In this case the model will retain a smooth transition from an elastic behavior to a plastic flow condition. The input data for the model is given as

```
MATErial ma
  SOLId
  ELAStic ISOTropic   E      nu
  PLAStic GENEralized Y_0   Y_inf beta
  PLAStic HARDening   H_iso <H_kin>
                            ! blank termination record
```

where now β denotes the speed at which the transition takes place (see references for more details). The finite deformation formulation uses a principal stretch formulation. Consequently, kinematic hardening is included only in the small strain version.

7.7 Fiber models

Axial fibers may be added to any material model. The fiber model is elastic and based on the stored energy function

$$W(\mathbf{C}) = \psi_f(I_4) \quad \text{where} \quad I_4 = A_I C_{IJ} A_J$$

The A_I are material structure values that describe unit vectors in the reference configuration that are oriented in the fiber direction.

7.7.1 Holzapfel-Gasser model

Currently, one form for the function $\psi_f(I_4)$ is of the type

$$\psi_f(I_4) = \frac{h_1}{2h_2}\left[\exp h_2(I_4 - 1)^2 - 1\right]$$

which is taken from reference 55 and is similar to that proposed by Holzapfel, Gasser and Stadler.[56,57] In this model h_1 has dimensions of moduli and h_2 is dimensionless. In the current configuration this functional gives stresses expressed by

$$\sigma_{ij} = \frac{1}{J} a_i a_j \frac{\partial \psi_f}{\partial I_4} \quad \text{where} \quad a_i = F_{iI} A_I$$

and thus are directed along the deformed fiber.

The data for this fiber model is added to any material model as (using a solid element as an example):

```
MATErial ma
  SOLId <fiber ma2>
  FINIte
  ELAStic HOLZapfel  h_1 h_2 A_1 A_2 A_3 ! First  fiber
  ELAStic HOLZapfel  h_1 h_2 A_1 A_2 A_3 ! Second fiber
                              ! blank termination record
```

up to 3 different orientations may be specified for any model. Multiple instances of the model can be given for more than 3 fibers. By adding the optional `ma2` the fiber will be an overlay on the *ma2* element set number instead of the set with number `ma1`. This permits getting stresses from the individual constituents. The above could be split into two instances if the stresses in individual fibers are wanted.

The fiber model may also be given as the sum of the two constituents composed of a continuum elastic (or inelastic) response using the data set

```
MATErial ma
  SOLId
  ELAStic <NEOH, MEOH, ....>   ....
  FIBER   HOLZapfel h_1 h_2 A_1 A_2 A_3
  FIBER   HOLZapfel h_1 h_2 A_1 A_2 A_3
  <PLAStic, VISCoelastic, etc.>
  <FINITe>
                              ! blank termination record
```

Note in this form the addition of the `FINIte` record may be optional if the elastic model is defined only in finite deformation. The only advantage of the above form is that the element routine is only called once, whereas used as an overlay, the element routine is called multiple times – once for each associated material set. A disadvantage is that all stresses from each constituent are combined, and thus are averaged values.

7.7.2 Weiss model

A second form for the function $\psi_f(I_4)$ is that proposed by Weiss[58] and is expressed as

$$\psi_f(I_4) = \frac{w_1}{w_2}\left[\exp w_2(I_4 - 1) - I_4^{w_2}\right]$$

The data for this fiber model is added to any material model as (using a solid element as an example):

```
MATErial ma
  SOLId <fiber ma2>
  FINIte
  ELAStic WEISs  w_1 w_2 A_1 A_2 A_3 ! First  fiber
  ELAStic WEISs  w_1 w_2 A_1 A_2 A_3 ! Second fiber
                             ! blank termination record
```

up to 3 different orientations may be specified for any model. Multiple instances of the model can be given for more than 3 fibers.

Again, the composite result that sums the two constituents with the continuum elastic response is given as

```
MATErial ma
  SOLId
  ELAStic <NEOH, MEOH, ....>   ....
  <PLAStic, VISCoelastic, etc.>
  <FINITe>
  FIBER   WEISs  w_1 w_2 A_1 A_2 A_3 ! First  fiber
  FIBER   WEISs  w_1 w_2 A_1 A_2 A_3 ! Second fiber
                             ! blank termination record
```

with a maximum of 3 `FIBEr` records.

For cases where the `A_i` vary in each element their values may be specified using the mesh command `STRUcture` (see Appendix A, section on structure vectors [STRU:A] for details). This input, however, is restricted to cases where one or two fibers exist in an element.

7.8 Acoustic fluid model

The only model available for acoustic fluid elements is a linear elastic behavior. The basic parameter to be specified is the acoustic wave speed, c. This is related to the elastic volumetric (bulk) modulus, K and the mass density, ρ, by the relation

$$c = \sqrt{\frac{K}{\rho}}$$

The element permits the alternative inputs of

```
MATErial ma
  ACOUstic
    FLUId velocity c
```

or

```
MATErial ma
  ACOUstic
    FLUId
    BULK modulus K
    DENSity mass rho
```

7.9 Lysmer-Kuhlmeyer absorbing boundary condition

For transient analysis of finite element models that truncate a very large (or infinite) region, it is necessary to use an absorbing boundary condition to prevent artificial reflection of arriving wave responses. A simple viscous model based on the work of Lysmer & Kuhlmeyer[59] is provided for 1, 2, and 3 dimensional models. The basic input of the material set is given by

```
MATErial ma
  LABC ! Lysmer Absorbing Boundary Condition
    ELAStic modulus E_value nu_value
    DENSity mass    rho_value
```

Additional optional parameters are:

```
    AREA    factor  area
    FACTor  p_wave s wave
    PLANe  <STREss,STRAin>
    AXISymetric
    QUADr  <NODAl,GAUSs>
```

and if not provided factors are set to unity, `PLANe STRAin` (same as general 3-D), and `NODAl`. This model may be used with `SOLId` and `ACOUstic` element types.

7.10 Robin material models

Two and three dimensional models may be supported by an elastic and/or viscous line or surface boundary. The model is similar to the one defined by Moireau *et al.*[60] The model is specified using the commands

```
MATErial ma
  ROBIn ! Robin Boundary Condition
    ELAStic modulus k_value
    VISCous modulus c_value
   <SURFace,LINE>             ! Default SURFace
```

```
 QUADrature value n_1 n_2  ! n_2 not required for LINE
 PLOT <ON OFF>             ! Default OFF
<NURB quad q_1 q_2>        ! For IgA analyses only
                           !     Omit QUAD specification
```

One may specify either `ELAStic` or `VISCous` or both. With `PLOT` specified as an `OFF` value the mesh is not displayed using the command `PLOT MESH`.

The Robin boundary condition restrains the nodes in the associated element to be restrained by an forces resulting from the boundary traction

$$\mathbf{t} = k\,\mathbf{u} + c\,\mathbf{v}$$

Since the parameters k and c are constant, the response is linear and does not account for changes in geometry.

7.11 Winkler material models

Two and three dimensional models may be supported by an elastic surface boundary in the normal to the reference geometry. The model is specified using the commands

```
MATErial ma
  WINKler ! Winkler Boundary Condition
    ELAStic modulus k_value
    QUADrature value n_1 n_2  ! n_2 not required for LINE
    PLOT <ON OFF>             ! Default OFF
   <NURB quad q_1 q_2>        ! For IgA analyses only
                              !     Omit QUAD specification
```

With `PLOT` specified as an `OFF` value the mesh is not displayed using the command `PLOT MESH`.

The Winkler boundary condition restrains the nodes in the associated element to be restrained by an forces normal to the boundary. Thus, the normal traction

$$t_n = k\,\mathbf{n}^T\mathbf{u}$$

where $\mathbf{n}$ is the normal to the reference geometry. Since the parameters k and $\mathbf{n}$ are constant, the response is linear and does not account for changes in geometry.

7.12 Interface material models

The physical behavior for interface models is defined using the basic form

```
MATErial ma
  INTErface                ! Interface clase of elements
    TYPE <1,2,3,4,5>       ! Particular interface model
      data for the model   ! Users are required to program
```

In general, users are required to program the solution steps in an element module defined by

```
subroutine intf0n(d,ul,xl,ix,tl,s,p,ndf,ndm,nst,isw)
```

where `n` is replaced by a 1 to 5.

A simple linear interface model is provided in `inf01` in which the interface model is restricted to one (1) degree of freedom (`dof`) with the remaining ones either linked or free. The model for the degree of freedom is linear with

$$q_{dof} = k \left[u^{(2)}_{dof} - u^{(1)}_{dof} \right]$$

The material data set then consists of the commands

```
MATErial ma
  INTErface                ! Interface clase of elements
    TYPE 1
      FACTor,,k            ! model: f = k * u
      DOF   ,,dof          ! degree of freedom for 'u'
      PLOT <ON, OFF>       ! plotting of element (optional)
```

N.B. This interface element cannot model finite deformation effects in which upon loading the node pairs separate by large amounts.

7.13 User material models

Additional material models may be added to the program by coding two subprograms: `UMATIn` and `UMATLn` where `n` ranges from 0 to 9 (see the *FEAP* Programmer Manual for details on writing the material model routines.[12]). Each user model is defined by a four character alphanumeric name and the data command to access the material model is given as:

```
UCON xxxx <vv(1:5)>
```

where `xxxx` is the unique 4 character name and `v(1:5)` are an optional array of up to 5 values which may be used to define material parameters. Additional parameters may be input within the `UMATIn` subprogram.

Generally, standard material commands (e.g., element type, mass density, etc.) should precede the `UCON` command.

Users should be cautious on ordering of stresses used in converting tensor components of stress to Voigt matrix form. All *FEAP* elements assume stresses are order as

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{22} & \sigma_{33} & \sigma_{12} & \sigma_{23} & \sigma_{31} \end{bmatrix}$$

If other ordering is used to code the `UMATLn` module a reordering may be provided by inserting the command

```
VOIGt order n1 n2 n3 n4 n5 n6
```

in the material set before the input of the user material property values using the

```
UCON  .....
   or
UMAT
   or
FCON
```

command. If an Abaqus `umat` is adapted to *FEAP* the reordering may be set using

```
VOIGt ABAQus
```

7.14 Mass matrix type specification

The mass matrix for continuum problems and the specific heat matrix for thermal problems may be either a *consistent*, *lumped*, or *interpolated* form. By default *FEAP* uses a lumped matrix. If $\mathbf{M}_{cons}$ is the consistent matrix and $\mathbf{M}_{lump}$ is the diagonal lumped matrix, the interpolated matrix is defined as:

$$\mathbf{M}_{interp} \;=\; (1 \,-\, a)\, \mathbf{M}_{cons} \;+\; a\, \mathbf{M}_{lump} \;. \tag{7.86}$$

The type of mass and, where required, the parameter a are input using the `MASS` command as shown in Table 7.7. Non-zero values for r and i add rotational lumped mass and, for frame elements, rotatory inertia effects, respectively.

Command	Type	Parameters
MASS	LUMPed/DIAGonal	$0, r$
MASS	CONSistent	$0, 0, i$
MASS	OFF	
MASS		a, r, i

Table 7.7: Material Model Mass Related Inputs

7.15 Rayleigh damping

The effects of damping may be included in transient solutions assuming a damping matrix in the form

$$\mathbf{C} \;=\; a_0\,\mathbf{M} \;+\; a_1\,\mathbf{K}\;. \tag{7.87}$$

This defines a form called *Rayleigh Damping*. The input for this form of damping is given by:

```
MATErial ma
  .....
    DAMPing RAYLeigh a0 a1
```

This command is only included for small deformation elements using a linear elastic material model and is used only for time dependent solutions specified by a `TRANsient` solution command. Rayleigh damping may also be defined for modal solutions (Section 14.5.1).

7.16 Element cross section and load specification

7.16.1 Resultant formulations

The plane stress and structural elements require specification of cross-section information. For the plane stress, plate, and shell elements the cross-section is a thickness which is specified using the **THICkness** command as shown in Table 7.9. The plate element also permits the effects of transverse shear deformation to be included and, if this is different than the 5/6 default value it is also given using the thickness command. The linear kinematics shell permits the stiffness values to be computed by integration of plane stress values through the thickness using q_t quadrature points. Inelastic materials are also permitted.

For frame elements the orientation of the cross-section axes is described by an orthonormal triad $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_3$ as described in Sect. 5.11. The local coordinates for the frame are described by

$$\mathbf{x} = x\,\mathbf{t}_1 + y\,\mathbf{t}_2 + z\,\mathbf{t}_3$$

with the origin locate at Node 1 of the two node element. Thus, the area and inertia of the cross section are described in terms of the y and z coordinates and may be input as indicated in Table 7.9.

For truss elements it is necessary to provide cross-sectional property for area as indicated in Table 7.9.

7.16.2 Section integration formulations

Structural element behavior may also be defined by numerical integration over the cross section using the `SECTion` command.

Two dimensional frame sections

For the two-dimensional frame elements the cross section is defined by a set of two or more *layer* commands. The form for each command is

```
SECTion LAYEr z-coord width n-quad
```

where `z-coord` is the coordinate in the depth direction and `width` is the width of the section at that location. A physical layer thickness is defined by the distance between two layer commands. The layers must be ordered from the bottom (the most negative `z-coord`) to the top (the most positive `z-coord`). For the cross-section shown in Fig. 7.1 the input data is given by

```
SECTion LAYEr z_1 w_1 q
SECTion LAYEr z_2 w_1 q
SECTion LAYEr z_2 w_2 q
SECTion LAYEr z_3 w_2 q
SECTion LAYEr z_3 w_1 q
SECTion LAYEr z_4 w_1 q
```

for a two-dimensional cross section.

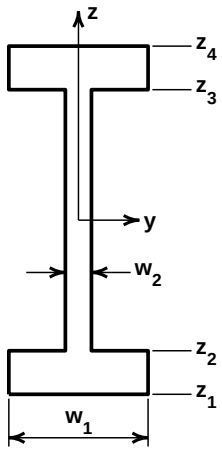
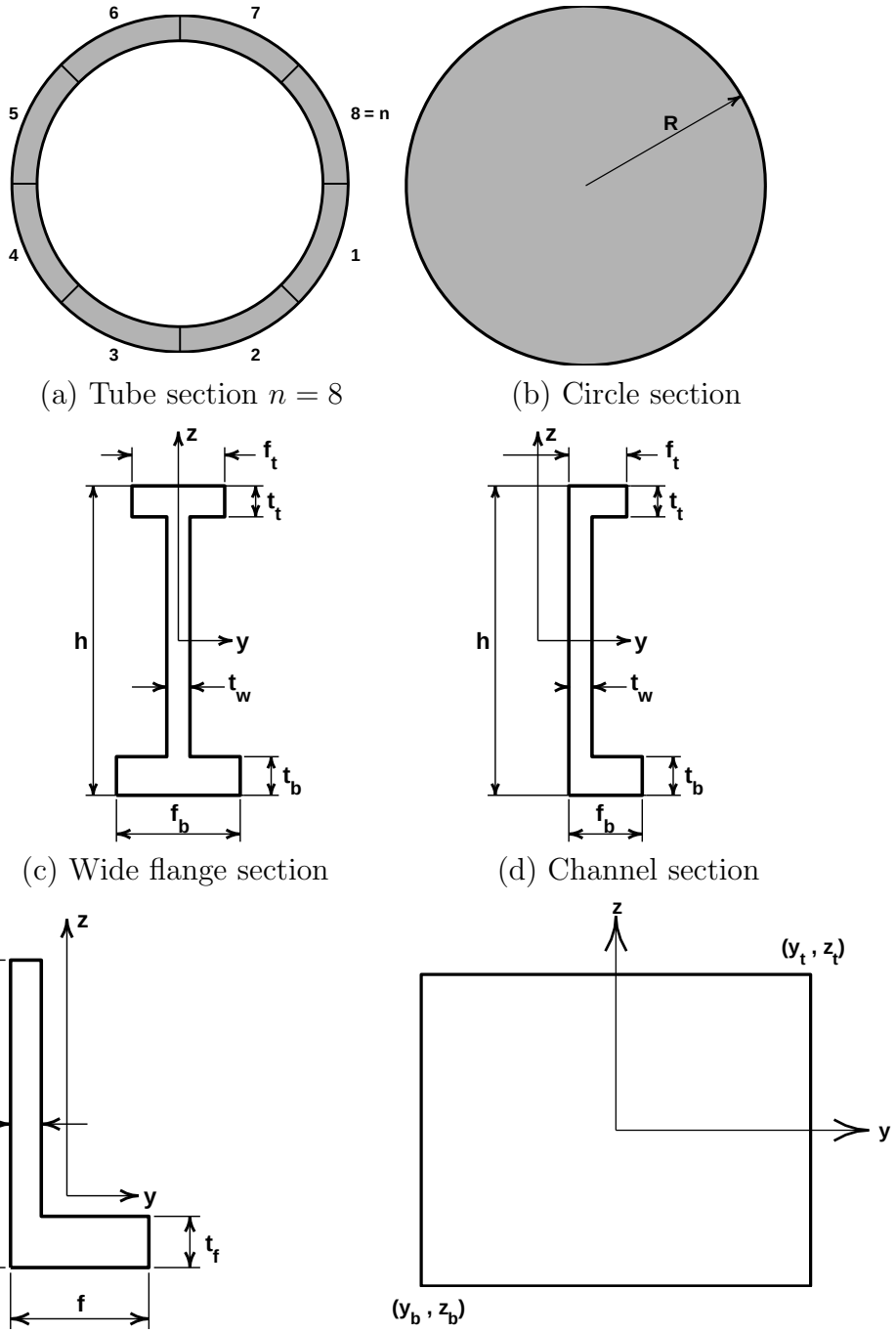


Figure 7.1: Cross-section for frame element



(a) Tube section $n = 8$ (b) Circle section

(c) Wide flange section (d) Channel section

(e) Angle section (f) Rectangle section

Figure 7.2: Cross-sections types for 3-dimensional frame analysis

Three dimensional frame sections

For three-dimensional truss and frame elements the cross section may be defined by alternate forms which include: `TUBE`, a thin circular tube; `RECT`angle, a rectangular solid section; `WIDE` flange, a wide flange composite section; `CHAN`nel, a channel composite section; `ANGL`e, an angle composite section; and `CIRC`le, a solid circular section as shown in Fig. 7.2. The basic form of a section command is:

```
SECTion TYPE (EV(i),i=1,6)
```

The data parameters `EV` for each type are summarized in Table 7.8 where all the parameters except the quadrature order are shown in Fig. 7.2.

TYPE	EV(1)	EV(2)	EV(3)	EV(4)	EV(5)	EV(6)	
TUBE	r	t	n	q_n			
CIRCle	r	q					
RECTangle	y_b	z_b	y_t	z_t	q_y	q_z	
WIDE flange	h	f_t	f_b	t_t	t_b	t_w	
CHANnel	h	f_t	f_b	t_t	t_b	t_w	
ANGLe	h	f	t_h	t_f			

Table 7.8: Types and data for integrated cross-sections.

In Table 7.8 r denotes radius, t thickness, h height, f flange width, t top, b bottom, q quadrature order, and n number of segments. The cross section is assumed to lie in a y-z plane.

The cross-section shown in Fig. 7.1 can be input for a three-dimensional frame element using the command set:

```
SECTion RECTangular -w_1/2 z_1 w_1/2 z_2 q_y q_z
SECTion RECTangular -w_2/2 z_2 w_2/2 z_3 q_y q_z
SECTion RECTangular -w_1/2 z_3 w_1/2 z_4 q_y q_z
```

where it is noted the y-coordinate locations require dividing the width by 2. Alternatively, this cross-section may be input using a wide-flange type section as:

```
SECTion WIDE (z_4 - z_1) w_1 w_1 (z_4 - z_3) (z_2 - z_1) w_2
```

where quadrature order is now selected by defaults within *FEAP*.

7.16.3 Loading on structural elements

Element loads for surface pressure and body force are input using the `LOAD`, `BODY` or `GROUp` force commands as shown in Table 7.9.

Command	Type	Parameters
THICkness		$h,\ \kappa,\ q_t$
CROSS	section	$A,\ I_{yy},\ I_{zz},\ I_{yz},\ J_{xx},\ \kappa_y,\ \kappa_z$
BODY	forces	$b_1,\ b_2,\ b_3$
GROUp	forces	$g_1, p_1,\ g_2, p_2,\ g_3, p_3$
LOAD	normal	q

Table 7.9: Cross Section and Body Force Inputs

Command	Solid	Truss	Frame	Plate	Shell	Membrane	Thermal
THICkness	X	-	-	X	X	X	X
CROSs	-	X	X	-	-	-	-
BODY	X	X	X	-	X	X	-
GROUp	X	X	X	-	X	X	-
LOAD	-	-	-	X	X	X	-

Table 7.10: Geometry and Loads vs. Element Types

The types of elements affected by the `THICkness`, `LOAD` and `BODY` commands is indicated in Table 7.10.

A typical input for `BODY` forces is

```
MATErial ma
  ...
 BODY forces b_1 b_2 b_3
```

where the body forces b_i are directed in the i-coordinate direction and, when proportional loads are present, are multiplied by the *total* proportional loading factor. An alternative which includes the ability to specify a body force g_i multiplied by a specified proportional load number p_i is given by

```
MATErial ma
  ...
 GROUp forces g_1 p_1 g_2 p_1 g_3 p_3
```

Finally, if both are given as

```
MATErial ma
  ...
 BODY forces b_1 b_2 b_3
 GROUp forces g_1 p_1 g_2 p_1 g_3 p_3
```

The total body force is computed as

$$f_i = b_i + g_i\, p_i(t)$$

where $pr_i(t)$ denotes the proportional load value for p_i at time t and is specified during the solution phase using the `PROP` solution command.

Note that use of `GROUp` forces to represent the effects of an earthquake ground motion require the product of g_i and p_i to be proportional to density times acceleration.

7.17 Miscellaneous material set parameter specifications

In addition to the above material, geometric and loading parameters the values for some other variables may also be set.

It is possible to replace global parameters for the type of two dimensional analysis using the `PLANe STREss`, `PLANe STRAin`, or `AXISymmetric` commands. Similarly the global value for the temperature degree of freedom to use in coupled thermo-mechanical problems may be changed for the current material set using the `TEMPerature` command. The formats are indicated in Table 7.11 and the affected element types in Table 7.12. The values for the number of quadrature points (in elements, not cross sections) to be used for computing arrays and element outputs may be set using the `QUADrature` command. Generally, *FEAP* will select an appropriate order of quadrature to be used in computing the arrays and for output of element quantities. Thus, care should be used in changing the default values.

For transient solutions, it is useful on occasion to solve a part of the problems by an *explicit* method while other parts are solved by an *implicit* method. (See Sect. 14.1.38 for time integration options). If a transient solution is requested in the solution commands, the default type for each material set is *implicit*. If it is desired to have the set of elements associated with a particular material set to be solved by an *explicit* form of the specified method the command

```
TRANsient EXPLicit
```

must be included. If desired, it is possible to include the command

```
TRANsient IMPLicit
```

in other material sets for clarity.

A more efficient implementation is also available using the method described in Sect. 14.4.5 in which individual elements are tagged as being either in the explicit or implicit group.

FEAP includes capabilities to solve finite deformation problems using the `SOLId`, `FRAMe`, `TRUSs`, `SHELl`, `MEMBrane` and `GAP` elements. To select the finite deformation element it is necessary to use the `FINIte` deformation option instead of the default `SMALl`

deformation option. This may be done for all materials using the `GLOBal` command. There are three different element technologies which may be selected `DISPlacement` (which is the default), `MIXEd`, or `ENHAnced strain` types. The data options for these are indicated in Table 7.11 and the affected element types in Table 7.12.

Command	Type	Parameters
QUADrature PENAlty		n_{array}, n_{output} k_{pen}
ADAPtive	ERROr	η
TEMPerature		T_{dof}
SMALl FINIte NONLinear	deformation deformation	
DISPlacment MIXEd ENHAnced	 strain	
PLANe PLANe AXISymmetric	STREss STRAin	
TRANsient TRANsient	IMPLicit EXPLicit	Implicit time integrator Explicit time integrator

Table 7.11: Miscellaneous Material Model Inputs

Command	Solid	Truss	Frame	Plate	Shell	Membrane	Thermal
QUADrature	X	-	-	-	X	X	X
PENAlty	-	-	-	-	-	-	-
ADAPtive ERRor	X	-	-	-	-	-	-
TEMPerature	X	X	X	X	X	-	-
SMALl	X	X	X	-	X	X	-
FINIte	X	X	X	-	X	X	-
NONLinear	-	X	X	-	-	-	-
DISPlacement	X	-	-	-	-	-	-
MIXEd	X	-	-	-	-	-	-
ENHAnced	X	-	-	-	-	-	-
PLANe STREss	X	-	-	-	-	-	X
PLANe STRAin	X	-	-	-	-	-	X
AXISymmetric	X	-	-	-	-	-	X

Table 7.12: Miscellaneous Material Commands vs. Element Types

Chapter 8

Nodal mass, dampers and springs

FEAP has options to add discrete mass, damping, and stiffness terms to a single node in the problem.

8.1 Nodal mass

Mass may be added at a node as *lumped* terms at each degree of freedom. The data for discrete masses are included as input in the form

```
MASS
  m,mg,M1_m,M2_m,M3_m ... Mndf_m
  n,ng,M1_n,M2_n,M3_n ... Mndf_n
                           ! blank termination record
```

where `m, n` are node numbers, `mg, ng` are generation increments to nodes, and `Mi_m, Mi_n` are discrete mass values. Generation of missing nodes will take place if the `mg` value is non-zero. Mass values will be interpolated linearly for the *i-th* degree of freedom.

8.2 Nodal dampers

Damping values also may be specified for any node. Each linear damper is fixed at one end and attached to a degree of freedom at the other. Damping values are input as

```
DAMPer
  m,mg,C1_m,C2_m,C3_m ... Cndf_m
  n,ng,C1_n,C2_n,C3_n ... Cndf_n
                           ! blank termination record
```

where `Ci_m, Ci_n` are discrete damper values for the *i-th* degree of freedom.

8.3 Nodal stiffness: Springs

Finally, linear stiffness (springs) may be attached to any node. Each linear spring is fixed at one end and attached to a degree of freedom at the other. Stiffness values are input as

```
STIFness
  m,mg,K1_m,K2_m,K3_m ... Kndf_m
  n,ng,K1_n,K2_n,K3_n ... Kndf_n
                           ! blank termination record
```

where `Ki_m, Ki_n` are discrete stiffness values for the *i-th* degree of freedom.

Chapter 9

Include and looping: Data reuse

Often in constructing a model it is possible to replicate one part to produce a new part of the mesh. *FEAP* provides several options to facilitate such reuse. The basic method is to place the part of the problem to be reused in a separate file, called an include file, and to input the data by adding a statement `INCLude filename` where the data is to be inserted. This feature is described in the next section. A second option is to mark the data using a `SAVE` command and to `READ` the data where it is again needed. This is described in Section 9.2. Finally, it is possible to reread the data parts several times using a `LOOP-NEXT` option as described in Section 9.3.

9.1 Include commands in mesh input

Any set of data input records may be placed in a separate file and read using the `INCLude` command. The form for an include is a single record

```
INCLude filename
```

where `filename` is the name of the file containing the input data items. This command may be used at any time and include files may call other include files (to a maximum level of 9). Thus, if the nodal coordinates are created by another program and written to a file named `Blockxy` [1], they may be input as *FEAP* data using:

```
COORdinates
INCLude Blockxy
                    ! blank termination record
```

The information in each file must always be in the format required by *FEAP*. If another format is written, then it is necessary to either translate the data to the correct form

[1] Upper and lower case letters are different in UNIX or LINUX environment but the same in a Windows one

or to write and link a user routine which can input the data. The creation of user routines is discussed in the *FEAP Programmers Manual*.[12]

9.2 READ and SAVE commands in mesh input

A group of mesh input statements also may be retained for future use by placing them between the statements

```
SAVE,filename
.....
.....
SAVE,END
```

`filename` may be any 1 to 14 alphanumeric characters. Thus if a `SAVE MSH1` is used a new file named `MSH1` will be created to store the mesh commands to be saved.

For example, the following option may be used to generate nodal forces with a variation in a load parameter.

```
PARAmeter
  a= 5.
                ! end with blank record
SAVE,msh1       ! may also be SAVE mes1
PARAmeter
  b= a/2
                ! end with blank record
FORCe
  31,0,b
  32,1,a
  34,0,a
  35,0,b
                ! end with blank record
SAVE,END
```

A different loading state may then be specified by:

```
PARAmeter
  a= -4.
                ! terminator
READ,msh1
```

The value of b will be recomputed using the new value of a and the nodal forces will then be recomputed. Many options are possible using the features of parameters, expressions, INCLude, and SAVE and READ commands.

9.3 LOOP-NEXT to replicate mesh parts

Many models for problems analyzed by finite element methods have mesh parts which are similar except for stretching and rotation transformations. *FEAP* provides input capabilities to generate the model using `LOOP-NEXT` commands. The basic input structure is given by the command sequence

```
LOOP,n
  ...
NEXT
```

where n defines the number of times to repeat the commands contained within the loop. The value of n may be a constant or a parameter. Any standard *FEAP* mesh commands may be used between the `LOOP` and `NEXT` statements, however, it is easiest to use commands which do not require explicit definitions for node or element numbers.

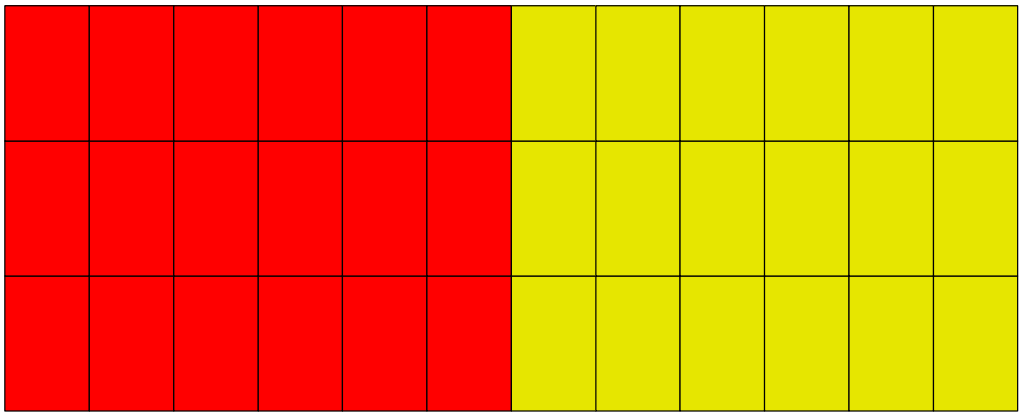


Figure 9.1: Two blocks using `LOOP-NEXT` commands

A simple example is the repetition of two blocks of identical elements in which the material number is different. Assume first that a file named `Imblock` is constructed which contains the commands

```
BLOCk
  CART n1 n2 0 0 ma
    1   0  0
    2   a  0
    3   a  b
    4   0  b

PARAMeter
   ma = ma + 1
```

Then a second file is given which defines the initial values of parameters and the looping control. This file is given by the statements shown in Table 9.1 where we note the use of the loop using the TRANsform command. The above example produces the mesh

```
FEAP * * Two block problem
  0  0  0  2  2  4
PARAmeters
  a  = 5
  b  = 4
  n1 = 6
  n2 = 3
  ma = 1

LOOP,2
  INCLude Imblock
  TRANsform
    1 0 0
    0 1 0
    0 0 1
    a 0 0
NEXT
MATE 1
  SOLID
    ELAStic ISOTropic 1000 0.25

MATE 2
  SOLID
    ELAStic ISOTropic 2000 0.25

END
```

Table 9.1: LOOP-NEXT mesh construction

shown in Fig. 9.1 and is trivial (also not much is gained over a construction using two block commands directly).

A more involved example is shown in Fig. 9.2 for a disk containing circular holes. This example was constructed using the commands shown in Table 9.2. The file Iwseg contains the mesh for one part of the repeating mesh as shown in Fig. 9.3.

Many more involved mesh constructs may be considered using the LOOP-NEXT commands. When using this option with blending functions, however, *do not* place SNODe or SIDE commands within any looping instructions. In this case a correct structure is:

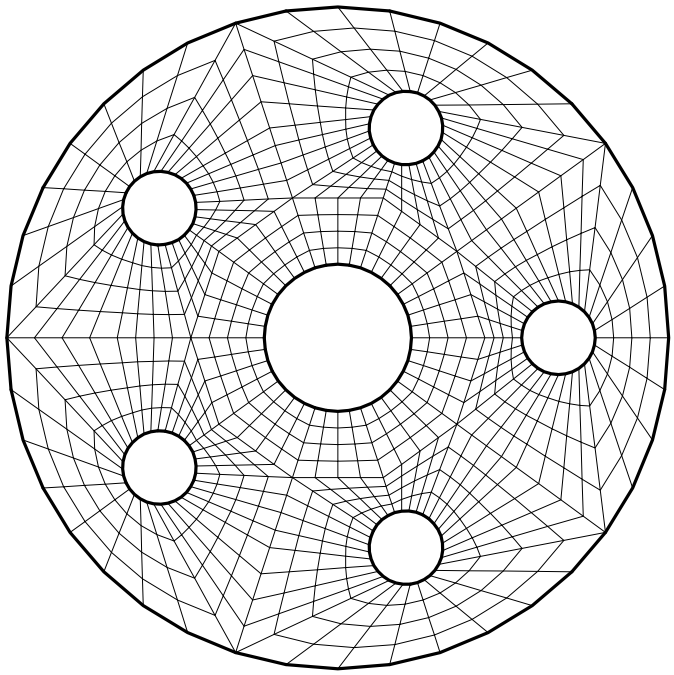


Figure 9.2: Disk with holes

```
LOOP 5
  TRANSform
    cosd(th)  sind(th)    0
   -sind(th)  cosd(th)    0
        0         0       1
        0         0       0
  INCLude Iwseg
  TRANSform
    cosd(th)  sind(th)    0
    sind(th) -cosd(th)    0
        0         0       1
        0         0       0
  INCLude Iwseg
  PARAmeter
    th = th + 72

NEXT
```

Table 9.2: `LOOP-NEXT` disk mesh construction

```
SNODe
  1   ...
```

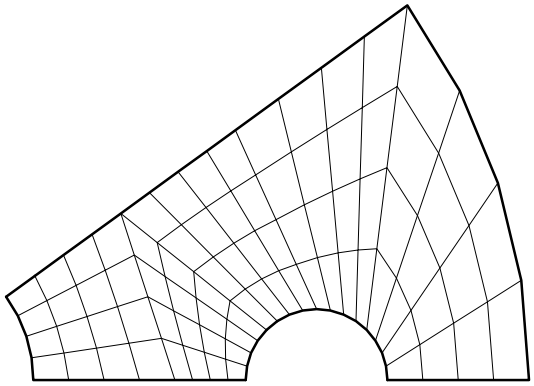


Figure 9.3: Mesh segment for disk with holes

```
      etc.

SIDE
  POLAr   ... (or other)
      etc.

LOOP,n
  TRANsform coordinates
    ....

  BLENd
    .....
    etc.
NEXT
```

As a rule, any other commands which describes node or elements may be placed within a `LOOP-NEXT` pair.

Often after using a loop to read multiple parts of a mesh it is necessary to merge all the parts together so that common interfaces have the same node numbers. This is performed using a `TIE` command that is placed after the mesh `END` command (see Sect. 11.1).

9.3.1 Problems with parameters

The LOOP-NEXT commands may also be used to solve problems which have a changing parameter. The basic construct is given as

```
PARAMeter
  <set initial parameter value(s)>
```

```
LOOP no_loops

  INCLude name_of_input_file

  PARAmeter
    set_next_value(s)

NEXT

<INTEractiv>

STOP
```

In general the values for the parameters must be incremental changes or change by a multiplier factor. For example, it is possible to study problem behavior under mesh refinement provided the initial mesh is described by a few parameters. Similarly, one can investigate problem behavior for changes in material parameters.

9.4 Node and element numbers: *AUTo or *NODe and *ELEm

When using the include and loop options described above it is often necessary to assign new node and element numbers to the input values. It is possible to set node numbers using parameters with constructs such as

```
COORdinates
  n+1 1  0.0 0.0
  n+5 0 10.0 0.0
  ...
```

and then reassign the value of the parameter `n`. However, a more expedient method is to use `*AUTo` option or the `*NODe` and `*ELEment` option.

If each of the individual parts of the mesh description start with numbering from one (1), then use of the `*AUTo` may be used to merge them into a mesh with consecutive numbering. The command is inserted before the first part as

```
*AUTo
```

followed by each part. The parts may be given by the usual forms for `COORdinates` and `ELEMents`, either in the same file or in `INCLude` files.

To use a `*NODe` option a command

```
*NODe = 'expression'
```

where `'expression'` can be any *FEAP* constant, parameter or function expression. The value obtained from the expression will then be added to any nodal values appearing in an input. For example an input of an element statement as:

```
*NODe=31
ELEMent
   5 0 1 1 14 ...
   ....
```

would result in the actual numbers for the element being reported in the output file (and in the program arrays) as

```
5   0  1 32 45 ...
```

Use of a `*ELEment` option is given by a command

```
*ELEment = 'expression'
```

and any input of an element value would be incremented by the value of the expression.

It is not necessary to use the `*AUTo` or the `*NODe` and/or `*ELEment` commands with `BLOCk` or `BLENd` inputs.

The default value for `*NODe` and `*ELEment` is zero.

Chapter 10

END and miscellaneous basic mesh commands

The above set of commands are part of the basic mesh input commands available in *FEAP* to generate a mesh. The basic set also include the commands `PRINt` and `NOPRint` which turn on and off, respectively, the writing of data to the *FEAP* output data file. Once a mesh has been generated and checked it is usually not necessary to continue writing the input data to the output file. For large problems the writing not only generates large disk files but also requires additional processing time.

The final data item for the specification of the mesh data is the `END` command. Once this command is issued *FEAP* stops processing mesh input commands, may generate missing data, and looks for commands to manipulate the mesh or to solve a problem using a `BATCh` or `INTEractive` method of processing data. The options to manipulate the mesh are described in Chapter 11 and procedures to solve and plot results are presented in Chapters 14 and 15, respectively.

The basic structure for defining a finite element mesh for *FEAP* has been presented in the previous sections. The basic structure defined was:

```
FEAP * * title record  ! start analysis
  0  0  0  ndm ndf nen ! Control record

PRINt/NOPRint ! place data in output file or not

Define the mesh data for nodes and elements

Define boundary conditions and loads

MATErial number
```

```
  type_element ...
    material parameters
    geometric parameters, etc.

END  ! Last record of mesh input
```

In the next sections we describe how the mesh data may be further modified and also the steps to construct and display a solution for the problem.

Chapter 11

Mesh manipulation commands

Once an initial mesh is completely defined it may be further processed to merge nodes with the same coordinates using the `TIE` command, or force a sharing of degrees-of-freedom using the `LINK`, `CLINk`, `ELINk`, `ILINk` and/or `SLINk` commands. These commands may be given in any order immediately following the mesh `END` command. While they may be in any order the data is first saved in temporary files and *FEAP* later executes the commands in a definite order. Thus if data printing is on information may appear in a different order than given in the input file.

11.1 TIE command

The ability to merge nodes which have the same coordinates to within a specified relative tolerance permits the generation of parts of a mesh separately without having to consider a common node numbering system between the individual parts. The `TIE` command permits merging based on material set numbers, region numbers, a range of node numbers, or on all the defined node numbers. The latter is achieved by entering the command as:

```
TIE
```

without any parameters. *FEAP* will locate all nodes for which all coordinates are the same to within a tolerance given by

$$|x_i^\alpha - x_i^\beta| \leq \text{tol}\,|x_i^{max} - x_i^{min}| \quad \text{for } \alpha \neq \beta$$

where i is a coordinate direction,α and β are node numbers, and x_i^{max} and x_i^{min} are the maximum and minimum values for the $i-$coordinate. A default value is set for the relative tolerance (*gtol*) based on the number of nodes in the problem Currently this is set as

$$\text{gtol} = \frac{1}{1000\,\sqrt{\text{NUMNP}}}$$

however this may be replaced by a user specified value using

```
TIE GAP gtol
```

or

```
TIE TOL gtol
```

or where `gtol` is a fraction of the mesh size.

The basic form for all other tie options is given by the form

```
TIE type n1 n2 n3 n4
```

where `type` may be any one of the forms given below (or blank) and `n1` to `n4` are parameters for the alternate forms.

A single range of node numbers to search may be specified by giving numeric values to `n1` and `n2`. For example, if the merge is to be done only for nodes numbered between 34 and 65 the command is issued as:

```
TIE NODE  34  65
```

In order to merge nodes from from different ranges of numbers the above command may be repeated.

Instead of using node numbers it is possible to merge all nodes which are located at a given position using the command

```
TIE COORdinate  x1 x2 x3
```

where `x1, x2, x3` is the coordinate set for the point to be merged.

It is also possible to merge parts based on material numbers. For example, if a problem with two bodies is generated using material set 1 for body one and material set 2 for body two, a merge may be achieved for the parts of each body without any possibility of merging nodes in body one to those in body two. This is achieved using the commands:

```
TIE MATErial 1  1 <tol>
TIE MATErial 2  2 <tol>
```

If it is desired to tie nodes for materials 1 and 2 together, the command

```
TIE MATErial 1  2
```

may be used.

Alternatively, the nodes to be merged may be associated with a *region*. In this option it is necessary to include `REGIon` commands as part of the element generation process (i.e., using either `ELEMent` or `BLOCk`). An example of this option is explained as part of Example 4 in the *Example Manual*. The basic command to merge parts in *Region* m to those in *Region* n is

```
TIE REGIon n1  n2
```

The parameters `n1` and `n2` may have the same or different values.

When the tie option is used one node from a merged pair is deleted from the mesh and its number on the element connections replaced by the retained number. It is not possible to display or output values for the deleted node. If printing is in effect at the end of the mesh generation process, the nodes deleted are listed in the *FEAP* data output file. For plots, the projections will also be performed assuming the deleted node does not exist.

It is also possible to revert to the original *untied* mesh using the command form

```
TIE OFF
```

This is accomplished by saving the original untied mesh to a disk file along with arrays containing force, displacements and boundary conditions defined by the mesh commands `FORCe`, `DISPlacement` and `BOUNdary`, respectively. Values specified by edge or coordinate forms (e.g., `EFORce` or `CFORce`, etc.) are not saved and if needed must be reinput. The `TIE OFF` command may also be given a mesh manipulation or a solution command. That is, the command may appear inside a `BATCh` set as:

```
BATCH
  ....
  TIE OFF
  ....
END ! Batch
```

11.2 LINK, CLINk, ELINk, ILINk and SLINk commands

The link options may be used to make the solution of one or more of the degrees-of-freedom associated with two nodes have the same value. This option is useful in creating repeating type solutions, that is, those in which the solution on a surface is repeated on an identical surface with a different location. The link may be performed based on node numbers using the `LINK` command, or for all nodes on an edge using the `ELINk` command.

LINK Command

The `LINK` command structure is given as a list following the basic command. The form is

```
LINK
  node-m1 node-m2 inc-m1 inc-m2 id1 id2 ... id-ndf
```

```
node-n1 node-n2 inc-n1 inc-n2 id1 id2 ... id-ndf
   ...  ! repeat for additional node ranges
        ! blank termination record
```

in which `node-m1` to `node-n1` is used in increments of `inc-m1` and `node-m2` to `node-n2` at increments of `inc-m2`. The linking of each degree of freedom is performed if the `id` code is zero and not linked if non-zero. A linked degree of freedom will have the same solution at each of the two (or more) node(s). Omitting the `id*` values links all degrees of freedom. This is the default.

CLINk Command

A link between nodes, materials or regions for nodes which have the same nodal coordinate postion may be set using the `CLINk` command. The basic input form is

```
CLINk
  <MATErial, REGIon> n1 n2 (idf(i),i=1,ndf)
```

in which `n1` and `n2` are two material or region values and `idf(i)` values of "0" are linked so that the global equation numbers are the same and a non-zero value is not linked. The other option is

```
CLINk
  NODE (idf(i),i=1,ndf)
```

in which a search of all nodes in the mesh is made and links established for all nodes that share the same coordinates. If necessary a tolerance for the search may be set before the options using

```
CLINk
  GAP   tol_value
  <MATErial, REGIon, NODE> .....
```

Multiple records are allowed for connecting different material or region sets.

ELINk Command

A link along an edge with the same coordinate to another edge with a different coordinate is specified as

```
ELINk
  dir x1 x2 id1 id2 .... id-ndf
     ...  ! repeat for additional values
          ! blank termination record
```

where `dir` is the coordinate direction and `x1` and `x2` the coordinate values in that direction to use in the link. The link for each degree of freedom is interpreted in the same manner as for the `LINK` command. A link will take place for all nodes which have the same values in all the other directions (than `dir`).

ILINk Command

A link between nodes of interface elements may be specified as

```
ILINk
  ma id1 id2 .... id-ndf
     ...  ! repeat for additional values
          ! blank termination record
```

This requires elements to be defined as type `INTErfrace` when generating the mesh.

SLINk Command

The `SLINk` command is identical to the `ELINk` command except no check on other directions than `dir` are made. Generally, this command should only be used to link a *surface* to a single node. For example, this allows for a load on a surface to be specified by a single force applied to the linked node.

11.3 `INITial` conditions

The solution of transient problems requires the setting of initial conditions.

Initial displacements

FEAP permits the setting of non-zero initial values of the generalized displacements using the mesh manipulation command:

```
INITial DISPlacements
  n1 inc d1 d2 ... d-ndf
  n2 .....
      ! Blank record to terminate
```

where `n1` is a node number, `inc` an increment to the next node `n2`, and `di` the values of the initial displacement for each degree of freedom. The command is placed after the mesh `END` command and before the first `INTEractive` or `BATCh` command.

Initial velocity & acceleration

The initial conditions for velocity or acceleration (for problems with second time derivatives) is given by

```
INITial <RATE VELOcity ACCEleration>
  n1 inc r1 r2 ... r-ndf
  n2 .....
      ! Blank record to terminate
```

and has the same input structure as the displacements.

Initial conditions by material or region

If the initial conditions in a material set or region are constant they may be set using the commands

```
INITial <MATErial REGIon>
  <DISPlacement RATE VELOcity ACCeleration> nn u(1:ndf)
     ....
      ! Blank record to terminate
```

where `nn` is the material set or region number and `u(1:ndf)` the initial condition value for each degree of freedom.

Alternatively, the commands may be given as part of the set of solution commands as described in Section 14.1.15.

11.4 PARTition command

The solution of coupled problems may be performed by *FEAP* either as a total problem or by partitioning the problem into separate smaller problems. For example, the solution of a coupled thermo-mechanical problem may be performed by solving the thermal and the mechanical parts of the problem separately for each solution time.

By default all the degree-of-freedoms in a problem are assigned to the first partition. To assign individual degree-of-freedoms to different partitions the command

```
PARTition
  <list of dof's in partition 1>
  <list of dof's in partition 2>
     ....
  <list of dof's in partition n>
       ! Blank record to terminant inputs
```

is inserted after the mesh `END` command and before the first solution command. Currently, the only 1 to 4 partitions may be solved. An active degree-of-freedom is indicated by a non-zero number in the list, while an inactive one is given as a zero (0 or a blank). Each record must have at least one active degree-of freedom. Also, to have a complete analysis for all defined degrees-of-freedom (i.e., the total number specified on the control record described in Section 5.1) each degree of freedom must appear at least once in the lists.

The use of partitions can significantly reduce the cost of solving some coupled problems since the size of the coefficient matrix for each of the parts can be much smaller than that of the total problem. Furthermore, in *FEAP* the type of algorithm to solve each part can be set individually. Thus, it is possible to set a static option for the mechanical part and a transient algorithm for the thermal part. The individual parts may also be symmetric whereas the fully coupled problem is often unsymmetric. Such is not the case for the fully coupled solution algorithm where some care must be given to prevent

numerical problems. One is the different order of the equations which may be treated as described in the next section.

An example for a coupled thermo-mechanical problem in three dimensions may be specified in three partitions as (with temperature as dof 4):

```
PARTitions
   1  1  1  0     ! Partition 1
   0  0  0  1     ! Partition 2
     ! Blank record
```

Here the first partition is just the mechanical (displacement) part and the second just the thermal part. Alternatively one could give the partitions as

```
PARTitions
   1  1  1  0     ! Partition 1
   0  0  0  1     ! Partition 2
   1  1  1  1     ! Partition 3
     ! Blank record
```

where now a third partition permits a fully coupled solution.

A second example could be for a fluid solution using a split algorithm in which partial velocities are computed in the first step, the pressure in the second step and the final velocities in a third step. For a 2-d problem with velocity as the first 2 dof's and pressure the third the partitions are given by

```
PARTitions
   1  1  0         ! Partition 1
   0  0  1         ! Partition 2
   1  1  0         ! Partition 3
     ! Blank record
```

Note that the velocity dof's appear in two partitions, however, the partition number (`NPART`) can be used by the programmer to control the actual solution process performed.

During a problem solution a partition is activated using the solution command

```
PARTition,,n1
```

where `n1` is the partition number to activate.

Any solution dependent commands which affect a partition *must* be provided after the `PARTition` command. For example, if a transient solution is used in a partition the activation is given as

```
PARTition,,n1
TRANS <NEWMark,BACK, etc.>
```

Any solution steps are provided in a similar manner.

The use of partitions may be disabled by using the command

```
MONOlithic
```

which activates all degree of freedoms and performs as if no partitions are used.

11.5 ORDEr command

In the solution of coupled problems the individual parts often involve solution of transient problems with different orders. For example, in the solution of coupled thermo-mechanical problems a transient heat problem is of first order while a transient mechanical problem is of second order. Solution of these problems in a fully coupled mode requires use of a single transient algorithm. Thus, for example to solve the fully coupled transient thermo-mechanical problem can be performed using any of the algorithms defined in Section 14.4.3. There can be numerical problems in solving the thermal problem if large numbers of time steps are used. The problems originate from the missing second order rate term in the thermal problem which may cause the numerical acceleration to generate an overflow and thus terminate an analysis. To avoid this difficulty the `ORDEr` command may be inserted as

```
ORDEr
  O1,O2,O3,....
```

where the `Oi` define the order of the transient term for each degree-of-freedom and for *FEAP* must be defined between 0 (static) and 2 (second order). Thus, for a coupled thermo-mechanical problem the temperature degree-of-freedom should be set to one (1) and the displacement degree-of-freedoms to two (2).

Chapter 12

Contact problems and tied interfaces

The solution of problems in which the boundaries of one part of the system may interact with the boundaries of another part are called *contact* problems. *FEAP* provides options to solve such problems by imposing conditions to prevent the penetration of one body into another. In order to specify a contact analysis it is necessary to define the *surfaces* of the bodies which are to be considered during a contact analysis. In addition a user must describe *pairs* of surfaces that are to be considered as possible contacting bodies. Finally, the behavior of one surface interacting and/or sliding against another must be specified by a *material* model.

In the current release of *FEAP* the control of the penetration between bodies is implemented as a penalty, an augmented Lagrangian or a Lagrange multiplier method. For general problems the penetration is monitored from a *slave* point and a *master* point. The slave point is a node and the master point is either a node or is interpolated from a facet associated with the boundary of an element. The former is referred to as a *node to node* strategy and the later a *node to surface* strategy.

An alternative to intermittent contact is the *tying* together of mesh surfaces where the nodes on one surface to not necessarily match the nodes on the other surface. In *FEAP* this is called a *tied interface*. The tying of interfaces is accomplished using a *dual mortar* Lagrange multiplier method and forms the first surface to surface implementation accomplished in *FEAP*.

A tied interface or a contact problem is described by inserting the surface, pair and interface material information into the input data file after the mesh `END` record and before the first set of solution commands. The data may be given either before or after mesh manipulation commands using a command sequence:

```
FEAP * * Start of Problem
```

```
  .......
END of mesh
CONTact options
  ......
END contact
```

defines the extent of contact input records. The parameters permitted for *options* are: *ON* or blank to define an active input; *OFF* to skip contact inputs; or *DEBUg* to output debugging information during input and solution steps. Detailed forms for all the contact commands are included in Appendix C.

12.1 SURFace definitions

After the CONTact command it is necessary to define at least two surfaces which will be considered during the contact. A surface command is defined by the form

```
SURFace number
  surface-type <snodes>
    surface data sets
                  ! termination record
```

where number is a numerical identifier for the surface which will be used as part of the PAIR data defined below. The surface-type defines the shape of a contact facet and must be selected from: POINt, LINE, TRIAngle, QUADrilateral or RIGId. The optional parameter snodes specifies the number of nodes for each facet. For low order elements this is not required, however, if higher order elements are used the number should be specified. *Note at present no high order contact is included so this option is for use with user generated modules only.* The LINE option is used for two dimensional problems. The TRIAngle and QUADrilateral options are used for three dimensional problems. The POINt and RIGId options may be used in two or three dimensional problems.

The surface data for a *rigid* type is set by a FUNCtion data set. The surface data sets for other types may be defined using a FACEt, BLOCk, BLENd, or REGIon option.

12.1.1 FACEt specification

The FACEt option to specify a contact surface requires specification of the node numbers for each element boundary segment. Typical data for a two dimensional problem with 2-node element boundary segments consists of:

```
SURFace number
  LINE <snodes>
    FACEts
```

```
    M MG Mnode-1 Mnode-2
    N NG Nnode-1 Nnode-2
       ......
                  ! termination record
```

where generation occurs from facets `M` to `N` using increments of `MG` to each `Mnode-i`. This is performed in a manner similar to the element generations using the `ELEMent` mesh command. The facet inputs must describe a single surface entity, that is, there can be no gaps between any facets. The facets do not need to be in order but must be complete for a single surface. A surface may be *open*, with two distinct end points, or *closed* as for a circular disk.

For slave surfaces used in the node to surface solution strategy or either surface used in the node to node strategy each contact surface may be defined by a set of *points*. Typical data for points consist of:

```
SURFace number
  POINt
    FACEts
      M MG Mnode
      N NG Nnode
         ......
                  ! termination record
```

where generation is defined between the points `M` to `N` with the node numbers incremented by `MG`.

12.1.2 BLOCk specification

The `BLOCk` option is analogous to the way surface loads are generated using the `CSURface` mesh command. The data for 2-node boundary segments is given as

```
SURFace number
  LINE <snodes>
    BLOCk SEGMent
      1 x-1 y-1
      2 x-2 y-2
      3 x-3 y-3
                  ! termination record
```

If only two master nodes are used to describe a block the segment is a straight line, whereas three points describe a parabola in the natural coordinate space. The `BLOCk SEGMent` command may be preceded by a `BLOCk GAP value` to increase or decrease a search tolerance and/or by a `BLOCk POLAr` command to perform the search in polar (or cylindrical) coordinates. Thus, to specify a cylindrical surface for 3-d problems the data is described by

```
SURFace number
  QUADrilateral <snodes>
    BLOCk POLAr
    BLOCk SEGMent
      1 r-1 theta-1 z-1
      2 r-2 theta-2 z-2
      3 r-3 theta-3 z-3
      4 r-4 theta-4 z-4
                  ! termination record
```

It is important that the 4-nodes are specified to give an *outward pointing normal* to the surface.

12.1.3 BLENd specification

Blending function inputs for two- and three-dimensional contact surfaces are constructed using supernodes, `SNODes`, (see, Sect. 5.2.5). The data for the super node sets must be contained within the normal mesh input data.

Two dimensional problems

For two-dimensional problems the `BLENd` option is analogous to the way sides are generated for blending function mesh generation. Thus, a contact surface may be defined using the data:

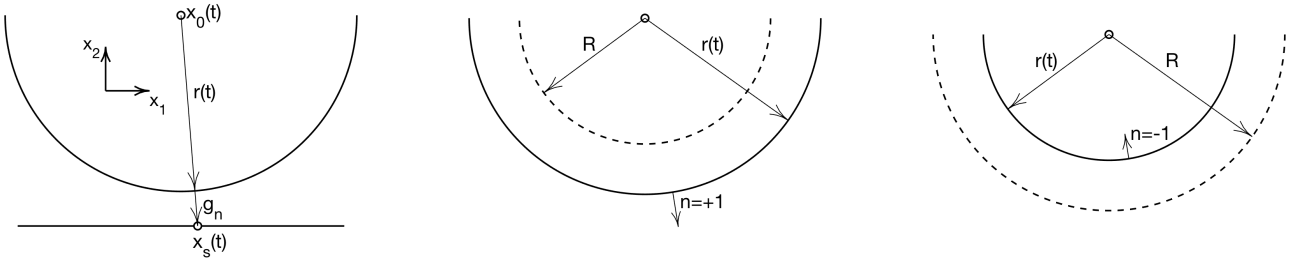
```
SURFace number
  LINE <snodes>
    BLENd SEGMent
      type sn-1 sn-2 sn-3 ....
                  ! termination record
```

where `type` is selected from the options `CART`esian, `POLAr`, `SEGMent`, or `ELIPse` and `sn-i` are super nodes (`SNODes`) for the segment.

Three dimensional problems

For three dimensional problems the `BLENd` data is used to define a *surface*. Each surface is defined by four `SNODes` and a contact surface is given by the data:

```
SURFace number
  QUAD <snodes>
    BLENd SEGMent
```



(a) Gap function (b) Positive n (c) Negativ n

Figure 12.1: Rigid contact between cylinder or sphere.

```
sn-1 sn-2 sn-3 sn-4
                ! termination record
```

The sequence of the super nodes `sn-1` to `sn-4` defines a right-handed rule with the normal pointing outward from the surface.

12.1.4 Rigid surface specification

The `RIGId` surface type requires the specification of a function and data to define the function. Currently, the function types for a cylindrical, spherical or plane surface may be defined. Each surface is allowed to *deform* to a similar shape or position. Thus, a cylindrical or spherical shape can change its radius without translation; similarly, a Cartesian plane may translate in its normal direction. Rigid surfaces are not attached to any elements and can only change by *specified* amounts – accordingly, there are no unknown parameters associated to each surface. A rigid surface is defined by the data

```
SURFace number
  RIGId
    FUNCtion type (list(i),i=1,n)
                  ! termination record
```

where `type` may be: `CYLInder`; `SPHEre` or `CARTesian`.

For a cylinder or sphere, the `list` data is given by: $+1$ for a surface normal directed outside the cylinder or sphere or -1 for the normal pointing inside; R the initial radius of the cylinder; u the displacement magnitude for moving the surface; and i the proportional load number which multiplies u. Thus, the deformed position of the cylinder is given by

$$r(t) = R + u\, n\, p_i(t) \quad .$$

The gap is given by (see Fig. 12.1)

$$r(t) + g_n = [(\mathbf{x}_s - \mathbf{x}_0) \cdot (\mathbf{x}_s - \mathbf{x}_0)]^{-1/2}$$

A Cartesian plane is described by the data: $\pm n$, where n is the coordinate direction of a normal with a positive sign used to indicate the normal points in a positive coordinate direction and a negative sign for one in a negative coordinate direction; X_n the value of the coordinate defining the reference (initial) position of the plane; u_n the magnitude of the displacement; and i the proportional load number. Thus, the deformed position of the plane is given by

$$x_n(t) = X_n + u_n\, p_i(t) \quad .$$

Rigid surfaces may only be used as *master* with the *slave* being a `POINt` or other type of surface associated with nodes on the finite element mesh.

12.2 MATErial models for contact

The behavior of one surface interacting with another may be modeled in different ways. The current release includes a very simple model in which the surface is considered as regular (no roughness or micro-mechanical details are to be specified) but may have frictional resistance. For frictionless contact no material definition is required - *FEAP* will assign default conditions. If friction is present it is necessary to define the Coulomb frictional parameter by including the data set

```
MATErial number
  STANDard
    FRICtion COULomb value
                    ! termination record
```

where `value` is a positive constant coefficient of friction.

12.3 PAIR definition

The interaction between two surfaces is controlled by the `PAIR` command. This command describes which two surfaces are to be considered, the type of contact solution, the solution method, and solution tolerances. A typical data set for solution of problems using a *node to surface* strategy is given by

```
PAIR number
  NTOS slave master
  SOLM PENAlty k-n k-t
  TOLE values  t-1 t-2 t-3
                    ! termination record
```

The parameter `number` is an identifier numeral for the pair. The node-to-segment (`NTOS`) solution strategy requires the specification of a `slave` surface identifier numeral and a `master` surface identifier numeral. The solution method may be given as `SOLM PENAlty` with `k-n` and `k-t` the *penalty* parameters used for normal penetration control and tangential stick control, respectively. Alternatively, the command may be given as `SOLM LAGM` to impose normal gap constraints using a Lagrange multiplier method. The parameter `k-n` may also be used to provide some *stiffness* on the surface. This stiffness is effective only during iteration process – final gap is imposed exactly using the Lagrange multiplier approach. The `TOLE`rance option defines the values for solution tolerances: `t-1` is a tolerance for defining initial penetration; `t-2` is a tolerance for considering a contact open; and `t-3` is an out of segment tolerance. Generally, some value for the out of segment tolerance is required to maintain contact when a slave node moves from one master segment to the next.

Use of a penalty method of solution can lead to undesirable penetration of one surface into the other. This may be reduced by using an augmented Lagrangian method of correction based on the Uzawa algorithm. To activate the augmented updates it is necessary to add an additional record to the pair specification as:

```
PAIR number
  NTOS slave master
  SOLM PENAlty k-n k-t
  AUGMent
  TOLE values  t-1 t-2 t-3
                   ! termination record
```

To activate the augmentation updates it also is necessary to specify an `AUGMent` command in the solution commands as described later in this section.

If friction is to be considered it is necessary to modify the pair set to include the material set, thus a typical set will be:

```
PAIR number
  NTOS slave master
  SOLM PENAlty k-n k-t
  MATE  ,,     m-1 m-2
  TOLE values  t-1 t-2 t-3
                   ! termination record
```

where `m-1` is the material number for the slave surface and `m-2` that for the master surface.

An option also exists to permit a specified gap between the master and slave surfaces. This may be used to model the half thickness of a shell. The command is given as a feature of a pair command as

```
PAIR number
```

```
 ....
THICkness CONStant gn_th
 ....
```

where gn_th is a normal distance between the surface pairs.

12.3.1 Pair time control

For solutions of problems in which there are known times when the contact of a pair will not be active for many solution steps a time function can be specified that acts as a switch. A pair can be specified as

```
PAIR number
  ......
  SWITch TIMF prop-number
  ......
                    ! termination record
```

In this case it is necessary to specify a proportional loading (See Sec. 14.1.28) that has positive and zero or negative values. For solution times where the proportional load is positive the pair will be active and for solution times where it is negative or zero the pair will be switched off.

12.3.2 Tied interface pairs

The tied interface option is implemented for 2-dimensional plane problems only. It may be used to connect any combination of linear, quadratic or cubic edged elements. Best results are achieved if the connecting surfaces have curves no higher than the order of the *lowest order* element. Thus, connecting linear edges to quadratic edges should have *linear or straight* mesh boundaries.

A tied interface is defined using the PAIR command. The data is given as:

```
PAIR number
  TIED slave master quad-order
  SOLM LAGM
                    ! termination record
```

where, in addition to the slave and master surface numbers, a quadrature order is specified by the parameter quad-order. The quadrature order should range between 2 and 5 (a default is specified as 4) and indicates the number of Gauss-Legendre points used to evaluate surface integrals on each segment.

12.4 Plot of contact information

Some details of a contact problem may be graphically displayed using `PLOT` command options as described below. The commands may be given in either batch or interactive modes.

12.4.1 Contact pair geometry

The mesh parts defining the contact pairs may be displayed using the command:

```
PLOT PAIR p1 p2 surf
```

where `p1` is the first pair to display and `p2` is the last pair to display. The value of `surf` is given as:

1. surf = 0: Plot both slave and master surface.
2. surf = 1: Plot both slave surface only.
3. surf = 2: Plot both master surface only.

12.4.2 Contact pair variables

Values for contact pair variables may be displayed using the command

```
PLOT CVARiable var vec pair
```

where `var` is the variable number to display, `vec` is the vector number from which to extract the variable and `pair` is the contact pair number. If the value of `vec` is omitted or zero the values are selected from the solution at the current time. The `vec` options are:

1. vec = 1: Solution from array `CH1` containing solutions from time t_n.
2. vec = 2: Solution from array `CH2` containing solutions from time t_{n+1}.
3. vec = 3: Solution from array `CH3` containing time independent data.

If the value of `pair` is omitted or zero all pairs are displayed; otherwise the number of the specific pair should be given.

The meaning of the variable `var` is different for each contact pair type. The most useful information is often the value of the gap obtained during solution; specifically, where the contact region is active is often desired. For 2- and 3-dimensional `NTOS` contact the number of the variable is 9 for the contact vectors `CH1` and `CH2`. Thus, the gap at the current time for all contact pairs may be displayed using the command:

```
PLOT CVAR 9
```

12.5 Example for Contact Input

As a simple example, consider the definition of a contact interaction between two blocks as shown in Figure 12.2. The *FEAP* input data for the contact part of the mesh is given by:

```
FEAP * * Start of Problem
  .......
END of mesh data
CONTact
  SURFace 1  ! Define first surface
    LINEar
      FACEts
        1  0  9  8
        2  0  8  7

  SURFace 2  ! Define second surface
    LINEar
      FACEts
        1 -3 19 16
        3  0 13 10

  PAIR 1     ! Define contact pair
    NTOS  1  2
      SOLM PENAlty  1.E+05

END contact data
```

As defined above, surface 1 is the *slave surface* and surface 2 the *master surface*. Note that in the above example no `MATE`rial parameters are specified. For the `PAIR` command a *penalty method* is requested and its parameter is the value of the penalty parameter associated with the normal direction.

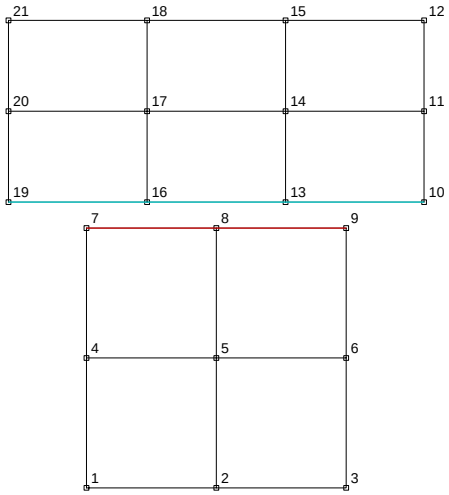


Figure 12.2: Contact analysis between two bodies

Chapter 13

Rigid-flexible body analysis

The combination of deformable solids with rigid body parts may be considered using *FEAP*. The capabilities are split into two classes. One limited to small displacement problems where translation and rotation parameters may be considered, and a second more general class in which rigid bodies may be interconnected using *joint* elements. The general class is primarily for cases which have both large displacements and large rotations. However, it may also be used for quasi-static response of linear problems.

13.1 Small displacement analyses

For the small displacement case the treatment for rigid parts is performed using a *master-slave* concept for prescribed degrees-of-freedom. A simple implementation is included in the current version which permits degrees-of-freedom for a slave node or a constant coordinate value to be represented in terms of degrees-of-freedom at a master point. It is possible to have some degrees-of-freedom rigid while others remain flexible. For example, a floor slab of a building may be constrained to be rigid for in-plane deformations but flexible in transverse (plate bending) motions. The commands for specifying the master-slave set are inserted after the mesh `END` command and before the first solution data set. The basic structure is:

```
MASTer
  TYPE (EV(i),i=1,n)
```

The `TYPE` options are : `NODE`, `SURF`ace, and `GAP`. For `NODE` the input record is:

```
NODE (Xm(i),i=1,ndm) (Xs(i),i=1,ndm) (RLINK(i),i=1,ndf)
```

The nodes closest to the specified coordinates will be selected as the master (`Xm`) and slave (`Xs`) nodes. Zero values in the `RLINK` pattern define the degrees-of-freedom to be

considered during the slave phase. The pattern must be consistent for proper behavior. Thus, if the x_1 and x_2 displacements are slaved so must the θ_3 rotation parameter. Similarly, for other patterns.

The record for SURFace is input as:

```
SURFace (Xm(i),i=1,ndm) dir (RLINK(i),i=1,ndf)
```

Here in addition to the master node coordinates, the direction of a normal to the plane passing through the master node must be given. Thus if `Xm` is given as (0 0 5) and `dir` as 3 then all other nodes within the gap value with coordinates (x_1 x_2 and 5) will be treated as slave nodes. The value of the gap may be reset from its default value of 10^{-8} using the `GAP EV(1)` command.

13.2 General rigid-flexible solids

In performing a rigid-flexible body analysis for problems which undergo large motions and rotations (or for quasi-static small deformation problems) it is necessary to designate the elements which are *rigid* and those which are *flexible*. In addition it is necessary to *activate* the analysis option. Designation of elements to be rigid or flexible is given during mesh generation and activation as part of the mesh manipulation commands.

The current release of FEAP does not fully support all rigid body options. Transient problems may be solved using the energy conserving algorithms; however, other algorithms may not converge quickly or at all.

13.2.1 FLEXible or RIGId groups

FEAP permits the use of both flexible and rigid finite elements. By default all elements are flexible. There are two ways to declare change the computation of element arrays from flexible to rigid: (1) During input of the `MATEerial` set; or (2) During input of the element connections.

Rigid material sets

The elements associated with a material set may be declared to be rigid by using the command set

```
MATErial ma
   <type>
```

```
    ...
RIGId material n_rig
    ...
```

where `n_rig` specifies the rigid body number.

This is a good option to use also when preparing an input file for large or complicated problems since testing can be done on individual parts but also include effects of the remaining parts using `a reduced order model` defined only in terms of the rigid body equations, 3-dofs in 2-dimensional problems and 6-dofs in 3-dimensional ones. Later analyses can be performed by merely removing the `RIGId` material record. This is also a good option for cases where *all* elements in the material set are rigid.

Rigid element connection sets

If it is desired to designate element connection sets as rigid the record

```
RIGId r_num
```

(where `r_num` is the rigid body number) is inserted in the input data just before the group generated using the `ELEM`ent, `BLOC`k, or `BLEN`d commands.

To designate elements as flexible the command

```
FLEXible
```

is inserted immediately before element groups which are to remain deformable. It is not necessary to include this statement if all elements are flexible.

13.2.2 Activation

As noted above *FEAP* permits material sets or groups of finite elements to be declared as rigid or flexible during the input of mesh data. In order to activate the rigid option, it is necessary to also define the type of integrations to perform for the rigid bodies and to define any interconnections (*joints*) that exist between different rigid bodies or a rigid body and a flexible body node. The activation is achieved by inserting a `RIGI`d command *after* the `END` of mesh record and *before* the first solution `BATC`h or `INTE`ractive command. Similarly, to define joint interconnections any `JOIN`t commands are also placed in the same location.

FEAP will automatically constrain groups of rigid elements which are contiguous to flexible elements to perform a combined flexible-rigid body analysis. At present the rigid body options are limited to solid (continuum) elements only. Both explicit and implicit transient solutions are possible; however, for the explicit option only the Spherical (Ball and Socket) Joint described below is permitted. The implicit formulation is

available for the energy-momentum formulation only and permits the use of several types of joints and all constraints are formulated using a Lagrange multiplier method. It is not possible to consider closed loops consisting of only rigid bodies since redundant Lagrange multiplier constraints will exist.

To activate the rigid body options and to define the integration method the single record

```
RIGId,Nrbdof,Npart,Ntype
```

is inserted between the `END` mesh command and the *first* solution command (`BATCh` or `INTEractive`). In this statement `Nrbdof` is the number of rigid body degree-of-freedoms, `Npart` is the partition number of the rigid body, and `Ntype` is the integration type. For most analyses the parameters may be omitted and *FEAP* will insert correct default values. The default values for `Nrbdof` are:

Mesh Dimension	Value
1	1
2	3
3	6

By default `Npart` is assigned to partition 1 and `Ntype` is set to the energy-conserving algorithm which is number 5. (N.B. Other options have not been tested and, thus may not be operational).

There are additional commands for controlling rigid body, these include restraining the rigid body to be fixed in space by a boundary condition:

```
RBOUndary
  n_body bc_1 bc_2  ....
```

In addition a load may be applied to an unconstrained rigid body (no boundary condition) using

```
RLOAd
  n_body f_1 f_2  ....
```

Non-zero rigid body loads or displacements may be scaled by specific proportional loads using the command

```
RPROp
  n_body p_1 p_2  ....
```

If the command is omitted or any `p_i` is zero the value is scaled by the *sum of all proportional loads*, thus, if the load is to remain constant it is necessary to specify a proportional load that remains constant.

Additional details for rigid body solution commands may be found in Appendix B..

13.2.3 Joints

Rigid bodies may be interconnected using *joints*. The specification of the joints is initiated using a `JOINt` command which also is located after the `END` mesh command and before the first `BATCh` or `INTEractive` solution command. Two of the selections from the library of joints are:

1. Ball and Socket: Two rigid bodies may rotate freely about a specified point. A ball and socket joint is specified by a record

   ```
   BALL RB_1 RB_2 X Y Z
   ```

   where `RB_1` and `RB_2` are the rigid body numbers associated with the ball and socket, and `X`, `Y`, and `Z` are the reference system coordinates for the location of the ball and socket.

2. Revolute: Two rigid bodies may be constrained to rotate relative to a specified direction in the reference coordinate system. A classical revolute is formed by combining the *FEAP* `REVOlute` with a `BALL` joint. The revolute is specified as:

   ```
   REVOlute RB_1 RB_2  X_1 Y_1 Z_1  X_2 Y_2 Z_2
   ```

   where now the two coordinate points identify the direction of the rotational axis in the reference state. This axis is free to rotate in space unless constrained by other restraints.

Other types of joints are described in Appendix A.

N.B. The rigid body options are in a development mode and are not operational for all types of solution methods.

Chapter 14

Command language programs

FEAP performs solution steps based upon user specified *command language statements*. The program provides nearly 200 commands which can be used to solve problems and graphically display results using standard algorithms, such as linear and transient methods to solve linear and non-linear problems (see Sect. 14.3 on basic methods for problem solution) . Appendix D of the manual describes all the solution commands and Appendix E describes all the graphics plot commands which are included in the current system. Command language statements can be combined to define a large number of solution algorithms and plot commands can display many features of the solution.

In this chapter we first summarize some of the basic commands in the program and then describe how they may be combined to form some standard solution algorithms. In the next chapter we describe the basic plot commands that can be used to graphically display results.

To enter the solution command and/or plot command language part of *FEAP* for a batch type solution the user includes the command `BATCh` in the input file. A batch solution is terminated by the command `END`.

Alternatively, a user may issue commands interactively in a step-by-step manner by inserting a command `INTEractive` in the input file. Many sets of `BATCH-END` or `INTEractive` commands may be included in any solution. Thus, to perform a solution the input file must contain at least one set of

```
BATCh
 ....  ! Solution specification steps
END
```

or

```
INTEractive
```

statements for any solution process to be possible.[1]. Termination of an INTEractive session occurs whenever the command QUIT (or just q) or EXIT (or just e) is given. Using an EXIT command provides an option to save a restart file, whereas the QUIT exits without saving any file.

As noted above, more than one BATCh-END and/or INTEractive sequence may be included in the input file to define the solution process desired.

We next describe some of the solution commands available in the system. This is followed in Section 14.3 with examples of algorithms that can be constructed using the command language approach.

14.1 Basic solution commands

A set of *basic* solution commands is:

```
ACCE   CAPT   CHEC   DEBU   DISP   DT     EIGE   EPRI
EIGV   EXTR   FORM   GET    INIT   LIST   LOOP   MASS
MESH   NEXT   NOPR   PARA   PLOT   PRIN   PROP   REAC
SHOW   SOLV   STRE   SUBS   TANG   TIME   TOL    TPLO
TRAN   UTAN   VELO
```

Descriptions to use the above commands are contained in Appendix D. All commands available in an installed program may be displayed during an interactive mode of solution by issuing the commands:

```
MANUal,,3
HELP
```

However, with the basic set of commands given above quite sophisticated solution algorithms may be constructed. Each of the commands may be issued in a lower or upper case mode.

In addition, brief uses for most commands may be obtained by issuing

```
HELP name
```

where name is the desired command name. For example in solution mode

```
HELP TANG
```

in interactive mode will display basic use for the TANG command. For commands implemented by a user, either the command name assigned (e.g., shown by the HELP alone) or

```
HELP UMACro n
```

[1]These also may be in a separate file and accessed using an INCLude statement in the input file

where `n` is the number of the module `umacrn` may be given. The `HELP` use can also be given in `MESH` or `PLOT` mode.

In the next subsections the features of the basic commands listed above are presented. This is followed in the next section with a description of using these commands to develop some of the possible solution algorithms.

14.1.1 `ACCCEleration` command

For transient problems using solid or structural elements the solution is given in terms of displacements, velocities and accelerations at each solution time t_n. The numerical values of the acceleration at each node may be given using the command:

```
ACCEleration ALL
```

Alternatively, the values for a few nodes may be given as

```
ACCEleration nodes n1 n2 inc
```

in which `n1` and `n2` define the first and last node to output and `inc` is the increment between node numbers.

It is possible to also use the `LIST` command (see below) to describe a set of nodes for which outputs are requested and then use the commands

```
ACCEleration LIST l1
```

to output the values. Here `l1` is the list number.

Finally, the values for a constant value of the nodal coordinates may be used to output a set of values. The command is given as

```
ACCEleration COORdinate d x tol
```

where `d` is the coordinate direction (e.g., 1,2 or 3), `x` is the value of the coordinate in the d-direction and `tol` is a search tolerance to find nodes.

14.1.2 `CAPTion` command

The caption used for output of plot contours is set to default values by *FEAP*. It is possible to change the caption to a user specified identifier using the `CAPTion` command. The caption may contain from 1 to 15 character (*no blank characters are allowed*). The command is given as:

```
CAPTion label
```

where `label` is a character string.

14.1.3 CHECk command

A command which always should be issued when first solving a problem is the CHECk command. In either a batch or interactive mode, the command is issued as:

```
CHECk       !perform check of mesh correctness
```

This command instructs *FEAP* to make basic checks for correctness of the mesh data prepared by the user[2]. One of the basic checks is an assessment of the element volume (or area) at each node based on the specified sequence of element nodes. If the volume Jacobian of an element is negative or zero at a node a diagnostic will be written to the output file. If all the volumes (or areas) are negative most of the system element routines will perform a re-sequencing of the nodes and repeat the check. If the re-sequencing gives no negative results the mesh will be accepted as correct.

A check also may reveal and report element nodes which have *zero* volume. This may be an error or may result from merging nodes on quadrilaterals to form triangles. This is an acceptable way to make 3-node triangular elements from 4-node quadrilateral elements, but in other cases may not produce elements preserving the order of interpolation of the quadrilateral. *It is the responsibility of the analyst to check correctness of finite element solution software.* One good procedure is the patch test in which basic polynomial solutions, for which the user can compute exactly the correct solution (by hand), can be checked (see Zienkiewicz, Taylor & Zhu[1,16] for a description of the patch test).

The CHECk command should always be used in situations where either a new mesh has been constructed or modifications to the element connection lists have been made. *No analysis should be attempted for a mesh with negative volumes as incorrect results will result.* Note, however, that if a correct mesh is produced after the CHECk command re sequences nodes, the data in the input file is *not corrected*, consequently, it will be necessary to always use a CHECk command when solving a problem with this data input file. Since the amount of output from a CHECk can be quite large, it is recommended that the user correct the mesh for subsequent solutions. Alternatively, it is possible to produce a new input data file, which is correct, using the OUTMesh command (see Section 14.1.22).

14.1.4 DEBUg command

The DEBUg command is used to activate additional print outputs during solution phase. Generally these are statements in user written programs that are contained between the instructions:

```
if(debug} the
```

[2]The check part of user developed elements must be implemented for the check command to work properly

```
    ..... ! Output statements
endif
```

where the logical parameter `debug` is contained in the common block named `debugs`. During a solution mode the debug feature may be activated using the command

```
DEBUg <ON> ndebug
```

where the parameter `ON` and `ndebug` are optional. The parameter `ndebug` is used to control some of the outputs. For example if it is desired to track the allocation of *FEAP* arrays during the solution phase the value of `ndebug` must be greater than one (1). Indeed, the command can be placed before the `feap` start record and the entire history of data management can be followed in the output file. Thus, the following commands track the memory allocation:

```
DEBUg ON 2
FEAP * * < Problem title >
  0  0  0 .....
```

would display the memory allocation during solution. This can help to identify problems that have memory use problems by identifying the ones requiring very large memory arrays (possibly ones larger than the 32-bit integer limit).

Debug prints may be deactivated using the command

```
DEBUg OFF
```

14.1.5 `DISPlacement` command

The solution of all finite element problems contains parts of the solution given by generalized displacements at nodes. The numerical values for all the nodal displacements at each node may be output using the command:

```
DISPlacement ALL
```

Alternatively, the values for a few nodes may be given as

```
DISPlacement nodes n1 n2 inc
```

in which `n1` and `n2` define the first and last node to output and `inc` is the increment between node numbers.

It is possible to also use the `LIST` command (see below) to describe a set of nodes for which outputs are requested and then use the commands

```
DISPlacement LIST l1
```

to output the values. Here `l1` is the list number.

Finally, the values for a constant value of the nodal coordinates may be used to output a set of values. The command is given as

```
DISPlacement COORdinate d x tol
```

where `d` is the coordinate direction (e.g., 1,2 or 3), `x` is the value of the coordinate in the d-direction and `tol` is a search tolerance to find nodes.

When solutions involve added global equations (see Sect. 5.11), their values may be output using the command

```
DISPlacement GLOBal
```

14.1.6 DT command

The solution of time dependent problems (or problems in which time is used as an artifact for evolving a solution) requires the specification of a time increment, Δt. The value for a time increment is set using the solution command

```
DT,,dt
```

where `dt` is the value for the time increment of the next step.

14.1.7 EIGEn-pair command

The element array for symmetric cases may be displayed as a spectral form

$$\mathbf{S} = \mathbf{Q}^T \mathbf{\Lambda} \mathbf{Q}$$

where $\mathbf{\Lambda}$ is a diagonal array of eigen-values and $\mathbf{Q}$ the eigen-vectors. The solution is computed and displayed using the command

```
EIGE VECT n1 i1
```

where `n1` is the element number and `i1` a parameter that when positive or zero computes the spectral arrays for the tangent (stiffness) matrix and when negative the form for the mass matrix.

The output for the eigen-values can reveal the number of zero-energy modes in an element to ensure that proper behavior is attained without spurious results.

14.1.8 EPRInt command

The `EPRINt` command is used to display the values in any computed element mass or tangent and residual arrays. The command should be issued after a `TANG` or `UTAN` command to get the latest values. The command is given by

```
EPRInt <MASS,  > k1
```

where `k1` is the element number for which the result is computed. The tangent and residual are computed if the second parameter is not `MASS`.

14.1.9 `EIGVector` command

After computation of normal modes (*e.g.*, using the `SUBSpace` command) the values of specific eigen-pairs may be output using the command

```
EIGVector ALL k1
```

where `k1` is the number of the pair. Other options exist for the `EIGVector` command as described in Appendix D.

14.1.10 `EXTRact` command

The `EXTRaction` command may be used to identify and output element connection or contact facet arrays for the surface of two or three dimensional meshes. The extract command results in an output of data that defines new elements that can be subsequently added to the mesh. The element number in each set always start with one (1) and, thus, when inserted into a mesh a `*AUTo` command should be placed before the first set of data that starts with one (See Section 9.4 for further details). The command has several basic forms as described below.

Extract Segments

Given only the element connections for a solid mesh individual parts of the *smooth* surface segments may be identified using the command

```
EXTRact SEGMents cosa
```

where the parameter `cosa` denotes the cosine of the angle between adjacent facets on the surface that are considered to be smooth (default is currently 0.92). The command should be given in an interactive mode and preceded by a `PLOT PERSpective`. Each located segment will appear as a plot on the screen and an option to output the element connections or contact surface facets as a mesh part to be used in a subsequent analysis. The file will have a name `MESH_SEGM000n`, etc. for subsequent segments. The value of `n` usually starts at one more than the number of materials in the mesh.

Extract surfaces

An alternative, option is to output and display surfaces for all or individual material parts using

```
EXTRact SURFace cosa ma
```

where ma is the material set to output. If ma is zero surfaces for all material sets are output. These files have name MESH_SURF000n.

The element output files from the extraction command have a form that begins with

```
ELEMents NODES=nne MATE=nnm PART= nnp TYPE=name
```

where nne is the number of nodes on a surface facet, nnm the material set number on each output record, nnp is the part number and name is the shape of the surface facet. Generally, the shape for three dimensional problems will be triangle or quadrilateral. For two dimensional problems name will generally be line. If more than one shape of facet exists, the output file may be split into more than one ELEMent record. Each of the records is followed by the set of element nodal connection list.

While the value for the material set is set identically to the value of nnm, it may be reset by merely changing the value of nnm to the value desired.

For contact facet sets the output file has the form

```
SURFACE  s_num
  type
    FACEts
       1 0   node list
       2 0     .etc.
```

where s_num is a surface number and type the shape of the surface facet (*e.g.* in 2-d it is LINE).

Extract interfaces

The extract command may also be used to create interface elements between two different unconnected material sets that share the same surface coordinates. The command is given by

```
EXTRact INTErface m1 m2 f1
```

where m1 and m2 define the element sets for two different material sets and f1 is the number for the output element set that has the filename

```
MESH_INTE000n
```

where n is the f1 value. The file contains interface elements with the definitions

```
ELEMents NODES=nne MATE=nnm TYPE=INTErface
```

Interface elements are numbered with node pairs associated with m1 and m2 then sequenced according to one or two dimensional elements. Each set starts with element 1 – thus, the command

```
*AUTo
```

should be used before the definition of the elements (see Section 9.4 for more information). Formulations for interface elements are described by material sets

```
MATErial ma
  INTErface
    TYPE <1,2,3,4,5>
      data for the material set properties
```

14.1.11 FORM command

The solution of problems by *FEAP* is obtained by computing residuals for the governing equations followed by a sequence of solution steps to (hopefully) reduce the size of the residual to a very small value.

The residual for each step may be computed using the FORM command given as

```
FORM
```

It is also possible to perform a check on convergence using the command form:

```
FORM CONVerge
```

If convergence is detected the remaining commands in any LOOP-NEXT set are skipped. Alternatively, convergence checking may be deferred until after the solution step (see SOLVe command below).

14.1.12 GEOMetric stiffness command

The GEOMetric command is used to specify behavior of the geometric stiffness matrix. Two forms of use are defined. In the first the command

```
GEOM <ON,OFF>
```

allows many non-linear and finite deformation elements to activate or suppress formulation of the geometric part of the stiffness matrix. The default is ON. This is sometimes useful for problems in which *structural* type behavior occurs. In this class of problems generally large in-plane forces may exist in the second iteration of a Newton solution scheme. This can cause the iteration to diverge in later iterations, whereas, if the geometric term is off convergence occurs. Using solution commands

```
TANGent,,1
GEOM OFF
TANGent,,1
GEOM ON
LOOP,,nt
  TANG,LINE,1
NEXT
```

can help in convergence of a step. The command `TANGent,,1` is equivalent to using the three individual commands

```
TANGent
FORM
SOLVe
```

The second use of the `GEOMetric` command is for buckling analysis. In this case the command is used without any arguments and an eigen-solution performed using the commands

```
TANGent,,1
GEOMetric
SUBSpace,,nvalues
```

Note that the `TANGent` command also must perform a linear solution in order for the in-plane force terms to be present for the computation of the geometric stiffness. At present only frame elements and linear shell elements have geometric matrix available. Thus, only very restricted solution options are available. However, using the identity matrix some cases can be solved for nonlinear problems also. See comments in the next section.

14.1.13 `GET` parameter command

During interactive solution it is sometimes necessary to know the value of *FEAP parameter values*. The `GET` command may be used to retrieve the value of any parameter using

```
GET parameter_name
```

The default value of a parameter is zero, thus, specification of any one or two character parameter always returns a value. The value of a parameter may be set using the command `PARAmeter` (see subsection below).

14.1.14 `IDENtity` command

The command `IDENtity` forms an identity matrix applied to all degree of freedoms. This command may be used to solve some non-linear eigen-problems. In concept a nonlinear eigen problem may be conceptually viewed as

$$[\mathbf{K}_m + \mathbf{K}_\sigma]\ \boldsymbol{\phi} = \lambda \mathbf{I}\, \boldsymbol{\phi}$$

where $\mathbf{K}_m$ is the part of the stiffness associated with material moduli and $\mathbf{K}_\sigma$ that from stress effects. When a buckling occurs, the eigenvalue λ will be zero, however, then the total stiffness matrix will be singular. Hence it is necessary to specify a shift to the coefficient matrix. A shift to the coefficient matrix may be specified as

```
IDENtity
TANGent,,,alpha
```

where `alpha` will cause the matrix

$$\mathbf{K} - \alpha\,\mathbf{I}$$

With an appropriate choice for α the matrix will be non-singular and the eigen problem performed. Reported eigenvalues compensate for the presence of any nonzero α.

14.1.15 INITial command

The solution of transient problems requires the setting of initial conditions. *FEAP* permits the setting of initial values of the generalized displacements using the solution command:

```
INITial DISPlacements
```

The command is placed between the `BATCh-END` pair and requires additional data for the values after the `END` command. Thus the input structure for a batch solution is given as:

```
BATCh
  INITial DISPlacements
END
  n1 inc d1 d2 ... d-ndf
  n2 .....
      ! Blank record to terminate
```

where `n1` is a node number, `inc` an increment to the next node `n2`, and `di` the values of the initial displacement for each degree of freedom.

The initial conditions for velocity (for problems with second time derivatives) is given by

```
INITial <RATE VELOcity>
```

and has the same input structure as the displacements.

In interactive mode the input is given immediately after the `INIT` command (although this is generally not recommended as an input form).

If the initial conditions in a material set or region are constant they may be set using the commands

```
INITial <MATErial REGIon>
  <DISPlacement RATE VELOCity ACCEleration> nn u(1:ndf)
```

where `nn` is the material set or region number and `u(1:ndf)` the initial condition value for each degree of freedom.

Alternatively, the commands may be given as part of the mesh manipulation commands as described in Section 11.3. This is preferable if any history variables depend on the initial solution values.

14.1.16 `LIST` command

The output for a specific set of node numbers may be defined using the `LIST` command. Only 1 to 3 lists are permitted with each list containing between 1 and 100 values. The definition of each list is given by the command set

```
BATCh
  LIST,,n
END
  n1 n2 n3 .... n8
  n9 ....
    ! Blank record to terminate input
```

where `n` is the list number (i.e., 1, 2 or 3) and `ni` are node numbers. Each input record contains up to 8 values.

The output for a specific set of material sets also may be defined using the `LIST,MATE` command. Only 1 to 3 lists are permitted with each list containing between 1 and 100 values. The definition of each list is given by the command set

```
BATCh
  LIST,MATErial,m
END
  m1 m2 m3 .... m8
  m9 ....
    ! Blank record to terminate input
```

where `m` is the list number (i.e., 1, 2 or 3) and `mi` are material set numbers. Each input record contains up to 8 values.

14.1.17 `LOOP` command

Repeated execution of solution commands is performed using the `LOOP-NEXT` commands. The basic structure of the command set is:

```
LOOP,,n
  ...
NEXT
```

in which `n` denotes the maximum number of times the loop is to be performed. Loops may be nested up to 8 levels so that the command set:

```
LOOP,,n1
  LOOP,,n2
    LOOP,,n3
       ...
    NEXT
  NEXT
NEXT
```

is permitted and the inner statements are computed `n1*n2*n3` times.

Looping permits the construction of very sophisticated solution algorithms and provides *FEAP* with much of its capabilities.

14.1.18 MASS command

The `MASS` command computes the inertia matrix for structural problems and the heat capacity matrix for thermal problems. Three options for issuing the command are available. The command

```
MASS LUMP
```

computes a diagonal (lumped) matrix; the command

```
MASS CONSistent
```

computes a symmetric consistent matrix; and

```
MASS UNSYmmetric
```

computes an unsymmetric matrix. Issuing the command as

```
MASS
```

computes the consistent (symmetric) mass by default.

Currently, the unsymmetric form is not used by any of the other commands available in *FEAP* and is provided only for user added features.

14.1.19 MESH command

The `MESH` command is used to permit changes to the mesh data to be made during solution steps. Use of the command requires additional data to be provided *after* the `END` batch solution command (or interactively after prompts). Thus, the basic structure to use the command in a batch mode is given by

```
BATCh
  ...
  MESH
  ...
END
  .... ! Mesh input data set 1
END    ! Mesh input data set 1
  .... ! Mesh input data set 2
END    ! Mesh input data set 2
```

Note that there must be as many mesh data sets as the number of times the `MESH` command is executed. Failure to have the require number will lead to an abnormal exit from *FEAP* with the the error message:

```
*ERROR* ENDCLR: End of file encountered in TINPUT
                Unit Number = 15
```

14.1.20 `NEXT` command

The `NEXT` command must be used with a matching preceding `LOOP` command (see Sect. 14.1.17 above). Thus, it must always have the structure

```
LOOP,,n
  ....
NEXT
```

14.1.21 `NOPRint` command

During solution, each command statement results in information being written to the output data file. The command `NOPRint` may be used to reduce the amount of information written and is given as

```
NOPRint
```

within the solution command set.

14.1.22 `OUTMesh` command

The command to output a mesh with unused or merged nodes removed is given as:

```
OUTMesh     !Output current mesh to "Ifile".opt
```

The output is written to a file with the same name as the input file but with a `.rev` or `.opt` extender added (the latter when a mesh optimizer has been previously specified). The file should include all the mesh input data and in addition most of the mesh manipulation commands (e.g., links and initial conditionsr). The output of elements is separated by material number. It is necessary to append the solution steps. It is not necessary to specify any `TIE` commands as the results from merges are incorporated as part of the mesh produced by the `OUTMesh` command.

This output step is normally required if the problem is to be partitioned for a parallel solution using *parFEAP* (see the *FEAP* Parallel User Manual[14] for further details).

14.1.23 `OUTPut` command

It is possible to have *FEAP* output to a file values for some of the arrays produced during a solution. The basic command is:

```
OUTPut arrayname
```

The permitted options for the array name are described in Table 14.1 and results are contained in the format

```
I  J  A(I,J)
```

where `I` is the row number of a matrix and `J` the column number. Vectors are output with `J = 1`. The output format is compatible with MATLAB.[61] By default only the real part of solutions are output to a file named `READ_arrayname_xxx`, where `arrayname` is the specified array and `xxx` is sequence from `001` to `999`. For complex solutions the imaginary part of the array may be output using the command specified as:

```
IMAGinary
OUTPut arrayname
```

It is necessary to issue the command

```
REAL
OUTPut arrayname
```

to again output any real part of solution values. The output of real displacement values creates a file with name `READ_DISP_xxx_yyy` where `xxx` denotes the time increment value and `yyy` the iteration number of the solution. With this option it is possible now to output to separate files the solution values at each iteration.

Output of the `TANGent` or `UTANgent` matrix can be done for either the unfactored or the factored form. To output an unfactored tangent the form

```
TANGent,,-1
OUTPut TANGent
```

should be used. If it is desired to have a factored tangent then the command form is

```
TANGent
OUTPut TANGent
```

Array	Description	Filename
DR	Residual from FORM	DR
FORM	Residual from FORM or RESID	FORM
TANG	Tangent from TANG	TANG
UTAN	Tangent from UTAN	UTAN
LMAS	Mass from MASS,LUMP	LMAS
CMAS	Mass from MASS,CONS	CMAS
UMAS	Mass from MASS,UNSY	UMAS
MASS	Mass from MASS,CONS or MASS	MASS
DAMP	Damping from DAMP,CONS or DAMP	DAMP
UDAM	Damping from DAMP,UNSY	UDAM
DISP	Solution values for nodal displacements	
VELO	Solution values for nodal velocities	
ACCE	Solution values for nodal accelerations	

Table 14.1: Output array options for OUTPut command.

14.1.24 PARAmeter command

The use of parameters in data input items is discussed in Sect. 4.2. Parameters may be changed during solution using the PARAmeter command. The value of a single parameter may be set using the solution command

```
PARAmeter parameter_name parameter_expression
```

where parameter_name is any parameter name and parameter_expression as described in Sect. 4.2 but here is limited to 15 characters. For example the use of the command as

```
PARAmeter a a+2
```

or

```
PARAmeter a=a+2
```

increments the parameter a by 2.

For setting multiple values the use of the PARAmeter command without a parameter name requires additional data to define the parameters and their values. The data is given after the batch END command. Thus this form is:

```
BATCh
  ...
  PARAmeter
  ...
END
  .... ! Parameter expressions
  LIST ! List parameters in output file
```

Care in this form must be taken to ensure that other solution commands do not also require data after the `END` command.

14.1.25 `PERIodic` response

For composite materials it is often desirable to compute average properties from a representative volume element (RVE). In *FEAP* the basic process is carried out by a Hill-Mandel average.[17,21,62] For this case a typical RVE may be defined and subjected to either displacement or periodic boundary conditions where the boundaries of the RVE are subjected to a homogeneous strain (for small deformations) or a deformation gradient (for finite deformations). The specification of the loading is given during mesh input by a `PERIodic` data set (see Sect. 5.5.7).

The computation of the averaged properties is determined using solution command set

```
LOOP,,n_its
  TANGent,,1
NEXT
HILL <TANGent,      >
```

to obtain the homogenized stress and/or thermal flux and the associated tangent array or

```
LOOP,,n_its
  TANGent,,1
NEXT
HILL STREss
```

to obtain stress and/or thermal flux averages only. The properties for non-linear problems may also be obtained, however, in this case it is necessary to define time variations to the deformation by a proportional loading.

The boundary of the RVE must be either fully restrained (boundary fixed) without any other non-zero boundary conditions or may be a periodic boundary in which the region is a rectangle or brick in which there are nodes on opposite faces with one or for 3-d two coordinates identical. In the periodic case the boundaries may be *linked* using the `ELINk` command.

In addition to overall average stress, the average of stress (and strain) over each material may be computed using the command

```
STREss AVERage
```

The contours of the RVE displacements may be plotted as either total ones or just the fluctuation values from the homogeneous state. The latter are displayed using the plot command

```
PLOT UHILl dof lines mesh
```

where `dof` is the component of the displacement, `lines` is the number of contour lines if positive or filled otherwise, and `mesh` superposes the mesh if zero. For further details on plotting contours see Section 15.2.

File inputs

The solution for transient problems in which gradients are provided in a separate file is performed as

```
LOOP,,ntimes
  HILL READ
  LOOP,,nits
    TANG,,1
  NEXT
  HILL TANGent
NEXT
HILL CLOSe
```

14.1.26 PLOT command

The issuing of the command `PLOT` from an interactive mode causes input to be requested for plot commands. In a batch mode the specific command to be used for a plot item is given in the second through fifth fields as:

```
PLOT type v1 v2 v3
```

where `type` defines the plot command and `vi` the parameters needed to perform the plot.

14.1.27 PRINt command

The command

```
PRINt
```

is used to restart solution command outputs following use of a `NOPRint` command. Alternatively, the command

```
PRINt OFF
```

is used to discontinue (nearly) all output from solution command executions.

Other options for using the print command include:

```
PRINt array
```

where `array` may have any of the values:

```
CMAS LMAS GEOM TANG UTAN RESI
```

to output the diagonal of the consistent mass, lumped mass, geometric stiffness, symmetric tangent, unsymmetric tangent or residual, respectively.

14.1.28 `PROPoritional` load command

The solution of transient problems often requires the specification of loads which may be expressed in the form

$$\mathbf{f}_I(t) = p_j(t)\,\mathbf{F}_I$$

where $\mathbf{F}_I$ are the nodal forces specified by a mesh `FORCe`, `EFORce`, or `CFORce` command and p_j is the j^th time dependent proportional load. The input of a proportional load is given in batch mode using the command set

```
BATCh
  PROP,,j
END
  .... ! Data for j-th prop-load
```

There are several basic options to input the data for the j^th load. A blank record defines a *ramp loading* with unit slope for positive times.

A *Type 1* proportional load may be given using the record (after the batch `END` statement)

```
1 k tmin tmax a1 a2 a3 a4 a5
```

and yields the proportional load

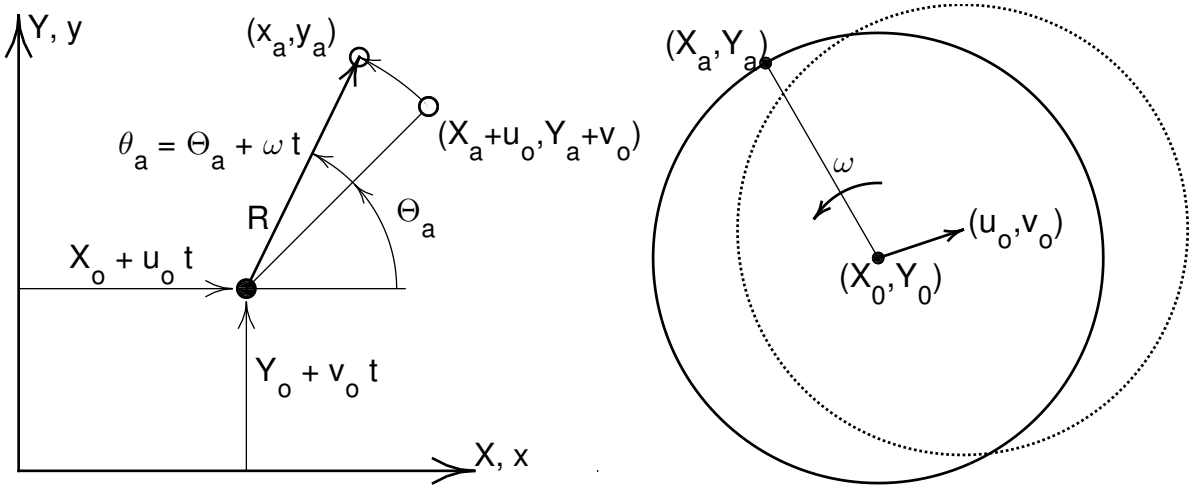
$$p_j(t) = a_1 + a_2\,(t - t_{min}) + a_3\,[\sin(a_4(t - t_{min}) + a_5)]^k$$

which remains valid only in the time range $t_{min} \le t \le t_{max}$. Note that the arguments of the *sine* function are given in *radians* not *degrees*.

For *Type 2* proportional loading a table input may be input using the command set:

```
BATCh
  PROP,,j
END
  2  np  ! Type 2 loading with np (t,p) pairs/input record
  t1,p1 t2,p2 ... t-np,v-np
        ! Blank termination record
```

where `np` defines the number of time-value pairs to appear on each record. By default `np = 1`. Piecewise linear interpolation is used to evaluate $p_j(t)$ between the times $t_i \le t \le t_i + 1$. Alternatively, a piecewise linear velocity time function may be specified in the same manner by using a *Type 7* (instead of 2) for the table specification. In this case the piecewise linear velocity is integrated to give a quadratic variation between the specified time values.



(a) Geometry of point. (b) Sliding of circle.

Figure 14.1: Type 3 Proportional loading: Translation and rotation in $x - y$ plane.

For *Type 3* proportional loading is a translation and rotation in the $x - y$ plane as shown in Fig. 14.1. Restrained nodal positions in the reference configuration X_a, Y_a have displacements

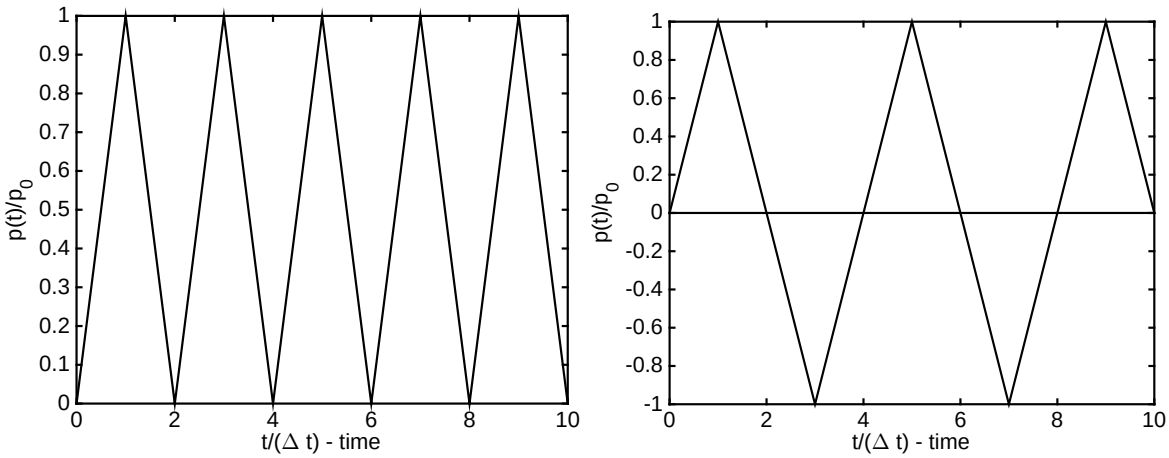
$$
\begin{aligned}
u_a(t) &= u_o\,\Delta t + R_a\,[\cos(\Theta_a + \omega_o\,\Delta t)\cos\Theta_a] \\
v_a(t) &= v_o\,\Delta t + R_a\,[\sin(\Theta_a + \omega_o\,\Delta t)\sin\Theta_a] \\
\Theta_a &= \tan^{-1}\left(\frac{Y_a - Y_o}{X_a - X_o}\right) \\
R_a &= [(X_a - X_o)^2 + (Y_a - Y_o)^2]^{1/2}
\end{aligned}
$$

where $\Delta t = t - t_{min}$. The data is input as

```
BATCh
  PROP,,j
END
3  t_min t_max  ap_1 ap_2 ap_3 ap_4 ap_5
```

where ap_1 $= u_o$, ap_2 $= v_o$, ap_3 $= \omega_o$, ap_4 $= X_o$ and ap_5 $= Y_o$. If the problem is in three dimensions $w_a(t) = 0$.

For the case of rolling without slip up a ramp at the bottom of the circle shown in Fig. 14.1(b) the angular velocity ω would be negative.



(a) $m = 1$ Type (b) $m = 2$ Type

Figure 14.2: Type 4 Proportional loading: Sawtooth.

Type 4 proportional loading is a periodic 'sawtooth' type loading and is specified as

```
BATCh
  PROP,,j
END
4  m  t_min t_max  p_0  Delta_t
```

The value of `m` may be 1 or 2, `p_0` is the amplitude and `Delta_t` the period (viz. Fig. 14.2).

Type 5 proportional loading is a polynomial loading and is specified as

```
BATCh
  PROP,,j
END
5  t_min t_max  p_0  p_1 p_2 p_3 p_4
```

The function is evaluated as

$$p(t) = p_0 + \Delta t\,(p_1 + \Delta t\,(p_2 + \Delta t\,(p_3 + \Delta t\,p_4)))$$

where $\Delta t = t - t_{min}$.

Type 6 proportional loading is an exponential and periodic loading and is specified as

```
BATCh
  PROP,,j
END
6 L t_min t_max  p_0  p_1 d_1 p_2 d_2
```

The function is evaluated as

$$p(t) = p_0 + p_1 \exp(d_1 \Delta t) + p_2 (\sin(d_2 \Delta t))^L$$

where $\Delta t = t - t_{min}$.

Type 7 proportional loading specifies the rate of loading (i.e., a velocity) from which the displacement is computed by integrating piecewise linear inputs. The input format is identical to *Type 2* loading except for the type number. Thus, an input may be specified as

```
BATCh
  PROP,,j  ! Prop load "j" -- give numeric or parameter value
END
  7  np  ! Type 2 loading with np (t,p) pairs/input record
  t1,p1 t2,p2 ... t-np,v-np
        ! Blank termination record
```

in which `np` pairs of time and value are input. Integration is carried out analytically from the piecewise linear segments to create smooth quadratic variations of displacement. *It is the user responsibility to check the value of displacements obtained such that desired behavior is achieved.*

See Section 14.8.1 for additional information on using proportional loading functions in analyses. Appendix D gives options to specify a user proportional loading function.

14.1.29 `REACtion` command

During the solution of all finite element problems by *FEAP* a set of residuals is computed. The values of these residuals, including all the restrained degree-of-freedoms may be output using the `REACtion` command. The numerical values for all the nodal reaction or residuals at each node may be output using the command:

```
REACtion ALL
```

Alternatively, the values for a few nodes may be given as

```
REACtion nodes n1 n2 inc
```

in which `n1` and `n2` define the first and last node to output and `inc` is the increment between node numbers.

It is possible to also use the `LIST` command (see above) to describe a set of nodes for which outputs are requested and then use the commands

```
REACtion LIST l1
```

to output the values. Here `l1` is the list number.

The values for a constant value of the nodal coordinates may be used to output a set of values. The command is given as

```
REACtion COORdinate d x tol
```

where `d` is the coordinate direction (e.g., 1,2 or 3), `x` is the value of the coordinate in the d-direction and `tol` is a search tolerance to find nodes.

The current reactions may may be saved to a file using the command

```
REACtion FILE
```

The initial file has a name `Rxxxx.reac0000`, where `xxxx` is extracted from the *save* file name. Subsequent use of the command increases the counter (the file is not destroyed after completion of the analysis, consequently, if the problem is rerun it is necessary to manually delete the file beforehand. This file may be input as nodal loading, for example if boundary conditions are changed from fixed to active, using the **mesh** command

```
REAC Rxxxx.reac0000
```

or using whichever file is wanted.

14.1.30 `SHOW` command: Viewing solution data

The `SHOW` command permits users to display the problem size and values for some of the solution parameters, include file contents, as well as to check the amount of data stored in arrays allocated in the solution space. The command is given as

```
SHOW option v1 v2
```

where `option` can have the values `COMM`on, `CONT`act, `DICT`ionary, `ELEM`ent, `INCL`ude, `PART`ition, *array name*, or be omitted. When omitted the `SHOW` command displays values for basic solution state and mesh parameters. Use of the `DICT`ionary option displays the names, type, and size for all arrays currently allocated in the solution space. Values stored in each array may be displayed by using the name as the *array name* option. If the array is large the `vi` parameters can be used to limit the amount of information displayed. For example use of the command:

```
SHOW X
```

would display to the screen (in interactive mode) and to the output file all the values of nodal coordinates for the problem. Alternatively,

```
SHOW X 21 40
```

would display the values stored in locations 21 to 40. If the problem is two dimensional the coordinate array is dimensioned

```
REAL (KIND=8) :: X(2,*)
```

and thus, position 21 corresponds to the 1-direction (x) for node 11 and position 40 to the 2-direction (y) for node 20.

Using the command `SHOW PART` displays the current partition data for solver type, number of equations, time integration method, and solution flags.

The use of `SHOW ELEM` displays a list of the currently loaded user elements, that is the routines that have names `elmt01` to `elmt50`. Similarly, the use of `SHOE CONT` displays the list user contact formulations that are loaded.

The use of `SHOW COMM` or `SHOW INCL` initiates a request to input the name of `include` files (these are the files contained in the `./include` and `./include/integer<4,8>` directories. For example, entering the name `cdata` displaces the current values for the number of nodal points, `numnp`, the number of elements, `numel`, the number of material sets, `nummat`, the maximum number of nodes on any element, `nen`, the number of active solution variables, `neq`, and the precision ratio between *real* and *integer* variables, `ipr`. This feature is primarily for use during development of new modules when need for detailed knowledge of current solution valuables is useful.

14.1.31 `SOLVe` command

The `SOLVe` command is used after a tangent matrix and a residual are computed to solve the set of equations

$$\mathbf{K}_T \, d\mathbf{u} = \mathbf{R}(\mathbf{u})$$

for the solution increments $d\mathbf{u}$. The command is given as

```
SOLVe
```

and results in the current solution state $\mathbf{u}$ being updated as

$$\mathbf{u} \leftarrow \mathbf{u} + d\mathbf{u} \quad .$$

For some problems it is not possible to fully update the residual without causing a divergence in the solution iterates. In this case a *line search* strategy[63] may be invoked using the solution option

```
SOLVe LINE val
```

where `val` is a parameter whose default value is set at 0.8. In this case a scaling parameter s is computed and updates are performed as

$$\mathbf{u} \leftarrow \mathbf{u} + s \, d\mathbf{u} \quad .$$

The use of line search is highly recommended for problems with inelastic material behavior (e.g., plasticity problems).

14.1.32 STREss command

The (generalized) stress state in each element may be output using the command

```
STREss ALL
```

Alternatively, the output for a limited number of elements may be performed using the command

```
STREss,,n1,n2,inc
```

where `n1` is an initial element, `n2` is a final element and `inc` and increment to use in selecting subsequent elements for output.

Use of the form

```
STREss NODE n1 n2 inc
```

performs a projection of stresses (and in some cases, strains) onto nodes and outputs the values for nodes `n1` to `n2` at increments of `inc`. For solid elements the first 6 values are stresses and the next 6 are strains. Printing may be truncated if components are zero. By default stresses between different material sets are allowed to be discontinuous. If it is desired to have them continuous the solution command

```
PROJect CONTinuous
```

should be given before issuing the `STREss NODE` command. A command

```
PROJect
```

will restore discontinuous projections

The projected values at a constant value of the nodal coordinates may be used to output a set of values. The command is given as

```
STREss COORdinate d x tol
```

where `d` is the coordinate direction (e.g., 1,2 or 3), `x` is the value of the coordinate in the d-direction and `tol` is a search tolerance to find nodes.

It is possible to also compute averages of stress and strain over each material in meshes of solid elements. This is given by

```
STREss AVERage
```

14.1.33 SUBSpace command

The subspace command may be used to solve the generalized linear eigen-problem

$$\mathbf{K}_T \, \mathbf{\Phi} = \mathbf{B} \, \mathbf{\Phi} \, \mathbf{\Lambda}$$

where $\mathbf{\Phi}$ are eigenvectors and $\mathbf{\Lambda}$ are eigenvalues. In the above $\mathbf{K}_T$ must be a symmetric tangent stiffness matrix and $\mathbf{B}$ must be a symmetric matrix. Three options exist for solution: $\mathbf{B} = \mathbf{M}$ the mass matrix; $\mathbf{B} = \mathbf{K}_G$ the geometric stiffness matrix and $\mathbf{B} = \mathbf{I}$ an identity matrix.

The command is given as:

```
SUBSpace PRINt n1 g1
```

where the parameter `PRINt`, if present, results in an output of the subspace arrays, `n1` is the number of converged pairs requested and `g1` is the number of extra (guard) vectors to use for acceleration of the convergence. The minimum required form is

```
SUBSpace,,n1
```

and *must* follow a `TANG` command and a `MASS`, `GEOM` or `IDEN` command. When `g1` is not explicitly specified, *FEAP* sets it to the maximum of `n1` or `8`. The subspace algorithm is a method for which the rate of convergence is controlled by the ratio of

$$\kappa = \frac{\lambda_{nl}}{\lambda_{nl+gl}} \tag{14.1}$$

and thus the smaller this ratio the more rapid the convergence. If the matrix $\mathbf{K}_T$ is singular the subspace method will not converge or will produce an error (see Sec. 14.5.1below to solve this case).

`ARPACK` command

While the subspace method can solve the generalized linear eigenproblem, there are more efficient methods based on the Arnoldi/Lanczos method. The ARPACK library[64] contains algorithms to compute the eigen-pairs for a variety of symmetric and unsymmetric real and complex problems. The current version of *FEAP* is provided with an interface to the package as an extension of the basic program. The module may be found in the subdirectory: `packages/arpack`.

If the package is loaded an eigen-solution for the symmetric real linear dynamics problem may be computed using the command:

```
ARPAck <SYMMetric,LUMP> n1 g1
```

The `SYMMetric` form may be used with either a *lumped* or a *consistent* mass matrix. The `LUMP` form may only be used with problems with a *lumped* mass where all diagonal terms are positive (thus, it may not be used with beam problems in which no rotational lumped mass is present).

14.1.34 `TANGent` matrix command

The `TANGent` command is used to form *symmetric* tangent arrays for use in an iterative solution strategy. The command given as:

```
TANGent
```

results in the computation of a tangent array $\mathbf{K}_T$ for the current solution state $\mathbf{u}$. When direct solution by a Gauss elimination algorithm is used the tangent matrix also is factored (without pivoting) into the form

$$\mathbf{K}_T = \mathbf{U}^T \mathbf{D}\, \mathbf{U}$$

where $\mathbf{D}$ is a diagonal matrix and $\mathbf{U}$ an upper triangular matrix.

Issuing the command as

```
TANGent,,1
```

is equivalent to the command set

```
TANGent
FORM
SOLVe
```

and results in a solution step and update as

$$\mathbf{K}_T d\mathbf{u} = \mathbf{U}^T \mathbf{D}\, \mathbf{U}\, d\mathbf{u} = \mathbf{R}$$
$$\mathbf{u} \leftarrow \mathbf{u} + d\mathbf{u}$$

A line search may be added (see `SOLVe` above) using the command

```
TANGent LINE 1,,val
```

where `val` is the line search parameter and has a default value of 0.8. This modifies the update step to

$$\mathbf{u} \leftarrow \mathbf{u} + s\, d\mathbf{u}$$

where s is an adjusted step size from the line search algorithm.

In computing the solution to eigen-problems in which zero eigen-values exist (e.g., the $\mathbf{K}_T$ has zero eigenvalues) it is necessary to specify a shift and compute the factorization of

$$\mathbf{K}_T - \alpha\, \mathbf{B} = \mathbf{U}^T \mathbf{D}\, \mathbf{U}$$

where α is the shift value. The general form then is given as:

```
TANGent LINE 1 alpha val
```

When transient solutions are computed *FEAP* computes a modified tangent matrix during the time integration process. In general cases the resulting tangent matrix is

$$\mathbf{K}_T^\star = c_1\,\mathbf{K}_T + c_2\,\mathbf{C} + c_3\,\mathbf{M}$$

where $\mathbf{C}$ is a damping matrix and c_i are parameters from the time integration procedure. Performing an eigen-solution to compute the vibration modes will cause an error unless only the true tangent matrix $\mathbf{K}_T$ is used. This may be computed using the command form:

```
TANGent EIGV 0 alpha
```

Note that it is still possible to use the shift α to avoid singular arrays (e.g., problems where rigid body modes are not fully prevented, see Sec. 14.5.1).

14.1.35 TIME command

The time command is used to advance the time where a solution is sought, i.e.,

$$t_{n+1} = t_n + \Delta t$$

where the value of Δt is set using a `DT` command as described above. The command is given as

```
TIME,,t-max
```

where `t-max` defines a solution time at which solution may cease. If `t-max = 0` it is set to infinity and solution continues to the maximum number of time steps requested.

Setting initial time

It is also possible to use the command to set the initial time where solution is to start. The command is then given as

```
TIME SET t-new
```

and results in the value of the discrete time to be set as

$$t_n = t_{new} \quad .$$

Explicit solutions: Critical time step size

In transient solutions computed using an *explicit* time stepping method it is necessary to keep the size of the time step within the Courant-Fredrick-Levy (CFL) limit. *FEAP* can compute the critical time step using material parameters and minimum nodal spacing to estimate the Δt_{cr} of the CFL condition. Using the command

```
TIME EXPLicit t-max c
```

the time step for each solution is taken as

$$\Delta t = c * \Delta t_{cr}$$

and solution continues until either the number of time steps specified on the `LOOP` command or the time `t-max` is reached. It is recommended that c be set smaller than 1.0 – the default is 0.9. This command should only be used with an explicit solution of problems with solid elements. See Sect. 14.1.38 for setting an explicit solution option.

14.1.36 `TOL` command

The basic `TOL` command is used to set the solution tolerances used to detect convergence in algorithms which use a `LOOP-NEXT` pair as

```
LOOP,,nits
  ...
  TANG,,1
  ...
NEXT
```

Looping will terminate either when `nits` iterations is performed or convergence of the solution is achieved to within a specified tolerance. The primary check by *FEAP* uses an energy convergence criterion

$$E_i = d\mathbf{u}^T\mathbf{R} \leq tol\, E_1$$

as a basis of detecting convergence. Thus when the energy of an increment becomes less than `tol` times the original (first) value in the step convergence is assumed. Some secondary checks are also made to ensure that the first residual computed has not become so small as to be meaningless. For example, this occurs under constant loading of a viscoelastic material when it approaches an long time equilibrium stage.

The basic tolerance command is given by

```
TOL,,tol_e tol_r
```

where `tol_e` is the energy convergence tolerance value and `tol_r` the residual value. It is not necessary to use this command unless the default value of 10^{-16} needs to be changed. By default the residual convergence tolerance is

$$tol_r = 100\,(tol_e)^{1/2}$$

Iterative solution of linear equations

The `TOL` command may also be used to set the precision in iterative solution of a linear algebraic equation set in the form

$$\mathbf{r} = \mathbf{R} - \mathbf{K}\, d\mathbf{u} = \mathbf{0}$$

In this case the command is given by

```
TOL ITERation rtol
```

or

```
ITERation TOL rtol
```

where `rtol` denotes convergence for $\mathbf{r}_i$ to a specified tolerance as

$$(\mathbf{r}_j^T \mathbf{r}_j)^{1/2} \leq rtol\, (\mathbf{r}_0^T \mathbf{r}_0)^{1/2}$$

The default value of `rtol` is 10^{-8}.

14.1.37 `TPLOt` command

In solution of transient problems it is often necessary to plot the time evolution for quantities at specified locations in the finite element mesh. The command `TPLOt` is used to save specified information and is given by the form:

```
BATCh
  TPLOt,,int
END
 type n1 n2       ! Tplot data point form 1
         or
 type ,, n2 x y z ! Tplot data point form 2
   ....
SHOW/NOSHow       ! Tplot end
```

where `type` options and parameters are shown in Table 14.2. The parameter `n1` specifies a node or element number for output, `n2` is the value number (e.g., a degree of freedom of a node, or a stress component in an element, etc.) and `x, y, z` is the coordinate locations where an output is needed. In `type` options which permit either `n1` or `x, y, z`, only one may be selected. Thus if `n1` is specified no `x, y, z` is given and vice versa. The value of z is not needed for two-dimensional problems.

Example: Save displacement for degree of freedom 2 at node 5

```
BATCh
  TPLOT
END
DISP 5 2
SHOW
```

Type	No.	Pt.	Coord.	Description
DISP	n1	n2	x, y, z	Displacement component $n2$ for node $n1$ or coordinate point x, y, z
VELO	n1	n2	x, y, z	Velocity component $n2$ for node $n1$ or coordinate point x, y, z
ACCE	n1	n2	x, y, z	Acceleration component $n2$ for node $n1$ or coordinate point x, y, z
REAC	n1	n2	x, y, z	Reaction component $n2$ for node $n1$ or coordinate point x, y, z
RSUM	n1	n2	n3	Reaction component $n1$ sum on nodes $n2$ to $n3$
SUMS	n1	n2	x, tol	Reaction component $n1$ sum on coordinate component $n2$ with value x (tolerance = tol)
STRE or ELEM	n1	n2	x, y, z	Stress component $n1$ for element $n2$ or coordinate point x, y, z
HIST	n1	n2	x, y, z	History variable $n1$ for element $n2$ or coordinate point x, y, z
CONT	n1	n2	x, y, z	Contact pair $n1$, variable $n2$
CHIS	n1	n2	x, y, z	Contact history for pair $n1$, slave node number $n2$, variable $n3$
ARCL	n1	n2		Arclength value
ENER	n1			Energy and momenta: n1=1,2,3 linear momenta, n1=4,5,6 angular momenta; n1=7 kinetic energy, n1=8 stored energy, n1=9 external work, n1=10 total energy, n1=11 angular momentum norm, n1=0 all.
USER	n1	n2	x, y, z	User option

Table 14.2: Time history plot options

Example: Save stress component 5 at point (5, 3.2) in a 2-d problem

```
BATCh
  TPLOT
END
STREss ,,5 5.0 3.2
SHOW
```

Here the component number is for the element which contains the point. Care should be taken to confirm that the correct element is used (e.g., check the printed output for the element number selected).

14.1.38 TRANsient command

The solution of time dependent problems in solid and structural mechanics where inertial effects are important may be performed in *FEAP* using one-step time integration methods. Similarly in transient heat conduction and other first order ordinary differential equation problems the solution may be obtained using one-step methods. The first class results in need to solve problems where the residual is given by

$$\mathbf{R}(\mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}) = \mathbf{0}$$

whereas the second class is given by

$$\mathbf{R}(\mathbf{u}, \dot{\mathbf{u}}) = \mathbf{0}$$

It is possible to solve problems in which both forms are active, however, for these the `ORDEr` command must be used to define the order for each degree of freedom (see, Sect. 14.4.4).

The initiation of a transient solution is specified by including the solution command

```
TRANsient type p1 p2 p3
```

where `type` is the integration type and p_i are solution parameters. Generally, it is not necessary to specify values for p_i unless changes from default values are needed. Table

Method	Description	ODE Order
`OFF`	Quasi-static method	0
`STATic`	Midpoint static method	0
`BACKward`	Backward Euler implicit method	1
`BDF2`	Backward difference formula method	1
`FORWard`	Forward Euler explicit method	1
`EXPLicit`	Forward Euler explicit method, 1 Requires: ORDEr set to 1	
`GEN1`	Generalized midpoint method	1
`NEWMark`	Newmark method	2 or 1
`ALPHa`	Generalized alpha-method (HHT)	2
`HHT`	HHT alpha-method	2
`CONServe`	Energy-momentum conserving method	2
`EXPLicit`	Newmark explicit method ($\beta = 0$)	2
`CENTral`	Central difference explicit method	2
`EULEr`	Euler implicit method	2
`USER`	User programmed method	-

Table 14.3: Transient integration methods in *FEAP*

14.3 defines all the algorithms currently available. As noted in the table it is possible for users to add their own algorithm (consult the Programmer Manual for details).

Care should be taken when requesting an integration method for use in problems where material behavior is non-linear or time dependent. The material models included with *FEAP* have not been coded to work with any time integration method where the residual is computed at an intermediate point between the specified discrete times t_n. The algorithms `OFF` (default), `BACK`, `NEWMark`, `EXPLicit`, `EULEr` and `CENTral` should work with all material models. The remaining methods work only for elastic and thermal models. Also when explicit methods are employed with in-elastic material behavior the material set data must contain the command

```
STARt INELastic
```

in order to activate all terms in the model.

For more information on each transient algorithm see `TRAN` in Appendix D, page 563.

14.1.39 `UTANgent` matrix command

The `UTANgent` command is used to form *unsymmetric* tangent arrays for use in an iterative solution strategy. The command given as:

```
UTANGent
```

results in the computation of a tangent array $\mathbf{K}_T$ for the current solution state $\mathbf{u}$. When direct solution by a Gauss elimination algorithm is used the tangent matrix also is factored (without pivoting) into the form

$$\mathbf{K}_T = \mathbf{U}^T \mathbf{D}\, \mathbf{U}$$

where $\mathbf{D}$ is a diagonal matrix and $\mathbf{U}$ an upper triangular matrix.

Issuing the command as

```
UTANGent,,1
```

is equivalent to the command set

```
UTANGent
FORM
SOLVe
```

and results in a solution step and update as

$$\mathbf{K}_T d\mathbf{u} = \mathbf{U}^T \mathbf{D}\, \mathbf{U}\, d\mathbf{u} = \mathbf{R}$$
$$\mathbf{u} \leftarrow \mathbf{u} + d\mathbf{u}$$

A line search may be added (see `SOLVe` above) using the command

```
UTANGent LINE 1,,val
```

where `val` is the line search parameter and has a default value of 0.8. This modifies the update step to

$$\mathbf{u} \leftarrow \mathbf{u} + s\, d\mathbf{u}$$

where s is an adjusted step size from the line search algorithm.

Currently, it is not possible to use unsymmetric tangents in the solution of an eigen-problem. It is possible to extend the program to this class of problems using the ARPACK library.[64] This package also has capabilities to use the Lanczos algorithm to produce faster eigen-pair extractions for symmetric problems.

14.1.40 `VELOcity` command

For transient problems using solid or structural elements the solution is given in terms of displacements, velocities and accelerations at each solution time t_n. Similarly for transient heat conduction problems the solution is given in terms of temperature (the generalized displacement) and rates (velocities) at each t_n. The numerical values of the velocity at each node may be given using the command:

```
VELOcity ALL
```

Alternatively, the values for a few nodes may be given as

```
VELOcity nodes n1 n2 inc
```

in which `n1` and `n2` define the first and last node to output and `inc` is the increment between node numbers.

It is possible to also use the `LIST` command (see below) to describe a set of nodes for which outputs are requested and then use the commands

```
VELOcity LIST l1
```

to output the values. Here `l1` is the list number.

Finally, the values for a constant value of the nodal coordinates may be used to output a set of values. The command is given as

```
VELOcity COORdinate d x tol
```

where `d` is the coordinate direction (e.g., 1,2 or 3), `x` is the value of the coordinate in the d-direction and `tol` is a search tolerance to find node

14.2 RESTart and SAVE commands

During solution of problems in which many steps are involved it is often desirable to *save* the solution state so that the analysis can later be continued from that point instead of again from the initial state. This is called a *restart* analysis.

A restart file is *saved* using the command

```
SAVE,,<0,k>
```

where `k` is an optional counter ranging from "1" to "999" that may be used to define a specific solution state or step. The solution step will be added as an extender to the restart file name set when starting *FEAP* (see Sect. 2.1) and yields a form

```
Rxxxx_00k
```

in the serial version. The form in the parallel version is

```
Rxxxx_000p00k
```

where "p" is the processor number.[3] A subsequent analysis may then be performed by using the **same** data input file but with the first solution command given as:

```
RESTart,,k
```

to indicate the desired step to use. If `k` is omitted the step "001" will be used.[4] After a restart the analysis may be continued using standard solution commands.

14.3 Problem solving

Each problem is solved by using a set of the command language statements which together form the *algorithm* defining the particular solution method employed. The commands to solve a linear static problem are:

```
BATCh                !initiate batch execution
  TANG               !form tangent matrix
  FORM               !form residual
  SOLVe              !solve equations
  DISPlacement,ALL   !output all displacements
  STREss,ALL         !output all element stresses
  REACtion,ALL       !output all nodal reactions.
END                  !end of batch program
```

The command sequence

[3] The form shown is for cases where the processor and step are less than 10.

[4] In parallel solutions the command follows the PETSc command.

```
TANG
FORM
SOLVe
```

is the basic solution step in *FEAP* and for simplicity (and efficiency) may be replaced by the single command

```
TANG,,1
```

This single statement is more efficient in numerical operations since it involves only a single process to compute all the finite element arrays, whereas the three statement form requires one for `TANG` and a second for `FORM`. Thus,

```
BATCh                !initiate batch execution
  TANG,,1            !form and solve
  DISPlacement ALL   !output all displacements
  STREss ALL         !output all element stresses
  REACtion ALL       !output all nodal reactions.
END                  !end of batch program
```

is the preferred solution form. Some problems have tangent matrices which are unsymmetric. For these situations the `TANG`ent command should be replaced by the `UTAN`gent command. The statements `DISP`lacement, `STRE`ss, and `REAC`tion control information which is written to the output file and to the screen. The commands `PRIN`t and `NOPR`int may be used to control or prevent information appearing on the screen - information always goes to the output file. Printing to the screen is the default mode. See Appendix D for the options to control the displacement, stress, and reaction outputs.

Additional commands may be added to the program given above. For example, inserting the following command after the solution step (i.e., the `TANG,,1` command) will produce a screen plot of the mesh:

```
PLOT MESH            !plot mesh
```

Further discussion for plotting is given in Chapter 15.

14.3.1 Solution of non-linear problems

The solution of non-linear problems is often performed using Newton's method which solves the problem

$$\mathbf{R}(\mathbf{u}) = \mathbf{0} \tag{14.2}$$

using the iterative algorithm

1. Set initial solution

$$\mathbf{u}^0 = \mathbf{0} \ . \tag{14.3}$$

2. Solve the set of equations

$$\mathbf{K}\,\Delta\mathbf{u}^i = \mathbf{R}(\mathbf{u}^i) \tag{14.4}$$

where

$$\mathbf{K} = -\frac{\partial \mathbf{R}}{\partial \mathbf{u}} \ . \tag{14.5}$$

3. Update the solution iterate

$$\mathbf{u}^{i+1} = \mathbf{u}^i + \Delta\mathbf{u}^i \ . \tag{14.6}$$

The steps are repeated until a norm of the solution is less than some tolerance.

FEAP implements the Newton algorithm using the following commands:

```
LOOP iter 10      !perform up to 10 Newton iterations
  TANG,,1         !form tangent, residual and solve
NEXT iter         !proceed to next iteration
```

The tolerance used for controlling the solution is

$$E^i = \Delta\mathbf{u}^i \cdot \mathbf{R}^i \tag{14.7}$$

with convergence assumed when

$$E^i < tol\, E^0 \ . \tag{14.8}$$

The value of the tolerance is set using the `TOL` command (default is 10^{-12}).

While the sample above specifies 10 iterations, fewer will be used if convergence is achieved. Convergence is tested during the `TANG,,1` command. If convergence is achieved, *FEAP* transfers to the statement following the `NEXT` command. If convergence is not achieved in 10 iterations, *FEAP* exits the loop, prints a `NO CONVERGENCE` warning, and continues with the next statement. For the algorithm given above, the only difference between a converged and non-converged exit from the loop is the number of iterations used. However, if there are commands inserted between the `TANG` and `NEXT` statements they are not processed for the iteration in which convergence is achieved. Obviously, solutions which do not converge during a time step may produce inaccurate results in the later solution steps. Consequently, users should check the output log of non-linear solutions for any `NO CONVERGENCE` records.

Remarks:

1. Blank characters before the first character in a command are ignored by *FEAP*, thus, the indenting of statements shown is optional but provides for clarification of key parts in the algorithm.

2. In the above loop command the `ITER` in the second field is given to provide clarity. This is optional; the field may be left blank.

By replacing the Newton steps

```
LOOP iter 10
  TANG,,1
NEXT iter
```

with

```
TANG              !form tangent only
LOOP iter 10      !perform 10 modified Newton iterations
  FORM            !form residual
  SOLVe           !solve linearized equations
NEXT iter         !proceed to next iteration
```

a modified Newton algorithm results. The modified Newton method forms only one tangent and each iteration is performed by computing and solving the residual equation with the same tangent. When *FEAP* forms the tangent while in a direct solution of equations mode the triangular factors are also computed so that the `SOLVe` only performs re-solutions during each iteration. While a modified Newton method involves fewer computations during each iteration it often requires substantially more iterations to achieve a converged solution. Indeed, if the tangent matrix is an accurate linearization for the non-linear equations, the asymptotic rate of convergence for a Newton method is quadratic, whereas a modified Newton method is often only linear (if the residual equation set is linear the tangent matrix is constant and both the Newton and modified Newton methods should converge after one iteration, that is, iteration two should produce a residual which is zero to within the computer precision).

The *FEAP* command language is capable of defining a large number of standard algorithms. Each user is urged to carefully study the complete set of available commands and the options available for each command. In order to experiment with the capabilities of the language, it is suggested that small problems be set up to test any proposed command language program and to ensure that the desired result is obtained.

14.3.2 Solution of linear algebraic equations

Direct solution of linear algebraic equations

The use of Newton's method results in a set of linear algebraic equations which are solved for the incremental solution values. *FEAP* includes several options to solve linear equations. The default solution scheme is a variable band, profile scheme discussed in Chapter 20 of *The Finite Element Method, Vol 1*, 5th edition[19] (or in Chapter 15 of

The Finite Element Method, Vol 1, 4th edition[20]). This solution scheme may be used to solve problems where the incremental displacements are either in real arithmetic or in complex arithmetic. The coefficient matrix of the linear equations results in large storage requirements within the computer memory. A profile optimization scheme is available to renumber the equations in an attempt to minimize this storage. The solution command `OPTI`mize may be used to perform the profile minimization. A summary of the results is given and may be compared to that without optimization. If necessary, the optimization may be omitted using the command `OPTI,OFF`. The default solution is without optimization.

For problems in which the memory requirements exceed that which is available to the program, there are alternatives which require reduced amounts of storage. The alternatives are available for problems in real arithmetic only. For problems with symmetric coefficient arrays (i.e., those for which the command `TANG`ent is used to form the array), a sparse solver may be used. The sparse solver is activated by issuing the solution command

```
DIREct SPARse
```

before the first use of the `TANG`ent command. *WARNING: If the sparse solver requires more memory than available, the program may 'crash' with no error message printed in a file or to the screen.* Alternatively, the profile solution scheme may be employed with a blocking scheme used to retain unneeded parts of the coefficient array during the solution process. This option is may be requested using the command

```
DIREct BLOCk
```

before giving any `TANG` or `UTAN` command. This solution scheme proceeds by writing blocks of the columns for the upper profile of the matrix into files named `AUPPER1`, `AUPPER2`, etc. and blocks of rows for the lower profile of the matrix into files `ALOWER1`, `ALOWER2`, etc. There must be sufficient free disk capacity on the computer to store the total coefficient array. The speed of solution is reduced using this option by the need to write and read data from the hard disk drive. The blocked solution scheme may be used for either symmetric or unsymmetric coefficient arrays.

FEAP has also been interfaced to the serial form of the SuperLU, UMFPACK, WSMP and Pardiso equation solvers. Information on downloading an interface is available at `projects.ce.berkeley.edu/feap`.

Iterative solution of linear algebraic equations

The final option available is an iterative, preconditioned conjugate gradient scheme (PCG method). The PCG method is applicable to symmetric, positive definite coefficient arrays only. Thus, only the `TANG`ent command may be used. The PCG with preconditioning using the diagonal of the matrix is requested by the command

```
ITERation
```

before the first TANGent command. A block nodal preconditioner may be requested using the command

```
ITERation BPCG
```

Experience to date suggests the iteration method is effective and efficient only for three dimensional linear elastic solids problems. Success has been achieved when the solids are directly connected to shells and beams; however, use with thin shells has resulted in very slow convergence - rendering the method ineffective. Use with non-linear material models (e.g., plasticity) has not been successful in static problem applications. Use of the PCG method in dynamics improves the situation if a mass term is available for each degree of freedom (i.e., lumped mass on frames with no rotational mass will probably not be efficient).

14.4 Transient Solutions

FEAP provides several alternatives to construct transient solutions. To solve a non-linear time dependent problem using Newton's method with an *implicit* time integration method (*i.e.*, requiring solution of a set of algebraic equations) the following commands may be issued:

```
DT,,0.01           !set time increment to 0.01
TRANsient method   !specify "method" for time stepping
LOOP time 12       !perform 12 time steps
  TIME             !advance time by 'dt' (i.e., 0.01)
  LOOP iter 10     !perform up to 10 Newton iterations
    TANG,,1        !form tangent, residual and solve
  NEXT iter        !proceed to next iteration
  DISP,,1 12       !report displacements at nodes 1-12
  STRE NODE 1 12   !report stresses at nodes 1-12
NEXT time          !proceed to next time step
```

In addition to output for DISPlacement, transient algorithms permit the output of VELOcity and ACCEleration (see Appendix D). Care should be taken with printed output as it can lead to very large output files. One can minimize the output to occur every few steps using an extra loop as

```
DT,,0.01           !set time increment to 0.01
TRANsient method   !specify "method" for time stepping
LOOP print 20      !
  LOOP time 10     !perform 10 time steps
    TIME           !advance time by 'dt' (i.e., 0.01)
```

```
    LOOP iter 10    !perform up to 10 Newton iterations
      TANG,,1       !form tangent, residual and solve
    NEXT iter       !proceed to next iteration
  NEXT time         !proceed to next time step
  DISP,,1 12        !report displacements at nodes 1-12
  STRE NODE 1 12    !report stresses at nodes 1-12
NEXT print          !proceed to next print sequence
```

The above set of command language statements will perform 200 time steps but only 20 output sets.

FEAP provides several alternatives to construct transient solutions. A transient solution is performed by giving the solution command language statement

```
TRANsient method
```

The type of transient solution to be performed depends on the *method* option specified. *FEAP* solves three basic types of transient formulations: (1) Quasi-static solutions; (2) First order ordinary differential equation solutions - ODE1; and (3) Second order ordinary differential equation solutions - ODE2. Table 14.3 shows the methods and solution order of the differential equation to be solved for each method available in *FEAP*.

14.4.1 Quasi-static solutions

The governing equation to be solved by the quasi-static option is expressed as:

$$\mathbf{R}(t) \;=\; \mathbf{F}(t) \;-\; \mathbf{P}(\mathbf{u}(t)) \;=\; \mathbf{0} \tag{14.9}$$

where, for example, the $\mathbf{P}$ vector is given by the stress divergence term of a solid mechanics problem as:

$$\mathbf{P}(\mathbf{u}(t)) \;=\; \mathbf{P}_\sigma \;=\; \int_\Omega \mathbf{B}^T \boldsymbol{\sigma}\, dV \;. \tag{14.10}$$

The solution options for this form are:

1. The default algorithm which solves

$$\mathbf{R}(t_{n+1}) \;=\; \mathbf{F}(t_{n+1}) \;-\; \mathbf{P}(\mathbf{u}(t_{n+1})) \;=\; \mathbf{0} \tag{14.11}$$

   using the commands

```
LOOP time nstep
  TIME
  LOOP Newton niters
```

```
      TANG,,1
    NEXT
    ... Outputs
  NEXT
```

The default option does not require a TRANsient command; however it may also be specified using the command

```
TRANsient OFF
```

2. Quasi-static solutions may also be solved using a generalized midpoint configuration for the residual equation. This option is specified by the command

   ```
   TRANsient STATic alpha
   ```

   and solves the equation

$$\mathbf{R}(t_{n+\alpha}) \;=\; \mathbf{F}(t_{n+\alpha}) \;-\; \mathbf{P}(\mathbf{u}(t_{n+\alpha})) \;=\; \mathbf{0} \tag{14.12}$$

   where

$$\mathbf{u}(t_{n+\alpha}) \;=\; \mathbf{u}_{n+\alpha} \;=\; (1-\alpha)\,\mathbf{u}_n \;+\; \alpha\,\mathbf{u}_{n+1} \tag{14.13}$$

   and

$$\mathbf{F}(t_{n+\alpha}) \;=\; \mathbf{F}_{n+\alpha} \;=\; (1-\alpha)\,\mathbf{F}_n \;+\; \alpha\,\mathbf{F}_{n+1}\;. \tag{14.14}$$

   The parameter α must be greater than zero; the default value is 0.5. Setting α to 1 should produce answers identical to those from option 1. The transient option to be used must be given prior to specifying the time loop and solution commands shown above.

14.4.2 First order transient solutions

The governing equation to be solved for first order transient solutions is expressed as:

$$\mathbf{R}(t) \;=\; \mathbf{F}(t) \;-\; \mathbf{P}(\mathbf{u}(t), \dot{\mathbf{u}}(t)) \;=\; \mathbf{0} \tag{14.15}$$

where, for example, $\mathbf{u}$ are the nodal temperatures $\mathbf{T}$ and the $\mathbf{P}$ vector is given by:

$$\mathbf{P} \;=\; \int_\Omega (\nabla N)^T\,\mathbf{q}\,dV \;+\; \mathbf{C}\,\dot{\mathbf{T}} \tag{14.16}$$

with $\mathbf{C}$ the heat capacity matrix.

The solution options for this form are:

1. A backward Euler method which solves the problem

$$\mathbf{R}(t_{n+1}) \;=\; \mathbf{F}(t_{n+1}) \;-\; \mathbf{P}(\mathbf{u}(t_{n+1}), \dot{\mathbf{u}}_{n+1}(t)) \;=\; \mathbf{0} \tag{14.17}$$

   where

$$\dot{\mathbf{u}}_{n+1} \;=\; \frac{1}{\Delta t}[\mathbf{u}_{n+1} \;-\; \mathbf{u}_n]\;. \tag{14.18}$$

   The command:

```
TRANsient BACK
```

is used to specify this solution option.

2. A generalized midpoint method which solves the problem

$$\mathbf{R}(t_{n+\alpha}) \;=\; \mathbf{F}(t_{n+\alpha}) \;-\; \mathbf{P}(\mathbf{u}(t_{n+\alpha}), \dot{\mathbf{u}}_{n+\alpha}(t)) \;=\; \mathbf{0} \tag{14.19}$$

where

$$\dot{\mathbf{u}}_{n+\alpha} \;=\; \frac{1}{\Delta t}[\mathbf{u}_{n+1} \;-\; \mathbf{u}_n] \;. \tag{14.20}$$

This solution option is selected using the command

```
TRANsient GEN1 alpha
```

where $0 < \alpha \leq 1$ (default is 0.5); $\alpha = 1$ should produce answers identical to those from the backward Euler option.

3. A second order backward Euler formula method which solves the problem

$$\mathbf{R}(t_{n+1}) \;=\; \mathbf{F}(t_{n+1}) \;-\; \mathbf{P}(\mathbf{u}(t_{n+1}), \dot{\mathbf{u}}_{n+1}(t)) \;=\; \mathbf{0} \tag{14.21}$$

where

$$\dot{\mathbf{u}}_{n+1} \;=\; \alpha\, \mathbf{v}_{n+1} + (1-\alpha)\, \mathbf{v}_n \tag{14.22}$$

with

$$\begin{aligned} \mathbf{v}_k &= \frac{\mathbf{u}_k \;-\; \mathbf{u}_{k-1}}{\Delta t_k} \quad ; \quad k = n, n+1 \\ \Delta t_k &= t_k - t_{k-1} \end{aligned} \tag{14.23}$$

and

$$\alpha = \frac{2\,\Delta t_{n+1} + \Delta t_n}{\Delta t_{n+1} + \Delta t_n} \tag{14.24}$$

The command:

```
TRANsient BDF2
```

is used to specify this solution option.

14.4.3 Second order transient solutions

The governing equation to be solved for second order transient solutions is expressed as:

$$\mathbf{R}(t) \;=\; \mathbf{F}(t) \;-\; \mathbf{P}(\mathbf{u}(t), \dot{\mathbf{u}}(t), \ddot{\mathbf{u}}(t)) \;=\; \mathbf{0} \tag{14.25}$$

where, for example, the $\mathbf{P}$ vector is given by:

$$\mathbf{P} \;=\; \mathbf{P}_\sigma \;+\; \mathbf{C}\,\dot{\mathbf{u}} \;+\; \mathbf{M}\,\ddot{\mathbf{u}} \tag{14.26}$$

with $\mathbf{C}$ the damping and $\mathbf{M}$ the mass matrix.

The solution options for second order problems are:

1. A Newmark method[65] which solves the problem

$$\mathbf{R}(t_{n+1}) \;=\; \mathbf{F}(t_{n+1}) \;-\; \mathbf{P}(\mathbf{u}_{n+1}, \mathbf{v}_{n+1}, \mathbf{a}_{n+1}) \;=\; \mathbf{0} \tag{14.27}$$

where

$$\mathbf{v}_n \;=\; \dot{\mathbf{u}}_n \quad ; \quad \mathbf{a}_n \;=\; \ddot{\mathbf{u}}_n \tag{14.28}$$

with updates computed as:

$$\mathbf{u}_{n+1} \;=\; \mathbf{u}_n \;+\; \Delta t\, \mathbf{v}_n \;+\; \Delta t^2 \left[\, (0.5 - \beta)\, \mathbf{a}_n \;+\; \beta\, \mathbf{a}_{n+1}, \; \right] \tag{14.29}$$

and

$$\mathbf{v}_{n+1} \;=\; \mathbf{v}_n \;+\; \Delta t \left[\, (1 - \gamma)\, \mathbf{a}_n \;+\; \gamma\, \mathbf{a}_{n+1} \right] \tag{14.30}$$

in which β and γ are parameters controlling stability and numerical dissipation. The command

```
TRANsient NEWMark beta gamma idyn0
```

is used to select this integration scheme. Optionally, the command

```
TRANsient,,beta gamma idyn0
```

also selects the Newmark algorithm. The parameter `idyn0` defines the starting condition for the first iteration in non-linear solutions, options are describe in Table 14.4. Default values are $\beta = 0.25$, $\gamma = 0.5$ and `idyn0 = 1`. The use of other options results in different *predictor* values for the solution process as described in the book by Géradin & Rixen.[66]

The second order problem using the Newmark method may require special care in computing the initial state if non-zero initial conditions or loading terms exist. To compute the initial state it is necessary to first compute a mass matrix and then the initial accelerations. The commands are

```
TRANsient NEWMark
INITial (DISPlacements and/or RATEs)
FORM ACCEleration
LOOP time nstep
  TIME
  LOOP Newton niters
```

IDYN0	Starting condition
1	$\mathbf{u}^0_{n+1} = \mathbf{u}_n$
2	$\mathbf{v}^0_{n+1} = \mathbf{v}_n$
3	$\mathbf{a}^0_{n+1} = \mathbf{0}$
4	$\mathbf{a}^0_{n+1} = \mathbf{a}_n$

Table 14.4: Starting conditions for Newmark method

```
      TANG,,1
    NEXT
    ... Outputs
NEXT
```

It is also necessary to use this sequence for the following method. If $\mathbf{F}(0)$, $\mathbf{u}(0)$, and $\mathbf{v}(0)$ are zero, the FORM, ACCEleration command should be omitted to conserve memory resources.

In the above the setting of any non-zero initial displacements or rates may be specified using the INITial command. The initial command requires additional data which in a BATCh solution option appears immediately after the END command. In an interactive mode a user receives a prompt to specify the data.

2. A Hilber-Hughes-Taylor (HHT) method[67] which solves the problem

$$\mathbf{R}(t_{n+\alpha}) \;=\; \mathbf{F}(t_{n+\alpha}) \;-\; \mathbf{P}(\mathbf{u}_{n+\alpha}, \mathbf{v}_{n+\alpha}, \mathbf{a}_{n+\alpha}) \;=\; \mathbf{0} \tag{14.31}$$

where

$$\mathbf{u}_{n+\alpha} \;=\; (1-\alpha)\,\mathbf{u}_n \;+\; \alpha\,\mathbf{u}_{n+1} \tag{14.32a}$$

$$\mathbf{v}_{n+\alpha} \;=\; (1-\alpha)\,\mathbf{v}_n \;+\; \alpha\,\mathbf{v}_{n+1} \tag{14.32b}$$

$$\mathbf{a}_{n+\alpha} \;=\; \mathbf{a}_{n+1}\;. \tag{14.32c}$$

The displacement and velocity quantities at t_{n+1} are updated using the Newmark formulas given above. This solution option is selected using the command

```
TRANsient ALPHa beta gamma alpha
```

The alpha parameter should be specified between zero and 1. Default values are $\beta = 0.5$, $\gamma = 1$, and $\alpha = 0.5$. The parameter α differs from the relation for α_H in reference[67] which required negative values. The two are related through

$$\alpha = 1 + \alpha_H$$

3. The HHT method may also be input in the simpler form

```
TRANsient HHT alpha
```

with $0.5 \le \alpha \le 1$ and the parameters β and γ will be automatically computed to produce the algorithm given in reference:[67]

$$\beta = \tfrac{1}{4}\,(2-\alpha)^2 \quad \text{and} \quad \gamma = \tfrac{3}{2} - \alpha$$

4. An Euler implicit solution with the acceleration given as:

$$\mathbf{a}_{n+1} = \frac{1}{\Delta t}\,(\mathbf{v}_{n+1} - \mathbf{v}_n) \tag{14.33}$$

and

$$\mathbf{u}_{n+1} \;=\; \mathbf{u}_n \;+\; \Delta t\,\mathbf{v}_{n+1} \tag{14.34}$$

This solution option is selected using the command

```
TRANsient EULEr
```

Since (14.34) does not involve the acceleration it also may be used to integrate a first order transient problem. This permits the integrator to be used for problems in which some equations are second order and others are first order (e.g., transient thermo-mechanical problems). It should be noted however that the integrators are only first order accurate and possess considerable numerical dissipation if the time steps are too large.

5. An energy conserving form of the alpha method[39,68,69] (i.e., similar to the HHT method) with the acceleration given as:

$$\mathbf{a}_{n+\alpha} \;=\; \frac{1}{\Delta t}\left[\,\mathbf{v}_{n+1} \;-\; \mathbf{v}_n\,\right]. \tag{14.35}$$

This solution option is selected using the command

```
TRANsient CONServe beta gamma alpha
```

The alpha parameter should be specified between zero and 1. Default values are $\beta = 0.5$, $\gamma = 1$, and $\alpha = 0.5$. Note that the conserving form does not involve the accelerations in the equations of motion (only displacement and velocity); consequently, it is not necessary to compute initial accelerations as in the Newmark and HHT methods. For linear problems the conserving method gives identical results (except for accelerations) as the Newmark method; however, the parameters to achieve the equality are different. Default parameters should achieve equality provided Newmark is started by accounting for any non-zero accelerations at time zero.

WARNING: The conserving algorithm should not be used with inelastic materials or elements which do not support the form.

6. An explicit solution to the equations

$$\mathbf{R}(t_{n+1}) \;=\; \mathbf{F}(t_{n+1}) \;-\; \mathbf{P}_\sigma(\mathbf{u}_{n+1},\mathbf{v}_{n+1}) \;-\; \mathbf{M}\,\mathbf{a}_{n+1} \;=\; \mathbf{0} \tag{14.36}$$

which uses the Newmark formulas with $\beta = 0$ and specifies *gamma* by the command

```
TRANsient EXPLicit gamma
```

The gamma parameter should be greater or equal to 0.5, the default is $\gamma = 0.5$. A solution using the explicit option uses the command sequence:

```
TRANsient EXPLicit
INITial (DISPlacements and/or RATEs)
MASS LUMP          (form lumped mass array)
FORM ACCElerations (initial acceleration)
DT,,dt             (set time step size)
LOOP time nstep
  TIME EXPLicit    (Controls critical DT)
  FORM EXPLicit
  ... Outputs
NEXT time
```

Often an initial step may produce a zero residual and this causes *FEAP* to exit the loop in which the computation is made. This may be avoided by replacing the above algorithm by

```
TRANsient EXPLicit
INITial (DISPlacements and/or RATEs)
MASS LUMP          (form lumped mass array)
FORM ACCElerations (initial acceleration)
DT,,dt             (set time step size)
LOOP time nstep
  TIME EXPLicit    (Controls critical DT)
  LOOP,,1
    FORM EXPLicit
  NEXT loop
  ... Outputs
NEXT time
```

IMPORTANT: If inelastic constitution is used the Mesh Input Command: `STARt INELastic` **must be specified as part of the** MATErial **set for each inelastic constitutive model. That is, insert**

```
STARt INELastic
```

for each inelastic model. Failure to do this will result in an elastic only solution. (See Sect. 7.5).

7. A central difference explicit solution to the equations

$$\mathbf{R}(t_{n+1}) \;=\; \mathbf{F}(t_{n+1}) \;-\; \mathbf{P}_\sigma(\mathbf{u}_{n+1}, \mathbf{v}_{n+1}) \;-\; \mathbf{M}\,\mathbf{a}_{n+1} \;=\; \mathbf{0} \tag{14.37}$$

uses a centered finite difference to compute the acceleration as

$$\mathbf{a}_{n+1} = \frac{\mathbf{u}_{n+2} - \mathbf{u}_{n+1}}{\Delta t_m \, \Delta t_{n+2}} - \frac{\mathbf{u}_{n+1} - \mathbf{u}_n}{\Delta t_m \, \Delta t_{n+1}} \tag{14.38}$$

where

$$\Delta t_m = \frac{1}{2}\left(\Delta t_{n+1} + \Delta t_{n+2}\right) \quad .$$

The command

```
TRANsient CENTral
```

requests the central difference method. To keep the solution process trivial for diagonal mass existing damping effects use the velocity approximation

$$\mathbf{v}_{n+1} = \frac{u_{n+1} - u_n}{\Delta t_{n+1}} \quad .$$

IMPORTANT: If inelastic constitution is used the Mesh Input Command: `STARt INELastic` **must be specified as part of the** MATErial **set for each inelastic constitutive model. Failure to do this will result in an elastic only solution.** (See Sect. 7.5).

FEAP permits the type of transient problem to be changed during the solution phase. Thus, it is possible to compute a configuration using a quasi-static option and then change to a solution mode which includes the effects of rate terms (e.g., inertial effects).

14.4.4 Mixed first and second order transient solutions

Many problems involve the solution of transient forms in which some equations are first order and some are second order. In this case it is necessary to use the `ORDEr` command between the mesh input and the transient solution algorithm. The basic form of the mesh input is then given by (see Sect. 11.5):

```
FEAP * * ...
  ...
END

ORDEr
  o1 o2 o3 .. o-ndf

BATCh
  TRANs NEWMark (example)
  ....
END
```

where `o1` to `o-ndf` define the order of the transient problem for each degree of freedom. In solving such problems the transient integrator specified by the `TRANs` command *must* be one of the forms defined above for second order problems. Each of these integrators also has a form set for quasi-static (zero order) and first order forms.

Care should be taken when selecting the form to compute the tangent matrix as most of the mixed form problems have *unsymmetric* tangent matrices. Thus, one should usually select the `UTAN` command to form and solve the linear equations. An exception exists however for thermo-mechanical problems where coupling occurs only from the thermal expansion strain $\boldsymbol{\alpha}\,\Delta T$ (see Sect. 7.1). The matrix form for a coupled linear thermo-mechanical problem is given by

$$\begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}} \\ \ddot{\mathbf{T}} \end{Bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{H} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{T}} \end{Bmatrix} + \begin{bmatrix} \mathbf{K}_u & \mathbf{C} \\ \mathbf{0} & \mathbf{K}_t \end{bmatrix} \begin{Bmatrix} \mathbf{u} \\ \mathbf{T} \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_u \\ \mathbf{f}_t \end{Bmatrix}$$

where $\mathbf{M}$ is the mass matrix; $\mathbf{H}$ is the specific heat matrix; $\mathbf{K}_u$ is the stiffness matrix; $\mathbf{K}_t$ is the conductivity matrix; $\mathbf{C}$ is the thermal coupling matrix; and $\mathbf{f}_u$, $\mathbf{f}_t$ are loading terms. Here a solution may be achieved using a symmetric solution and multiple iterations for each step. An appropriate algorithm is as above with the batch solution given by

```
BATCh
```

```
  TRANs NEWMark
  DT,,dt
  LOOP time nt
    TIME
    TANG
    LOOP,,3
      FORM
      SOLVe
    NEXT
      ... Outputs
  NEXT time
END
```

Convergence will be obtained for a linear coupled problem only after the second iteration – that is, the *third residual* will be numerically zero. Replacing `TANG` by `UTAN` in the above will yield a numerically converged solution in the second iteration.

For the symmetric solution form a more efficient solution may be obtained using partitions (see Sect. 11.4) in which the thermal solution alone is obtained first and then followed by the solution of the mechanical problem. For the linear problem both will converge in one iteration, with the second iteration giving a numerically zero value.

14.4.5 Explicit/implicit solutions

In some problems it is possible to integrate the equations using a split into implicit integrated and explicit integrated parts. This is especially effective when a lumped mass is used for the explicit part since then no simultaneous equations are solved. *FEAP* permits element sets to be defined for this type of partition by including the following commands immediately after the mesh is defined:

```
BATCh
  MESH
END
EXPLicit
  MATE m1 <i1>
  MATE m2 <i2>
   .etc.
    ! Terminate with blank record
END ! mesh modification
```

The `m1, m2` are material set numbers and the `i1, i2` are extra iterations allowed for the explicit part. In some cases it may be desirable to have a consistent mass when computing the residual to increase accuracy in the solution. Generally, `i1, i2` may be omitted as noted by the `<,,>` brackets.

14.5 Transient solution of linear problems

The solution of second order linear equations by the finite element method leads to the set of equations

$$\mathbf{M}\,\ddot{\mathbf{u}}(t) \;+\; \mathbf{C}\,\dot{\mathbf{u}}(t) \;+\; \mathbf{K}\,\mathbf{u}(t) \;=\; \mathbf{F}(t)\;. \tag{14.39}$$

In structural dynamics, the matrices $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ denote *mass*, *damping*, and *stiffness*, respectively. The vector $\mathbf{F}$ is a force vector. For the case where $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ are constant *symmetric* matrices a solution to Eq. 14.39 may be constructed by partitioning the solution into the parts

$$\mathbf{u} \;=\; \begin{bmatrix}\mathbf{u}_u\\ \mathbf{u}_s\end{bmatrix} \tag{14.40}$$

where $(\cdot)_u$ denotes an unknown part and $(\cdot)_s$ a specified part. With this division, the equations are then written in the form:

$$\begin{bmatrix}\mathbf{M}_{uu} & \mathbf{M}_{us}\\ \mathbf{M}_{su} & \mathbf{M}_{ss}\end{bmatrix}\begin{bmatrix}\ddot{\mathbf{u}}_u\\ \ddot{\mathbf{u}}_s\end{bmatrix} + \begin{bmatrix}\mathbf{C}_{uu} & \mathbf{C}_{us}\\ \mathbf{C}_{su} & \mathbf{C}_{ss}\end{bmatrix}\begin{bmatrix}\dot{\mathbf{u}}_u\\ \dot{\mathbf{u}}_s\end{bmatrix} + \begin{bmatrix}\mathbf{K}_{uu} & \mathbf{K}_{us}\\ \mathbf{K}_{su} & \mathbf{K}_{ss}\end{bmatrix}\begin{bmatrix}\mathbf{u}_u\\ \mathbf{u}_s\end{bmatrix} \;=\; \begin{bmatrix}\mathbf{F}_u\\ \mathbf{F}_s\end{bmatrix}\;. \tag{14.41}$$

A solution is then constructed by first solving the first row of these equations. The value of the reactions (i.e., $\mathbf{F}_s$) associated with the known part of the solution $\mathbf{u}_s$ may be computed later if it is needed. The solution of the first row of these equations may be constructed by several approaches. The equations may be integrated in time directly using a numerical step-by-step procedure (e.g., the Newmark method); the solution may be constructed using normal modes and if necessary specified multiple support conditions added; the equations may be solved in the frequency domain. In the next sections we discuss a solution using modal methods and in a subsequent section a solution for the a response due to periodic excitations is presented.

14.5.1 Normal mode solution

The normal modes are obtained by assuming the vector $\mathbf{u}_s$ is zero and setting

$$\mathbf{u}_u = \boldsymbol{\phi}_j\,\exp(i\,\omega_j\,t) \tag{14.42}$$

where $\boldsymbol{\Phi}_j$ is a *mode shape* and ω_j is its associated *natural frequency*. Differentiating with respect to time leads to the problem

$$\left[-\,\omega_j^2\,\mathbf{M}_{uu} \;+\; i\,\omega_j\,\mathbf{C}_{uu} \;+\; \mathbf{K}_{uu}\right]\,\boldsymbol{\phi}_j\,\exp(i\,\omega_j\,t) \;=\; \mathbf{F}_u \tag{14.43}$$

in which $i = \sqrt{-1}$. The normal modes of free vibration then may be obtained by setting the force vector $\mathbf{F}_u$ to zero and, for the present, ignoring the damping matrix $\mathbf{C}_{uu}$. For this case the problem reduces to:

$$\left[-\,\omega_j^2\,\mathbf{M}_{uu} \;+\; \mathbf{K}_{uu}\right]\,\boldsymbol{\phi}_j \;=\; \mathbf{0} \tag{14.44}$$

which may be solved as the general linear eigen-problem

$$[\mathbf{K}_{uu}]\,[\mathbf{\Phi}] \;=\; [\mathbf{M}_{uu}]\,[\mathbf{\Phi}]\,[\mathbf{\Lambda}] \tag{14.45}$$

where

$$[\mathbf{\Phi}] \;=\; \begin{bmatrix}\boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_n\end{bmatrix} \tag{14.46}$$

is the set of *normal modes* and

$$[\mathbf{\Lambda}] \;=\; \begin{bmatrix}\omega_1^2 & 0 & \cdots & 0\\ 0 & \omega_2^2 & \cdots & 0\\ 0 & 0 & \ddots & 0\\ 0 & \cdots & 0 & \omega_n^2\end{bmatrix} \tag{14.47}$$

is a diagonal matrix of the *natural frequencies squared*.

The solution for the normal modes are normalized so that

$$\mathbf{\Phi}^T\mathbf{M}_{uu}\mathbf{\Phi} \;=\; \mathbf{I} \tag{14.48}$$

and

$$\mathbf{\Phi}^T\mathbf{K}_{uu}\mathbf{\Phi} \;=\; \mathbf{\Lambda}\;. \tag{14.49}$$

In *FEAP* the solution for part (and for small problems all) of the normal modes may be obtained using a *subspace iteration* method and the solution commands:

```
MASS
TANGent
SUBSpace,,nf
```

where `nf` is the number of modes to compute. Additional parameters may be given to use a lumped (diagonal) mass, to specify a shift on the tangent matrix, and/or to improve the convergence properties of the subspace method (See Appendix D for specifying additional options).

The eigen-pair for the `nf` modes computed above may be output using the command

```
EIGVector ALL k1
```

where `k1` is the value of the mode (between `1` and `nf`) to be placed in the output file (See Appendix D for specifying additional options).

If the modes for an *unsupported structure* (which will contain zero eigen-values) are desired, the tangent matrix is *singular* and the subspace method (and many other methods) will fail to converge or an error may result during the construction of the triangular factors of the $\mathbf{K}$ matrix. In this case a *shift* may be used where the frequencies squared are given as

$$\omega_j^2 \;=\; \tilde{\omega}_j^2 \,+\, \chi\;. \tag{14.50}$$

Now the general linear eigen-problem is given by:

$$([\mathbf{K}_{uu}] - \chi\,[\mathbf{M}_{uu}])\,[\mathbf{\Phi}] = [\mathbf{M}_{uu}]\,[\mathbf{\Phi}]\left[\tilde{\mathbf{\Lambda}}\right] \tag{14.51}$$

which may be solved using the command language algorithm

```
MASS
TANGent,,,chi
SUBSpace,,nf
```

in which `chi` denotes the value of χ in Eq. 14.50. In this case the subspace algorithm computes the `nf` eigen-values closest to χ. The choice for χ affects the accuracy and convergence properties of the subspace algorithm. For the case where $\mathbf{K}_{uu}$ is positive semi-definite (i.e., its eigen-values are all zero or positive). It is best to first use a *negative* value for `chi`. The value should be sufficiently negative to prevent a poor factorization of $\mathbf{K}_{uu}$.With this value compute the eigen-solution for `nf` set to the number of zero eigen-values plus 2. The zero eigen-values should be very small compared to the first non-zero value. Now set the value of `chi` to about one-half the value of the first non-zero value and compute the number of desired values. Using this value should produce a good rate of convergence and good accuracy.

Damping effects

The effects of damping may be included in the modal formulation and still retain real normal modes by assuming a damping matrix in the form

$$\mathbf{C} = a_0\,\mathbf{M} + a_1\,\mathbf{K}\,. \tag{14.52}$$

This defines a form called *Rayleigh Damping*. With this form the Damping matrix has the property:

$$\begin{aligned}\mathbf{\Phi}^T\mathbf{C}_{uu}\mathbf{\Phi} &= a_0\,\mathbf{I} + a_1\,\mathbf{\Lambda} \\ &= 2\,\boldsymbol{\zeta}\,\mathbf{\Lambda}^{\frac{1}{2}}\end{aligned} \tag{14.53}$$

where $\boldsymbol{\zeta}$ is a diagonal matrix of damping ratios. The damping ratio may be related to the parameters a_0 and a_1 as

$$2\,\boldsymbol{\zeta} = a_0\,\mathbf{\Lambda}^{\frac{1}{2}} + a_1\,\mathbf{\Lambda}^{-\frac{1}{2}}\,. \tag{14.54}$$

Values for the parameters a_0 and a_1 may be computed from two values of frequencies where a specified damping ratio ζ is desired. If the two frequencies are denoted by ω_i and ω_j the parameters are given by

$$a_0 = 2\,\zeta\frac{\omega_i\,\omega_j}{\omega_i + \omega_j} \tag{14.55}$$

and

$$a_1 \;=\; \frac{2\,\zeta}{\omega_i + \omega_j}\;. \tag{14.56}$$

Other values may also be selected. For further information consult "Dynamics of Structures", by A.K. Chopra.[70] The data input to *FEAP* is given by the command

```
RAYLeigh,,a-0 a-1
```

Rayleigh damping may also be included in transient problems solved by time integration methods. In this case the damping matrix may be specified independently for each material as global or material parameters (see Section 7.15).

Solution of transient problems

Using normal modes the solution of the transient problem is constructed by substituting the solution

$$\mathbf{u}_u(t) \;=\; \mathbf{\Phi}\,\mathbf{v}(t) \tag{14.57}$$

into the first row of Eq. 14.41 and premultiplying by $\mathbf{\Phi}^T$. Using the orthogonality properties from Eqs. 14.48, 14.49, and 14.53 the result is given by:

$$\ddot{\mathbf{v}} \;+\; 2\,\boldsymbol{\zeta}\,\mathbf{\Lambda}^{\frac{1}{2}}\,\dot{\mathbf{v}} \;+\; \mathbf{\Lambda}\,\mathbf{v} \;=\; \mathbf{\Phi}^T\mathbf{F}_u(t) \;=\; \mathbf{G}(t) \tag{14.58}$$

which is a set of uncoupled second order differential equations. An individual equation is given by:

$$\ddot{v}_j \;+\; 2\,\zeta_j\,\omega_j\,\dot{v}_j \;+\; \omega_j^2\,v_j \;=\; g_j(t)\;. \tag{14.59}$$

Each of the equations may be integrated numerically or, if the loading is assumed in some functional form, exactly. For example, assuming piecewise linear variation in a time step *FEAP* performs an exact integral provided support solutions all have zero values. The numerical solution using modal methods is given by the command language algorithm

```
DT,,dt-value
LOOP time n-steps
  TIME
  MODAl
   .....  ! outputs/plots
NEXT time
```

If desired the nodal and element values may be output using the `DISP` and `STREss` commands, respectively. Also graphical display of the solution may be obtained using the standard `PLOT` options.

Specified multiple support excitation

In the previous sections the modal response was constructed by assuming all specified support locations had zero values. A solution to Eq. 14.41 which includes the effects of non-zero support excitations may be constructed by expressing the solution in the form:

$$\begin{bmatrix}\mathbf{u}_u\\ \mathbf{u}_s\end{bmatrix} \;=\; \begin{bmatrix}\boldsymbol{\Phi}\\ \mathbf{0}\end{bmatrix}\,[\mathbf{v}] \;+\; \begin{bmatrix}\boldsymbol{\Psi}\\ \mathbf{I}\end{bmatrix}\,[\mathbf{w}] \tag{14.60}$$

where the arrays $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ represent the normal modes of vibration and *static* modes to satisfy non-zero specified boundary conditions, respectively. For the static modes we solve the problem

$$\mathbf{K}_{uu}\,\boldsymbol{\Psi} \;+\; \mathbf{K}_{us}\,\mathbf{I} \;=\; \mathbf{0}\;. \tag{14.61}$$

The solution for the normal modes is obtained from Eq. 14.45.

Once these modes are known, the first row of Eq. 14.41 may be premultiplied by $\boldsymbol{\Phi}^T$ to give

$$\begin{aligned}\boldsymbol{\Phi}^T\mathbf{M}_{uu}\boldsymbol{\Phi}\,\ddot{\mathbf{v}} \;+\; \boldsymbol{\Phi}^T\mathbf{M}_{uu}\boldsymbol{\Phi}\,\dot{\mathbf{v}} \;+\; \boldsymbol{\Phi}^T\mathbf{K}_{uu}\boldsymbol{\Phi}\,\mathbf{v}&\\ = \mathbf{G} \;-\; \boldsymbol{\Phi}^T\left[\mathbf{M}_{uu}\boldsymbol{\Psi}+\mathbf{M}_{us}\right]\,\ddot{\mathbf{w}} \;-\; \boldsymbol{\Phi}^T\left[\mathbf{C}_{uu}\boldsymbol{\Psi}+\mathbf{C}_{us}\right]\,\dot{\mathbf{w}}\;.&\end{aligned} \tag{14.62}$$

Invoking the orthogonality conditions Eqs. 14.48, 14.49, and 14.53 leads to the set of decoupled equations

$$\ddot{\mathbf{v}} \;+\; 2\,\boldsymbol{\zeta}\,\boldsymbol{\Lambda}^{\frac{1}{2}}\,\dot{\mathbf{v}} \;+\; \boldsymbol{\Lambda}\,\mathbf{v} \;=\; \boldsymbol{\Phi}^T\mathbf{F}_u(t) \;=\; \mathbf{G}(t) \;-\; \mathbf{A}_1\,\ddot{\mathbf{w}} \;-\; \mathbf{A}_2\,\dot{\mathbf{w}} \tag{14.63}$$

where

$$\mathbf{A}_1 \;=\; \boldsymbol{\Phi}^T\left[\mathbf{M}_{uu}\boldsymbol{\Psi} \;+\; \mathbf{M}_{us}\right] \tag{14.64}$$

and

$$\mathbf{A}_2 \;=\; \boldsymbol{\Phi}^T\left[\mathbf{C}_{uu}\boldsymbol{\Psi} \;+\; \mathbf{C}_{us}\right]\;. \tag{14.65}$$

For Rayleigh damping only one matrix is required since

$$\mathbf{A}_2 \;=\; a_0\,\mathbf{A}_2\;. \tag{14.66}$$

In *FEAP* these equations are integrated using the energy momentum method in which the discrete time values are given as

$$\mathbf{v}_n \;\approx\; \mathbf{v}(t_n) \tag{14.67}$$

and the solution advanced using the equations:

$$\mathbf{v}_{n+1} \;=\; \mathbf{v}_n + \Delta t\,\dot{\mathbf{v}}_n + \frac{1}{2}\Delta t^2\,\ddot{\mathbf{v}}_{n+\frac{1}{2}} \tag{14.68}$$

and

$$\dot{\mathbf{v}}_{n+1} \;=\; \dot{\mathbf{v}}_n + \Delta t\, \ddot{\mathbf{v}}_{n+\frac{1}{2}} \;. \tag{14.69}$$

Values at the mid time step $t_{n+\frac{1}{2}}$ are computed as:

$$\mathbf{v}_{n+\frac{1}{2}} \;=\; \frac{1}{2}\left(\mathbf{v}_n + \mathbf{v}_{n+1}\right) \tag{14.70}$$

$$\dot{\mathbf{v}}_{n+\frac{1}{2}} \;=\; \frac{1}{2}\left(\dot{\mathbf{v}}_n + \dot{\mathbf{v}}_{n+1}\right) \tag{14.71}$$

and

$$\ddot{\mathbf{v}}_{n+\frac{1}{2}} \;=\; \frac{1}{\Delta t}\left(\dot{\mathbf{v}}_{n+1} \;-\; \dot{\mathbf{v}}_n\right) \;. \tag{14.72}$$

Finally, the equations of motion are written at the mid step giving:

$$\ddot{\mathbf{v}}_{n+\frac{1}{2}} \;+\; 2\,\boldsymbol{\zeta}\,\boldsymbol{\Lambda}^{\frac{1}{2}}\,\dot{\mathbf{v}}_{n+\frac{1}{2}} \;+\; \boldsymbol{\Lambda}\,\mathbf{v}_{n+\frac{1}{2}} \;=\; \mathbf{G}_{n+\frac{1}{2}} \;-\; \mathbf{A}_1\,\ddot{\mathbf{w}}_{n+\frac{1}{2}} \;-\; \mathbf{A}_2\,\dot{\mathbf{w}}_{n+\frac{1}{2}} \;. \tag{14.73}$$

The values of the time derivatives for $\mathbf{w}_{n+\frac{1}{2}}$ are determined from the inputs of $\mathbf{w}_n$ using Eqs. 14.68 to 14.72.

The specification of the data for a problem which is to be subjected to multiple support excitations requires the following data and solution steps:

1. During mesh input, specify the base patterns and their associated proportional loading factors. Base patterns are given by the mesh `BASE` command with data for each node given as follows

```
BASE
 node1 gen1 (base-set1(i),i=1,ndf)
 node2 gen2 (base-set2(i),i=1,ndf)
   etc. for additional nodes
                                    ! Blank terminator record
```

   In the above non-zero `base-set(i)` values define the individual base set numbers. A zero value indicates the degree-of-freedom is assigned to the unknown part of a solution vector. Base sets should be numbered from one (1) to a maximum number.

   Recall that it is also necessary to assign each node with a non-zero base set to a specific proportional load set using the `FPROportional` mesh command. For example, this may be done using the data set:

```
FPROportional
 node1 gen1 (prop-set1(i),i=1,ndf)
 node2 gen2 (prop-set2(i),i=1,ndf)
   etc. for additional nodes
                                    ! Blank terminator record
```

Warning: Degree-of-freedoms with the same base set number *must* have the same proportional load set number.

2. During the solution process it is necessary to compute the normal modes and their associated natural frequencies using the command statements:

```
MASS
TANGent
SUBSpace,,nf
```

Subsequently it is necessary to issue the commands:

```
BASE
TRANSient CONServing
```

followed by the modal solution commands:

```
DT,,delta-t
LOOP time n-steps
  TIME
  MODAl
   .... output statements
NEXT time
TRANSient CONServing
```

The solution steps indicated above are order dependent. Modes must exist in order to perform the `BASE` step. The base step computes the base modes $\mathbf{\Psi}$ and constructs the array $\mathbf{A}_1$ needed to set up the multiple support excitation steps given above. It also requires a factored stiffness matrix constructed by the `TANGent` command. Since base supports are provided, no *shift* should be included on the tangent command.

14.6 Periodic inputs on linear equations

The solution of second order linear equations by the finite element method leads to the set of equations given by Eq. 14.39. If the applied loading is periodic the force may be expressed in the form

$$\mathbf{F}(t) \;=\; \hat{\mathbf{F}}(\omega)\exp(i\,\omega\,t) \tag{14.74}$$

where $i = \sqrt{-1}$ and ω is a specified periodic input frequency. The notation $\hat{(\cdot)}$ denotes a complex quantity. Thus, the intensity of the force is assumed to be a complex vector. Accordingly,

$$\mathbf{F}_r \;=\; \Re(\hat{\mathbf{F}}) \tag{14.75}$$

$$\mathbf{F}_i \;=\; \Im(\hat{\mathbf{F}})\;. \tag{14.76}$$

The real part of the force may be input using the mesh commands `FORCe`, `CFORce`, `EFORce`, and/or `CSURface`. The imaginary part may only input by node number using

the mesh command `IFORce`. Specified boundary displacements may be input using `DISP`, `CDIS`, or `EDIS` for the real part or `IDIS` for the imaginary part.

For the case where $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ are constant matrices a solution to Eq. 14.39 may be constructed by assuming the solution in the form:

$$\mathbf{u}(t) \;=\; \hat{\mathbf{u}}(\omega)\exp(i\,\omega\,t) \tag{14.77}$$

which may be differentiated to define the time derivatives of $\mathbf{u}$. This leads to the equation:

$$\left[-\,\omega^2\,\mathbf{M} \;+\; i\,\omega\,\mathbf{C} \;+\; \mathbf{K} + i\,\eta\mathbf{K}\right]\,\hat{\mathbf{u}}(\omega) \;=\; \hat{\mathbf{F}}(\omega) \tag{14.78}$$

which may be solved for each specified frequency and load to give a solution for the $\hat{\mathbf{u}}(\omega)$. In the above the stiffness term includes the option of: (a) a Kelvin damping where η is a constant or (b) a Rayleigh damping form where $\eta = \omega\,a_1$.

There are some cases where part of the displacement vector $\hat{\mathbf{u}}$ is known and non-zero. For this case we can partition Eq. 14.78 into parts. Let the coefficient matrix be given by

$$\hat{\mathbf{A}} = -\,\omega^2\,\mathbf{M} + i\;(\omega\,\mathbf{C} + \eta\,\mathbf{K}) + \mathbf{K} \tag{14.79}$$

and partition the solution into

$$\hat{\mathbf{u}} \;=\; \begin{bmatrix}\hat{\mathbf{u}}_u \\ \hat{\mathbf{u}}_s\end{bmatrix} \tag{14.80}$$

where $(\cdot)_u$ denotes an unknown part and $(\cdot)_s$ a specified part. With this division, the equations to be solved may be written in the form:

$$\begin{bmatrix}\hat{\mathbf{A}}_{uu} & \hat{\mathbf{A}}_{us} \\ \hat{\mathbf{A}}_{su} & \hat{\mathbf{A}}_{ss}\end{bmatrix}\begin{bmatrix}\hat{\mathbf{u}}_u \\ \hat{\mathbf{u}}_s\end{bmatrix} \;=\; \begin{bmatrix}\hat{\mathbf{F}}_u \\ \hat{\mathbf{F}}_s\end{bmatrix}\;. \tag{14.81}$$

A solution may be achieved by first solving the equation set

$$\hat{\mathbf{A}}_{uu}\,\hat{\mathbf{u}}_u \;=\; \hat{\mathbf{F}}_u \;-\; \hat{\mathbf{A}}_{us}\,\hat{\mathbf{u}}_s \tag{14.82}$$

for the unknown part of the solution vector. Again, during mesh input, the real and imaginary parts may be specified for $\hat{\mathbf{u}}_s$. This may be done using the mesh command options `DISP`lacement, `CDIS`pl, `EDIS`pl, and/or `CSUR`face for the real part and `IDIS`pl for the imaginary part. Once the unknown part is computed the reaction forces may be determined from the remaining part as:

$$\hat{\mathbf{F}}_u \;=\; \begin{bmatrix}\hat{\mathbf{A}}_{uu} & \hat{\mathbf{A}}_{us}\end{bmatrix}\begin{bmatrix}\hat{\mathbf{u}}_u \\ \hat{\mathbf{u}}_s\end{bmatrix}\;. \tag{14.83}$$

In *FEAP* the above is implemented by first declaring the problem to be complex. This is accomplished by starting a problem as

```
*COMPLEX
FEAP * * Title information
    etc.
```

The constant real arrays $\mathbf{M}$, $\mathbf{C}$, $\mathbf{K}$ are then formed and stored in a sparse matrix format in which only the non-zero terms are retained. Then for each specified frequency ω the array $\hat{\mathbf{A}}_{uu}$ is formed and stored in an in-core profile form. This matrix is complex and at present only an in-core profile solution scheme is available in *FEAP*. The force vector is assembled from specified nodal values and from element loads with all proportional load factors set to unity. The solution is then performed and the unknown part combined with the known part to form the total solution vector $\hat{\mathbf{u}}$. The command:

```
CXSOlve,,omega eta
```

is used to perform this step. The first issue of the command will form the arrays $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ which are then used in all subsequent specification of a frequency ω and parameter η. The solution over a series of frequencies may be performed using a parameter for the frequency (e.g. `om`) and loops. For example, the statements

```
LOOP,,10
  PARAm om=om+do
  CXSOlve,,om eta
    ...
NEXT
```

would compute `10` solutions for the frequency at intervals of `do`. An initial value for the parameter `om` and the increment `do` should be set also using a `PARAmeter` input.

After a solution is available the usual *FEAP* commands may be used to output or plot the solution. For example, the command

```
DISPlacement COMPlex k1 k2 k3
```

outputs the real and imaginary parts of the solutions for nodes `k1` to `k2` at an interval of `k3`, whereas the command

```
DISPlacement,,k1 k2 k3
```

outputs only the real part of the displacement for the same increments. Similarly,

```
DISPlacement IMAGinary k1 k2 k3
```

outputs the imaginary part of the displacement for nodes `k1` to `k2` at increments of `k3`. Finally,

```
DISPlacement CMPL k1 k2 k3
```

outputs the real and imaginary parts of the displacement for the nodes with coordinates in cylindrical form and is primarily recommended only for 2-d problems.

The plot commands `PLOT,REAL` and `PLOT,IMAG` set the display contours to the real and imaginary parts, respectively. The usual plot commands (e.g., `PLOT,CONT,i`) then give the desired solution part.

14.7 Time step control in transient solutions

FEAP provides alternatives to control the size of the time increment during transient solutions. The simplest merely monitors the number of iterations required to converge a Newton solution step; thus, it is ineffective in solution of linear problems. The second method monitors the change in constitutive variables (e.g., change in accumulated plastic strain) to control the accuracy of the solution at the material model level. This is not effective for linear elastic models but may be used effectively for some quasi-static or transient solutions. These two methods are described by the `AUTO` command as described in Appendix D.

14.8 Time dependent loading

The loading applied to a problem may be changed during a solution process. This may be achieved by specifying new nodal loads for each time step using the commands

```
BATCh
 ...
LOOP time steps
  MESH
   ...
NEXT
 ...
END batch commands
FORCe
 ...
END mesh data
FORCE
 ...
END mesh data
 ... ! Must be at least 'steps' sets
```

in which a set of new forces appears for each time step performed. The use of the `MESH` command within a solution strategy permits the alteration of any nodal or element data. It is not permitted to change the size of the problem by adding new nodes or elements (elements may be `ACTI`vated or `DEAC`tivated based on region descriptions); however, nodal forces, displacements, boundary restraint codes, etc. may be changed. Material parameters may be changed but not the type of material model (i.e., it is not permitted to change a model from elastic to elasto-plastic during the solution process).

14.8.1 Proportional time functions

The above form, while general in concept, requires extensive amounts of data to describe the behavior. *FEAP* can easily treat loading states which may be written in the form

$$\mathbf{F}(t) = p_j(t)\,\mathbf{F}_j \tag{14.84}$$

where $p_j(t)$ is a set of time dependent (proportional loading) factors and $\mathbf{F}_j$ is a fixed loading pattern on a mesh.

To perform an analysis involving proportional loads, during mesh input it is necessary to specify:

1. Nodal force patterns $\mathbf{F}_j$;

2. Associations between force patterns and proportional loading factors using the $FPRO$portional command during mesh generation. This command has the form

```
FPROportional
  NODE NG J1 J2 ... Jndf
     ...
                        ! Termination record
```

   where the `Ji` define the proportional load number p_j assigned to a degree of freedom. A zero value will use the *sum* of all specified proportional load factors as the multiplier for an associated force or displacement, whereas, a non-zero value will use only the individual p_j factor. Alternatively, it is possible to use the `EPROp`, `CPROp` or `LOAD` group commands to specify the proportion al function $p(t)$.

3. Last, the proportional loading function $p_j(t)$ is defined by:

A proportional loading function is specified using the solution command

```
PROPortional,,Ji Jj
```

where `Ji` and `Jj` define a range of loading to be input. Note the use of two commas to describe the missing (required) field. If `Jj` is zero only the `Ji` function is to be input. A functional type of proportional loading is

$$p(t) = a_1 + a_2(t - t_{min} + a_3[\sin(a_4(t - t_{min}) + a_5]^k \quad ; \quad t_{min} \le t \le t_{max} \tag{14.85}$$

and is input by the statements

```
PROP,,J1
 ...
END
1 K T-min T-max A1 A2 A3 A4 A5
```

This is called Type 1 loading and requires a 1 in the first column defining the parameters. Note that the argument of the *sine* function is given in *radians* and not in degrees. *A blank record* is considered to be a Type 1 loading with default parameters:

$$t_{min} = 0 \ ; \ t_{max} = 10^6 \ ; \ a_2 = 1 \ ; \ k = a_1 = a_3 = a_4 = a_4 = 0 \ . \tag{14.86}$$

A piecewise linear set of values may be input using the *Type 2* proportional loading function which is specified by a `PROP`ortional command whose data is:

2	n				
t_1	p_1	t_2	$\cdots$	t_n	p_n
t_{n+1}	p_{n+1}	$\cdots$	$\cdots$	t_{2n}	p_{2n}
$\cdots$	$\cdots$				

by default $n = 1$ and the values appear as time-value pairs on each record. *Input terminates with a blank record.* A piecewise velocity specification may be given in the same way by using a *Type 7* (instead of 2) proportional function. The velocity is integrated to give quadratic variation between the specified values.

Proportional loading may be turned off using the command

```
PROP OFF time dt
```

The parameter `time` may be used to reset the solution time to a new value and the parameter `dt` may be used to define a new time increment size. Note that, omitting the parameters sets both values to zero. After use of this command the magnitude of the proportional function $p(t)$ is unity for all times, however, new proportional loading functions may be defined if desired.

See Section 14.1.28 and Appendix D for additional options to specify the proportional function $p(t)$.

14.9 Continuation methods: Arclength solution

Many non-linear static problems have solutions which exhibit limit load states or other types of variations in the response which make solution difficult. Continuation methods may be employed to make solutions to this class of problems easier to obtain. *FEAP* includes a version of continuation methods based on maintaining a constant length of a specified load-displacement path. This solution process is commonly called an *arclength* method. To employ the arclength method in a solution the command `ARCL`ength is used. A typical batch algorithm for an arclength solution is given by:

```
BATCh
```

```
  ARCLength,,<k = 0 to 5>
  DT,,delta-t
  LOOP time n-steps
    TIME
    LOOP newton n-iters
      TANGent,,1
    NEXT iteration
     ..... (outputs, etc.)
  NEXT,time
END
 <0,1>              ! Arclength data for all options
 <node dof d_inc>   ! Arclength data for options 4 & 5
```

The options `k = 1 to 3` use load levels as the control, whereas, options `k = 4,5` use a displacement parameter. The last record gives the node, degree of freedom and displcement increment for displacement control options (see description in Appendix D). Using the command as

```
ARCLength
```

defaults to option 2 and thus only one record is required after the `END` command.

Remark: It is *not* permitted to use a `PROP`ortional loading command with the arclength procedure.

14.10 Quasi-Newton methods

Two quasi-Newton methods are available in FEAP as options to reduce the solution effort where computation and factoring of the tangent array are very expensive. For problems with a symmetric tangent the BFGS method may be used (see Appendix A. for details on input and use). For problems with unsymmetric tangents the Broyden method provides an option (see Appendix A for input on this command also).

14.11 Augmented solutions: Incompressibility and constraints

FEAP has options to employ penalty method solutions to enforce constraints. A penalty method is used to (1) enforce incompressibility constraints in continuum elements; (2) enforce the magnitude of the gap in satisfaction of contact constraints; and (3) as an option of the `GAP` element. The use of large penalty parameter values in

some material models and some finite deformation analyses make a Newton iteration loop difficult or impossible to converge. Often when the penalty parameter value is reduced so that acceptable convergence of the iteration is achieved it is observed that the constraint is not accurately captured. In these cases it is possible to achieve a better satisfaction of the constraint by using an *augmented Lagrangian* solution strategy. The augmented Lagrangian solution scheme implemented in *FEAP* is based on the Uzawa algorithm briefly discussed in references[19] and.[1] The command language program to perform an augmented solution is given by:

```
LOOP augment n-augm
  LOOP newton n-iter
    TANGent,,1
  NEXT iteration
  AUGMent,,augf,gap
NEXT augment
```

where `augf` is a factor used during the augmented update and `gap` is the maximum gap permitted in a contact interface (where a gap is a penetration). If a penalty value v_p is computed by a relation

$$v_p = K * c(u)$$

where K is a material parameter and $c(u)$ is a constraint to be made close to zero, then the Uzawa update is computed as

$$v_a = v_a + augf \cdot v_p \ .$$

By default `augf` is set to unity (1).

The balance between a choice of K, `augf`, and the number of augmented iterations is quite delicate. For the Uzawa algorithm the convergence rate for the augmented loop is *linear* (even when the Newton loop convergence is asymptotically quadratic). Improvement can be obtained by increasing the value of K thus reducing the number of augmented loops which need to be computed. Alternatively, it is possible to use a parameter for `augf` which is increased for each augmented loop.

When used in a time loop

```
LOOP time nt
  TIME
  LOOP augment na
    LOOP newton nn
      TANG,,1  (or UTAN,,1)
    NEXT newton
    AUGMent
  NEXT augment
    .... other commands
NEXT time
```

convergence will be checked for both the loop level containing the AUGM statement and the one containing the TANG or UTAN statement. For other case the number of augmented iterations (n-augm) should be kept quite small as convergence of the iteration process is only checked by the TANGent (or UTAN) command. If convergence is achieved in this loop execution passes to the AUGMent command and another augmentation is performed until the n-augm augmentation iterations are performed.

14.11.1 Incompressibility constraint

The use of augmented solutions to satisfy incompressibility is restricted to solid elements of *mixed* and *enhanced* forms. Use with a standard displacement model would lead to very poor performance and, thus, is not permitted.[1,19] By default, augmentation is always possible using any of the above algorithms. However, it is possible to turn off the feature for elements associated with specific materials. This is defined by giving the material model options as

```
MATErial ma
  SOLID
    ....
    AUGMent OFF
```

Alternatively, if many materials are used and it is desired to only activate augmenting in a few materials the commands may be given as:

```
GLOBal
  AUGMent OFF
```

followed by (this is order dependent)

```
MATErial ma
  SOLID
    ....
    AUGMent ON
```

Use of the augmented feature for contact or the GAP element is off by default and must be specifically stated when it is to be used (see below for contact).

14.12 Solution of contact problems

The solution of a contact problem requires two basic steps. In the first step the determination of the node/element pairs to be in contact is determined. This involves a search over the *facets* defining the master and slave surfaces to determine which are in a contact state. This step is require first to ensure that the necessary storage is available to handle the stiffness coefficients generated in the second step.

There are several options which may be considered to handle the setting of the contact states between the master and slave surfaces. One option is to set the state and then perform a Newton type solution until convergence is obtained. Once convergence is obtained the contact state is checked again and then another Newton type solution. The command language statements to perform this type of solution are given as:

```
LOOP contact n-check
  CONTact CHECk
  LOOP newton n-iters
    TANGent,,1
  NEXT newton
NEXT contact
```

The three dimensional contact algorithm is programmed only for this option and the `CONTact CHECk` command must always exist within the solution algorithm statements in order for a contact solution to be performed.

A second option is to check at each iteration. In this case a single loop solution algorithm can be given as:

```
LOOP newton n-iters
  CONTact CHECk
  TANGent,,1
NEXT newton
```

Since a finite element problem is discrete in nature it is possible for a contact state to *oscillate* between facets and, therefore, no full convergence is obtained for the Newton solution strategy. Indeed the quadratic asymptotic convergence rate may not be obtained due to this phenomenon.

If an augmented strategy is used it is necessary to include additional statements in the solution loop. One method is given by:

```
LOOP augment a-iters
  AUGMent
  LOOP newton n-iters
    CONTact CHECk
    TANGent,,1
  NEXT newton
NEXT augment
```

where `a-iters` is the number of augmentation loops to perform. Generally, with this option the *penalty parameter* is set to some moderate value which permits slight penetration and 3-4 augmentation loops performed to obtain a good satisfaction of the impenetrability condition.

14.13 Time history plots

The response of specific solution quantities may be saved in files during solution using the TPLOt command. This permits the construction of time history plots during or after the completion of a solution using any program which is capable of constructing x-y plots from files (e.g., using Gnuplot or Matlab). The TPLOt command works only with time dependent problems and whenever the command TIME is executed writes data to files with the name designated for plots at the start of execution and an added extender. To recover the last computed data set it is necessary to conclude an analysis with a TIME command. The TPLOt command is given as

```
TPLOt
 ...
END
type n1 n2 x y z
show (optional to force echo of data list)
...
              ! Termination record
```

The parameters may have the values described in column one of Table 14.5. Each plot type may output up to 200 components. The output is placed in a file with name Pxxx.ext where the extender is given in column two of Table 14.5 and xxx is extracted from either the input file name or form a name given at the initiation of the problem as the plot file name. A letter a to j is added at the end of xxx to designate groups of up to 20 items of data (the maximum length of a written record is 252 characters). These files may be used by another program (e.g., MATLAB) to prepare x-y plots. The output records for each file type except arclength are written as

```
TIME  VALUE_1   VALUE_2 ....   VALUE_20
```

For arclength the file is written as

```
LOAD_VALUE  VALUE_1   VALUE_2 ....   VALUE_20
```

where LOAD_VALUE is the level of the arclength parameter. The actual level is the pattern of loading multiplied by this value.

Indicated data may be given either by the node number, or the coordinate of the point where the data is located (the closest node to the point will be selected). The energy components, if computed, should be ordered as: 1-3: linear momentum; 4-6: angular momentum; 7: kinetic energy; 8: stored (strain) energy; 9: dissipated energy; 10: total energy; and 11: angular momentum norm.

The components output by the STREss option in the TPLOt command are indicated in Table 14.6.

Type	File	n_1	n_2	x	y	z
`DISPlacement`	`.dis`	Node	dof	x	y	z
`VELOcity`	`.vel`	Node	dof	x	y	z
`ACCEleration`	`.acc`	Node	dof	x	y	z
`REACtion`	`.rea`	Node	dof	x	y	z
`STREss`	`.str`	Elmt	Comp.	x	y	z
`ARCLength`	`.arc`	Node	dof	x	y	z
`CONTact`	`.con`	Node	dof	x	y	z
`ENERgy`	`.ene`	Comp	Print	-	-	-

Table 14.5: Tplot types and parameters

No.	Solids	Frames (2d)	Frames (3d)	Truss	Shell (2d)	Shell (3d)	Plates	Thermal
1	σ_{11}	N	N_1	σ	N_{11}	N_{11}	M_{11}	q_1
2	σ_{22}	ϵ_0	N_2	ϵ	N_{22}	N_{22}	M_{22}	q_2
3	σ_{33}	V	N_3		S_{13}	N_{12}	M_{12}	q_3
4	σ_{12}	γ	M_1		M_{11}	M_{11}	S_{13}	$T_{,1}$
5	σ_{23}	M	M_2		M_{22}	M_{22}	S_{23}	$T_{,2}$
6	σ_{31}	χ	M_3			M_{12}		$T_{,3}$
7	ϵ_{11}	σ				S_{13}		
8	ϵ_{22}	ϵ				S_{23}		
9	ϵ_{33}							
10	ϵ_{12}							
11	ϵ_{23}							
12	ϵ_{31}							

Table 14.6: Components for `TPLOt STREss` option

14.14 Re-executing commands: HISTory command

A useful feature of the command language for interactive executions is the HISTory command. During the execution of solution commands the program compiles a list of all commands executed (called the *history list*) which may be used to re-execute one or several of the commands. The user may also SAVE this list as a file and at a later time READ the list back into the program. At any stage of interactive execution the list may be displayed by entering the command HIST,LIST or HIST; alternatively, a portion of the list may be displayed; e.g., HIST,LIST,5,9 will display only commands 5 through 9. A user may then re-execute commands by entering the command numbers from the history list. For example, HIST,,1 (note the double commas as field separators) would re-execute command 1, or HIST,,6,9 would re-execute commands 6 through 9 inclusive. The history commands also may be embedded in a normal command language LOOP-NEXT pair; e.g., entering the commands:

```
LOOP,,4
  HIST,,6 9
NEXT
```

performs a loop 4 times in which during each loop commands 6 through 9 are executed. If the history commands 6 to 9 involve a loop or next which is not closed it is necessary to provide a closing sequence before execution will commence.

14.15 Solutions using procedures

Many analyses require the use of a sequence of commands which are then reused throughout the solution process or in subsequent solution of problems. For example, in the analysis of static problems the sequence of commands:

```
TANG,,1
DISP ALL
STRE ALL
REAC ALL
STRE NODE 1 50   !(output first 50 nodes)
```

may be used. The repeated input of this sequence is not only time consuming but may result in user input errors. This sequence of commands may be defined as a PROCedure and saved for use during the current analysis or during any subsequent analysis. A procedure only may be defined during an interactive solution; however, it may be used in either a batch or interactive solution (including the solution in which the procedure is defined). A procedure is saved in the current directory in a file with the extender .pcd.

A procedure is created during an interactive analysis by entering the command:

```
PROCedure name v1 v2 v3
```

The name *procedure* may be abbreviated by the first four (or more) characters, *name* is any 1-8 character alphanumeric identifier which specifies the procedure name (*the first 4 characters must not be the same as an existing command name*), `v1,v2,v3` are any 1 to 4 alphanumeric parameter names for the procedure. The parameters are optional. For example the procedure for a static analysis may be given as:

```
PROCedure STATIC NODE
```

After entering a procedure name and its parameters, prompts to furnish the commands for the procedure are given. These are normal execution commands and may not contain calls to other procedures or `HIST` commands. The parameter names defined in the procedure (e.g., `NODE` in the above `STATIC` command) may be used in place of any numerical entries in commands. A procedure is terminated using an `END` command. As an example the complete static analysis procedure would read:

```
PROCedure STATIC NODE
  TANG,,1
  DISP ALL
  STRE ALL
  REAC ALL
  STRE NODE 1 NODE
END
```

Note that in the *nodal stress* command the parameter `NODE` is used twice. The first use is for the definition of the command and is an alphanumeric parameter of the command. The second `NODE` is a numerical parameter of the command. The value for this `NODE` parameter is taken from the one specified at the time of execution. The use of the static procedure is specified by entering the command line:

```
STATIC,,50
```

and, at execution, the 50 will be the value of the `NODE` parameter in the procedure definition above (e.g., the first 50 nodal stresses will be output). All characters in the *name* (e.g., up to 8 characters) of a procedure must be specified. It is not permitted to abbreviate the name of a procedure using the first four characters of the procedure *name*.

The procedure `STATIC` may be used in any subsequent analysis by entering the valid procedure name and parameters (if needed). Currently it is not possible to preview a procedure while a solution is in progress (they can be viewed from other windows in a multi-window compute environment). Thus in large analyses it is advised that a review of the `NAME.PCD` file be made to look at the contents. Extreme care should be exercised to prevent long unwanted calculations or outputs from an inappropriate use of a procedure. For example, a `STREss,ALL` is a viable command for small problems but can produce very large amounts of data for large problems.

14.16 Solutions using functions

During the solution phase it is sometimes useful to have parameters which change values according to some prescribed manner. For example, in the solution of a non-linear problem using an augmented strategy (e.g., see Section 14.11 above) one may wish to use a parameter on the `AUGM` command for the `augf` factor. This may be achieved using the command as:

```
AUGMent,,af
```

where `af` is now interpreted as a parameter. It is necessary now to be able to set the parameter to an initial value for each time step in the solution. This may be achieved by using a *function* data set which is read by *FEAP* by the command language statement

```
FUNCtion name
```

where `name` denotes the filename `name.fcn` which must be placed in the same directory as the input data file. Thus we could have two function data files which set and modify the parameter `af`. The file which sets the data is here given the name `setaf.fcn` and contains the single statement:

```
af = 1
```

The file which multiplies the parameter `af` is given the name `mulaf.fcn` and contains the single statement:

```
af = af*2
```

and here we indicate that the parameter is to double each time the function is called. A set of command language statements which use the above functions may then be given as:

```
LOOP time nt
  TIME
  FUNC setaf
  LOOP augment na
    LOOP newton nn
      TANG,,1
    NEXT newton
    AUGM,,af
    FUNC mulaf
     ...
  NEXT augment
NEXT time
```

The multiplier may also be given as a parameter if desired and either another function file prepared or a command language statement `PARA` may be used to set its value.

14.17 Output of element arrays

When solving problems difficulties often occur for which additional information is needed about the element. *FEAP* includes a capability to print the arrays produced by the highest numbered element (i.e., the one whose number is `NUMEL`) by the last command. The command is named `EPRI`nt. tangents. Similarly, the element mass matrix used for eigen computations could be output using the command immediately after the `MASS` command.

If additional information about the array is desired it is possible to make a spectral transformation, but for symmetric tangents only. This is obtained by using the command

```
EIGElement vector
```

Omitting the `vector` option outputs eigen-values only. This may be useful to ensure an element has the proper number of rigid body modes, or that it is correctly defined. Presence of any negative eigen-values should be very carefully interpreted as normally they imply solution difficulties.

Chapter 15

Plot outputs

FEAP provides for the construction of plots to represent features of the problem and its solution. Currently, the following basic input commands are included as part of the system.

```
ACCE   AWIR   AXIS   BACK   BORD   BOUN   BPLO   CAPT
CART   CENT   CLEA   CLIP   COLO   CONT   CVAR   CWIR
CYLI   DEFA   DEFO   DISP   DOFS   DPLO   EDEF   EIGE
EIGV   ELEM   ELPL   ESTR   EWIR   EXNO   EYES   FACT
FILL   FLUX   FRAM   FULL   HIDE   HIST   IMAG   INTE
JINT   LABE   LINE   LOAD   LOGO   MANU   MARK   MATE
MESH   NODE   NOFU   NOLA   NOPR   NORA   NORM   OUTL
PAIR   PBOU   PDIS   PELE   PERS   PFOR   PICK   PNOD
POST   PRAX   PRIN   PROF   PROJ   PROM   PSNO   PSTR
PWIR   QUAD   RANG   REAC   REAL   RECT   REFR   REGI
RENU   RFOR   SCAL   SCRE   SHOW   SIZE   SNOD   SPHE
SPLO   STRA   STRE   STRI   SWEE   SWIR   SYMM   TEXT
TIME   TITL   TORS   UNDE   UPLO   VELO   VWIR   WIND
WIPE   XYPL   ZOOM   JPEG
```

Appendix E contains specific instructions for use of each of the commands, including use of their parameters for advanced applications.

15.1 Screen plots

FEAP presents graphics to the screen designated when starting the program. Options include an X11-graphics mode for UNIX/Linux environments and a PC-graphics mode for Windows environments. In addition to basic plot construction, both options include use of the mouse to clip the plot region and to add basic features to the analysis (e.g.,

show locations for boundary restraints, applied loads, etc.). Plots are constructed using commands, similar to those described above for problem solution, and may be performed in a batch mode as

```
Macro> PLOT command options
```

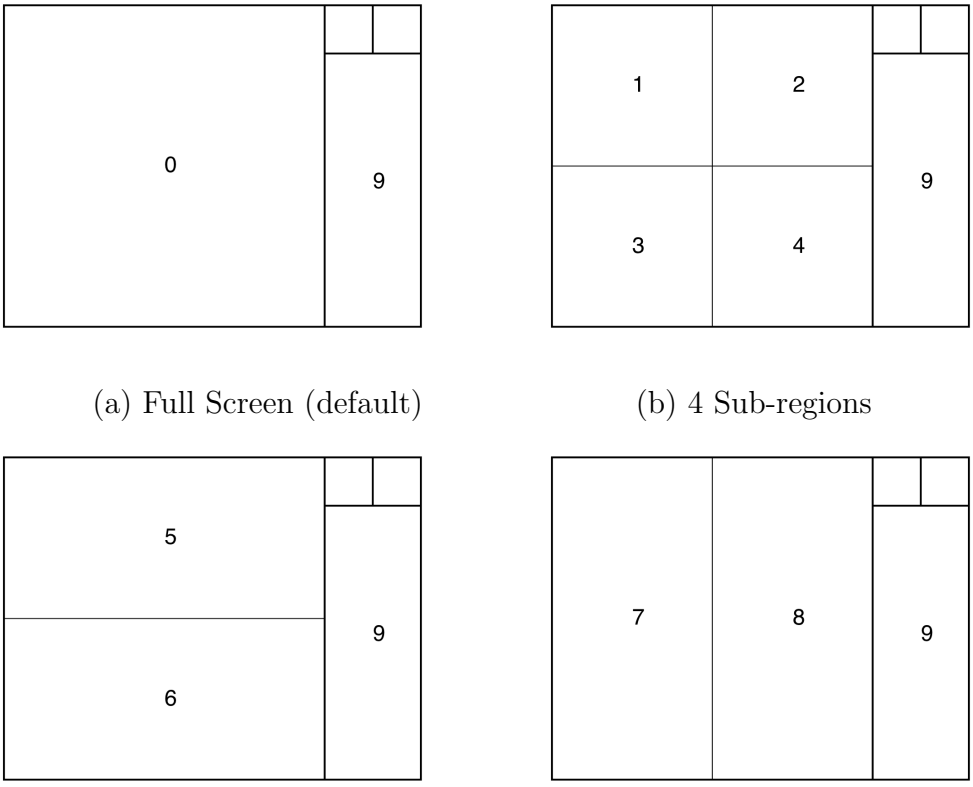
or in an interactive mode by first issuing the command

```
Macro> PLOT
```

The text screen will now have the prompt as

```
Plot>
```

Now it is only necessary to issue the plot command and its options. followed by the sequence of plot commands to be executed and thus when in an interactive mode the `PLOT` is not issued as part of the command.



(a) Full Screen (default) (b) 4 Sub-regions

(c) Horizontal split (b) Vertical split

Figure 15.1: Plot regions in graphics screen.

The complete list of graphics output features available using the `PLOT` command are described in Appendix E; however, some options to construct basic plots are described below.

15.1.1 Graphics plot regions

For graphics outputs the screen is divided into frames as shown in Fig. 15.1. By default the entire plot region [Part labeled "0" in Fig. 15.1(a)] is used to display graphics output. The part labeled "9" is used for information about the plot, usually contour intervals. To change the plot display region the command

```
PLOT FRAMe n
```

may be specified, where n may be assigned values from "1" to "8". The command

```
PLOT FRAMe
```

restores plots to region "0". For plots to frames "5" to "8" the alternative commands

```
PLOT HFRAme <1,2>  (same as PLOT FRAMe <5,6>)
```

or

```
PLOT VFRAme <1,2>  (same as PLOT FRAMe <7,8>)
```

may be specified. When contours are displayed in frames "1" to "8" a text label is added at the bottom of each frame.

15.1.2 Clear graphics screen

The plot screen may be erased at any time using the command

```
WIPE
```

For the `HFRAme` plot frame set the wipe may be given as

```
PLOT HWIPe <1,2>  (same as PLOT WIPE <5,6>)
```

and for the `VFRAme` form by

```
PLOT VWIPe <1,2>  (same as PLOT WIPE <7,8>)
```

Finally, the frame displaying the contour intervals may be cleared using the command

```
PLOT WIPE 9
```

Alternatively, the central portion of the screen may be erased using the command

```
CLEAr
```

15.1.3 Mesh plots

The command

```
PLOT MESH
```

will display a wire frame display of the mesh. Optionally, the interior of each element may be filled in a color by giving the command

```
PLOT FILL
```

The color for different materials will be selected based on each material number.

15.1.4 Deformed and undeformed plots

By default, all outputs are given using the *undeformed* configuration defined by the nodal coordinates. For solid mechanics problems in which the first `ndm` degrees of freedom are displacements a deformed plot may be plotted in which the nodal coordinates are given by:

$$\mathbf{x}_I = \mathbf{X}_I + cc \cdot \mathbf{u}_I$$

where $\mathbf{X}_I$ are the input nodal coordinates, $\mathbf{u}_I$ are the current values of the nodal displacements and cc is a scaling factor which is set using the command

```
PLOT DEFOrm cc nn es
```

The default value of the `cc` parameter is unity (1). If the parameter `nn` is zero (or absent) the deformed plot will be rescaled to fit into the plot window; if it is non-zero no rescaling will be made. The parameter `es` is used to scale eigen-vector plots

Thus, giving the commands as

```
PLOT MESH
PLOT DEFOrm 5 1
PLOT MESH
```

will create a screen plot with the undeformed mesh and the deformed mesh on the same figure. The current values of the displacements are multiplied by a factor 5 and no rescaling is requested.

Similarly, the command `UNDEform,,nn` returns plots to an undeformed configuration without rescaling if `nn` is non-zero or with rescaling if it is zero.

15.1.5 Node and element number locations

To place the nodes and their numbers on the screen the command

```
PLOT NODE
```

is given. To place the number for node 5 only, the command

```
PLOT NODE 5
```

is used. For large meshes outputs for all the node numbers become unreadable. Display of just the nodes is possible using the command

```
PLOT NODE -1
```

where the `-1` is used to indicate no plot of node numbers.

Similarly, all element numbers are placed on the mesh using the commands `ELEMent` or `ELEMent 4` to get all elements or only element 4, respectively.

15.1.6 Cartesian and perspective views

By default all plots are given in a Cartesian view where the x, y plane coincides with the plane of the screen. A perspective view of any mesh may be constructed using the command `PERSpective`. For three dimensional problems, the command `HIDE` should be used after the `PERSpective` command to develop all plots on the visible surfaces. To return to the Cartesian form of plots the command `CARTesian` is used.

Features may be added to mesh plots by using other commands. An outline of a mesh may be displayed using the command `OUTLine`. In three dimensions, the mesh surfaces are filled to prevent hidden surfaces from appearing.

Changing the view point

When a `PERSpective` view is selected a default view point is selected in the positive quadrant of the xyz Cartesian coordinate frame. The view point may be changed using the command

```
PLOT PERSpective 1
```

which requires additional data after the `BATCh END` record. The data consists of the following:

```
0
vx vy vz
ax ay az
```

where `vx, vy, vz` are coordinates of the new view point (in problem coordinates) and `ax, ay, az` are components of the axis to be vertical on the screen. In an interactive mode prompts will appear in the text window to guide the input.

An alternative in interactive mode is to use the plot `EYES` command where the location of the view point is selected from the two small boxes in the upper right of the graphics window using the mouse.

Rotation of perspective plots

The orientation of a graphical view may be changed using the command

```
PLOT ROTAte iaxis theta mesh
```

where `iaxis` is the Cartesian axis of the object about which a rotation is to be applied (1,2,3) and `theta` is the incremental angle in degrees of the applied rotation about the current view. If a parameter `mesh` greater than unity is given a plot of the new mesh will be placed on the screen. Usually the command should be given in an interactive mode after issuing the command `PLOT` followed by

```
PERSpective
HIDE
MESH (or some other display command)
```

and then

```
ROTAte iaxis theta mesh
```

This command may be repeated as often as desired with each new angle being applied to the current orientation. Giving the command as

```
ROTAte
```

with `theta` zero resets the orientation matrix to an identity and restores the view to the default of the current view point.

15.1.7 Boundary restraints

To display the boundary conditions for degree-of-freedoms 1 to 3 the command `BOUN` may be used. Alternatively, any individual directions restraints may be shown using `BOUN,dir`, where `dir` ranges from 1 to 3. At present, boundary conditions for degree-of-freedoms greater than 3 may not be displayed.

15.2 Contour plots

Once a solution is performed using the command language features described in Chapter 14 it is possible to display several features of the solution. For example, vectors of the nodal displacements may be shown using the command `DISP`lacement.

15.2.1 Displacement contours

Contours of the displacements within each element may be constructed using

```
PLOT CONTour dof
```

where `dof` is the number of the displacement to contour. A range of values will be selected and if a default mode is in effect the contours will be placed on the screen. If the default mode is inactive it is necessary to select the plot ranges (default values are suggested and may be accepted by using the enter key). The default mode may be turned on and off in interactive mode using the commands `DEFAult,ON` and `DEFAult,OFF`, respectively. Generally, the mesh will be shown as a black outline around each contoured surface facet. To eliminate the element edges in the mesh and obtain smooth contours the command

```
PLOT CONTour dof,,1
```

may be used. Any nonzero number may be used in the third parameter position. Alternatively, the edges may be removed using the command

```
PLOT OFF
```

They may be restored using the command

```
PLOT ON
```

This option works for all contours of displacements, stress, eigenvectors, etc.

Wire frame plots

Filled plots often hide some of the features of the results. An option to plot contours only for edges of elements (a wire frame plot) may then be used. The interior edges may be retained by omitting the `HIDE` plot command or exterior edges only by retaining it. The options for wire frame plots exist for contouring: Displacements (`CWIRe`; Velocities (`VWIRe`); Accelerations (`AWIRe`); Stresses (`SWIRe`); Principal stresses (`PWIRe`); Eigenvectors (`EWIRe`). Thus, the command

```
PLOT CWIRe 1
```

displays the shading on element edges for the DOF 1 solution variable. Options are identical to corresponding filled commands.

15.2.2 Stress contours

Contours of element variables, such as stresses, may be constructed using the `STREss,comp` or `SWIRe,comp` commands where `comp` is the component to be plotted. For *FEAP* solid elements the stress components are ordered as shown in Table 15.1. By default element variables such as stress or strain are allowed to be discontinuous between different material sets. If it is desired to have them continuous the *solution command*

```
PROJect CONTinuous
```

should be given before issuing the element variable plot command. A command

```
PROJect
```

will restore discontinuous projections

To construct contours the stress values are first projected to the nodes using a local least squares approach[71] or a row sum lumped approach.[1] For two-dimensional meshes it is also possible to show the unprojected stress contours using the `ESTRess,comp` command.

COMP	Description
1	11-Stress
2	22-Stress
3	33-Stress
4	12-Stress
5	23-Stress
6	31-Stress

Table 15.1: Component number for solid element stress value

COMP	Description
1	1-Principal Stress
2	2-Principal Stress
3	3-Principal Stress
4	Maximum Shear (2-D)
5	I_1-Stress Invariant
6	J_2-Stress Invariant
7	J_3-Stress Invariant

Table 15.2: Component number for solid element principal stress value

Principal stress contours

Projected principal values of stresses may also be displayed using the command `PSTRess,comp` where the components are ordered as shown in Table 15.2

The directions of the principal axes at nodes may be shown using the command `PRAXis`.

15.2.3 Strain contours

Contours of solid element strain variables may be constructed using the `STRAin,comp` command where `comp` is the component to be plotted. For *FEAP* solid elements the strain components are ordered as shown in Table 15.3.

To construct contours the strain values are first projected to the nodes using a row sum lumped approach.[1] For finite deformation problems the components are Almansi strains.

COMP	Description
1	11-Strain
2	22-Strain
3	33-Strain
4	12-Strain
5	23-Strain
6	31-Strain

Table 15.3: Component number for solid element strain value

No.	Solids Coupled	Frames (2d) Nodal	Frames (3d) Nodal	Truss	Shell (2d) Finite	Shell (3d) Finite	Plates	Thermal
1	σ_{11}	N_1	N_1	N	N_{11}	N_{11}	M_{11}	
2	σ_{22}	N_2	N_2	q	N_{22}	N_{22}	M_{22}	
3	σ_{33}	M_1	N_3		S_{13}			
4	σ_{12}		M_1		M_{11}	N_{12}	M_{12}	
5	σ_{23}		M_2		M_{22}	S_{13}	S_{23}	
6	σ_{31}		M_3			S_{23}	S_{23}	
7	ϵ_{11}				ϵ_{11}	M_{11}	χ_{11}	
8	ϵ_{22}				ϵ_{22}	M_{22}	χ_{22}	
9	ϵ_{33}				γ_{13}			
10	ϵ_{12}				κ_{11}	M_{12}	χ_{12}	
11	ϵ_{23}				κ_{22}		γ_{13}	
12	ϵ_{31}						γ_{23}	
13	q_1							q_1
14	q_2							q_2
15	q_3							q_3

Table 15.4: Components for `PLOT STREss` option

15.2.4 Thermal contours

The thermal flux ($\mathbf{q}$) contours may be plotted using the command

```
PLOT FLUX n
```

where `n` is the component. As shown in Table 15.4 the values are placed in the projection numbers 13 to 15, consequently, it is also possible to plot the flux using

```
PLOT STRE <13,14,15> ! Same as FLUX <1,2,3>
```

to get the same output.

15.2.5 Contours for *FEAP* element types

In addition to contouring solid elements, it is possible to contour results from other element types. A partial list of those available is shown in Table 15.4. These also

define the values in the outputs from STREss NODE. Since some of the contours use identical number it is desirable when such conflicts can occur to set the material number individually to make plots. This may be accomplished using the command

```
PLOT MATE ma
```

where ma defines the individual material number to plot.

15.2.6 Eigenvectors

To plot an eigen-vector it is necessary to provide the es scaling using a EDEForm,es or a DEFOrm command before issuing the eigen-vector plot command EIGVector,num or EWIRe,num where num is the number of the vector to plot. The es in the EDEForm command is a fraction of the mesh by which the eigenvector is scaled (the default is 0.10). The ordering for num is the same as that obtained for the eigen-values computed by the SUBSpace solution command.

15.2.7 History variables

For material models that use history variables contours for each variable may be displayed using the command

```
PLOT HISTory hpnum
```

where hpnum is the history plot number.

The specification of each hpnum and its associated history variable hvar is given with the material set data as

```
MATErial
  <SOLId,MEMBrane,SHELl>
     ...
     HIST plot hvar hpnum
     ...
```

The hvar numbers for the small strain models are listed in Table 15.5 and those for finite strain models in Table 15.6. User models can specify their history variable numbers also, however, the material HIST command must appear before the UCON command.

15.3 Plots for mesh subregions

In interactive mode it is possible to select a part of the mesh region for displaying plotted quantities. This is performed using the command PICK and then the mouse to

PLASticity	Plane strain & 3-d	1 = e^p; 2 to 7 = $\boldsymbol{\epsilon}^p$
	Plane stress	1 to 3 = $\boldsymbol{\epsilon}^p$; 7 to 9 = Back stress
	Generalized	1 = e^p; 4 to 9 = $\boldsymbol{\epsilon}^p$
VISCoelasticity	Plane	5 to 8 = $\mathbf{q}_1$ (viscoelastic strain)
	3-d	5 to 10 = $\mathbf{q}_1$ (viscoelastic strain)

Table 15.5: Small strain history variable list for plots.

PLASticity	Plane & 3-d	1 = e^p; 3 to 8 = $\mathbf{b}_e$; 9 to 14 = $\boldsymbol{\epsilon}^p$
	Generalized	1 = e^p; 3 to 8 = $\mathbf{b}_e$; 9 to 14 = $\boldsymbol{\epsilon}^p$
VISCoelasticity	Plane & 3-d	1 = ξ (damage); 2 to 7 = $\mathbf{S}$

Table 15.6: Finite strain history variable list for plots.

select two points bounding the region to be plotted. The points selected will be used to construct a square region and, thus, may be slightly different than selected. To return to a full mesh plot use the command `ZOOM`.

15.4 PostScript plots

FEAP provides for construction of files in the encapsulated PostScript format. To construct a PostScript file for graphics output the command `POST` is given. The first time the command is used a file is opened and named. The name of the file is `FEAPxxxx.eps`, where `xxxx` is a sequence of letters starting with `AAAA`. Information for all commands issued after the `POST` command will appear both on the screen device and in the file. The second time the command is given the PostScript file is closed. If another pair of `POST` commands are issued a new file will be created and closed. The files may be converted to hard copy in a UNIX environment using the `lpr` command.

PostScript files may be created in either a portrait or landscape mode. In addition, the *FEAP* logo is normally not placed in the file. Options exist to add the logo.

One example of using the PostScript options is a mesh plot and load for a given problem. For two-dimensional applications the set of commands:

```
PLOT POSTscript    !open a file to accept plot data
PLOT MESH          !plot mesh
PLOT LOAD,,-1      !plot load with tip on nodes
PLOT POSTscript    !close file
```

produces a file containing the mesh and load. This is the set of commands which produced Figure 5.1. If desired the location of the origin of the coordinate axes may be

displayed using the command AXIS. If the origin is outside the plot window the axes will not appear. It is possible to relocate the axes by giving the command AXIS,x,y,z where the x,y,z are dimensions in terms of the mesh coordinates.

In three dimensions it is usually preferable to select a perspective type plot and view options and then produce surface plots which *hide* parts of the mesh not visible. Thus, prior to issuing the graphical output commands one should issue the plot command sequence:

```
PLOT PERSpective   ! requires view information
PLOT HIDE          ! hides interior surfaces.
```

See the plot manual in Appendix E for more information on specifying the perspective view data.

After a solution has been computed, a PostScript file for contour plots may also be obtained. The contours of the vertical displacement for the example problem with the mesh shown in Figure 15.2 may be constructed by issuing the commands:

```
PLOT POSTscript    !open a file to accept plot data
PLOT CONT 2        !plot contours for dof 2
PLOT LOAD,,1       !plot load with tip on nodes
PLOT POSTscript    !close file
```

The CONTour command places the contours for degree-of-freedom 2, while the LOAD places the non-zero loads on the nodes. The results from this sequence are shown in Figure 15.2. To get contours for the velocity or acceleration the CONTour command is replaced by VELOcity or ACCEleration, respectively.

It is also possible to display the full disk using the SYMMetry command. In addition, by adding a parameter to the POSTscript command a border may be added to the display. This is accomplished using the command sequence:

```
PLOT SYMMetry 1 1  !reflect mesh about 1 and 2 coord.
PLOT POSTscript,,1 !open a file to accept plot data
PLOT CONT 2,,1     !plot contours for dof 2
PLOT LOAD,,1       !plot load with tip on nodes
PLOT POSTscript    !close file
```

The results are shown in 15.3.

While the above examples are shown for a BATCh execution, the same sequence may be given from an INTEractive execution. That is, while in an interactive mode issue the command PLOT and the prompt

```
Plot>
```

will appear in the command window. The plot sequence can then be issued one at a time. If any data is required, prompts may be given for the required input Usually, defaults are suggested and may be accepted by pressing the enter key. The need to specify parameters depends on settings of parameters at installation time. It may be necessary to disable or enable use of defaults using the command

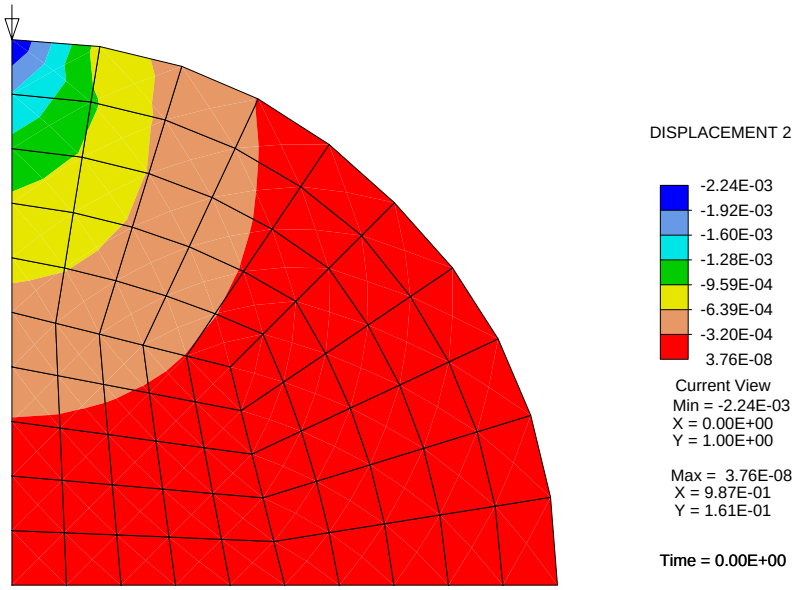


Figure 15.2: Contours of Vertical Displacement for Circular Disk

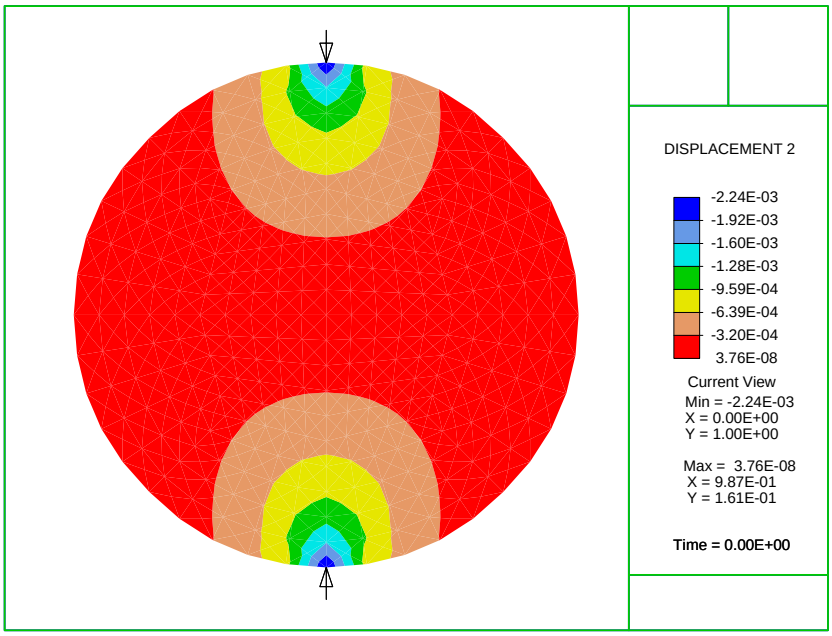


Figure 15.3: Contours of Vertical Displacement for Circular Disk

```
DEFAult <ON,OFF>
```

where either `ON` or `OFF` is selected to enable or disable prompts, respectively [1]. At installation time it is possible to have the parameter defaults either enabled or disabled. The need to specify parameters depends on settings of these parameters at installation time.

15.5 JPEG plots

In the UNIX environment, *FEAP* also permits an option to produce JPEG files of the entire current graphics window. Files are numbered sequentially starting from `Feap00001.jpg`. These can be useful in making animations and presentations, for example using the utility `ffmpeg`. The command is given as

```
PLOT JPEG
```

and, as noted above, places the entire current graphics screen in a numbered file.[2]

[1] Note: The `DEFA`ult command is at the intermediate level and will not appear if the `HELP` command is given at the basic level (i.e., `MANU`al= 0).

[2] To be available, the program must be compiled with an inclusion of the `./plot/jpeg` subdirectory.

Chapter 16

ACKNOWLEDGMENTS

The *FEAP* system has been in continuous development since 1976. The program has been used in the training of a large number of graduate students at the University of California, Berkeley, as well as, at many other institutions worldwide. Numerous contributions have been made to *FEAP* by several individuals during the last several decades. Indeed, without these contributions the program would not have many of the capabilities present today. I am sure that oversights will result in the following acknowledgments – I apologize in advance for the missing ones.

Many improvements related to element technology and solution strategies were contributed by the late Professor Juan C. Simo, both while he was at Berkeley as well as during his time at Stanford University. Juan was in all respects a co-developer of the program during the 1980's and 1990's. The basic strategies for solving non-linear problems resulted from his contributions during our years of interactions. Much of the element technology for finite deformation solid elements for the mixed and enhanced strain are based on his insights, especially his perceptions related to use of three field Hu-Washizu type formulations. The large motion beams and shells also resulted from his research contributions and subsequent contributions by Professor Adnan Ibrahimbegovic and Dr. Ignacio Romero. The coupled flexible-rigid body formulation included in *FEAP* was initiated with Professor Simo and further developed by Dr. Alecia Chen. Simo also added the rotational update routines involving quaternions to support the structural elements and the rigid body work.

The recent modifications for dynamic allocation of arrays and extensions for parallel solution have been developed in collaboration with Professors Sanjay Govindjee (UCB) and David Bindle (Cornell).

Additional improvements to *FEAP* resulted from contributions by former students, visiting scholars, and users of earlier versions of the program. Listed in alphabetical order with current affiliations, the contributors were: Ferdinando Auricchio (University

of Pavia, Italy), Ushnish Basu (LSTC, Livermore, California), the late Jerry Goudreau (Lawrence Livermore National Laboratory), Anna Haraldsson (Hannover University of Technology, Germany), Peter Helnwein, (University of Washington), Tom Hughes (The University of Texas, Austin), Eric Kasper (California State Polytechnic University, San Luis Obispo), Tod Larsen (SUNY), Bahram Nour-Omid, Karl Schweitzerhof (Institute for Mechanics, University of Karlsruhe, Germany), Tom Spelce, Prashanth Vijalapura (Dassault Systémes/Simulia, Providence, RI), Peter Wriggers (Hannover University of Technology, Germany), Giorgio Zavarise (University of Lecce, Italy),

Many additional contributions and suggestions for improvements have been made by Berkeley colleagues Francisco Armero, Filip Filippou, Sanjay Govindjee, and Panayiotis (Panos) Papadopoulos. Their continued interaction and contributions are gratefully acknowledged.

The careful reading of an earlier version of this manual by Dr. Damir Lazarevic of the University of Zagreb in Croatia is gratefully acknowledged. I thank him for this tedious task and hope others will contribute suggestions for additional improvements and corrections.

Finally, I acknowledge the inspiration and guidance of the late Professor Olek Zienkiewicz during our many years of interactions and development of the three volume finite element books. His insights and contributions greatly enhanced finite element analysis methods and provided a motivation for the development of a tool to investigate new areas and methodologies.

To all of the above contributors (and those I have inadvertently failed to cite) I am deeply grateful. Your contributions not only improved *FEAP* but usually led to my better understanding of the issues related to developing software to solve problems in computational mechanics.

Robert L. Taylor
Berkeley, California
June 2020

Appendix A

Mesh manual mages

FEAP has several options which may be used to input data to analyze a wide range of finite element problems in 1 to 3 dimensions. The following pages summarize the commands which are available to input specific parts of the mesh data. Generally the commands discuss the methods to:

1. Input coordinate locations:

```
COORdinate, BLOCk, BLENd,     TRIBlock,
SNODe,      POLAr, SPHErical, SHIFt,    *NODe
```

2. Input element nodal connections and relation to material models:

```
ELEMent, BLOCk, BLENd, TRIBlock, SIDE, *ELEment
```

3. Set boundary and loading conditions:

```
ANGLe,         CANGle,        EANGle,        ROTAte
BOUNdary,      CBOUndary,     EBOUndary
EULEr
FORCe,         CFORce,        EFORce
DISPlacement,  CDISplacement, EDISplacement
TEMPerature,   BTEMperature,  FOLLower,      CSURface,
FPROportional, CPROportional, EPROportional, MPROportional
```

4. Define element types and material parameters:

```
MATErial
```

5. Set group properties:

   `GLOBal, REGIon, FLEXible, TRANsform`

6. Set point stiffness, damping, mass:

   `STIFfness, DAMPing, MASS`

Some other commands exist to control printing to output file (`PRINt, NOPRint`), setting parameter values (`PARA`), and parsing input data (`PARSe, NOPArse`).

Provisions are also available for users to include their own input routines through use of `UMESHn` subprograms. Methods to write and interface user routines to the program are described in the *FEAP* Programmers Manual.

```
feap [ title of problem for printouts, etc.]
     numnp,numel,nummat,ndm,ndf,nen,nad,npd,nud
```

Each problem to be solved by *FEAP* initiates with a single record which contains the characters `FEAP` (either in upper or lower case) as the first entry; the remainder of the record (columns 5-80) may be used to specify a problem title. The title will be printed as the first line of output on each page. The `FEAP` record may be preceded by `PARA`meter specifications (see parameter input manual page).

Immediately following the `FEAP` record the *control* information describing characteristics of the finite element problem to be solved must be given. The *control* information data entries are:

numnp – Total number of nodal points in the problem.
numel – Total number of elements in the problem.
nummat – Number of material property sets in the problem.
ndm – Number of spatial coordinates needed to define mesh.
ndf – Maximum number of degrees-of-freedom on any node.
nen – Maximum number of nodes on any element.
nad – Increases size of element arrays to *ndf*×*nen*+*nad*.
npd – Maximum number of parameters for element properties. (default 200).
nud – Maximum number of parameters for user element properties. (default 50).

For many problems it is not necessary to specify values for *numnp*, *numel*, or *nummat*. *FEAP* can compute the maximum values for each of these quantities. However, for some meshes or when user functions are used to perform the inputs it is necessary to assign the values for these parameters.

The number of spatial coordinates needed to define the finite element mesh *(ndm)* must be 1, 2, or 3. The maximum number of the other quantities is limited only by the size of the dynamically dimensioned array used to store all the data and solution parameters. This is generally quite large and, normally, should not be exceeded. If the error message that memory is exceeded appears the data should be checked to make sure that no errors exist which could cause large amounts of memory to solve the problem. For example, if the error occurs when the `TANG`ent or `UTAN`gent solution macro statements are encountered, the profile of the matrix should be checked for very large column

heights (can be plotted using the PLOT,PROFile command). Appropriate renumbering of the mesh or use of the solution command OPTImize can often significantly reduce the storage required. For symmetric tangent problems the use of the sparse solution routine, which invoked using the solution command DIREct,SPARse, often requires significantly less memory. For some problems with symmetric tangents a solution can be achieved using the iterative conjugate gradient solution method (invoked by the ITERation solution command.

```
*ele = number
```

The `*ELEment` command is used to specify a base value for all subsequently input element quantities. The value may be reset as many times as needed to define a complete mesh. The default value is zero (0). The command usually will be used in conjunction with a `*NODe` command.

It is sometimes necessary to combine mesh data generated in two parts, each of which may number nodes and elements starting with 1,2,3, etc. In this case it will be necessary to use the `*ELEment` command to increment the values related to element numbers on each record.

Consider two parts of a mesh which have been created with element numbers `1` to `E1` for mesh 1 and numbers `1` to `E2` for mesh 2. These are to be combined to form a mesh containing `E1 + E2` elements. The structure for the mesh input would be

```
FEAP * * COMBINE
  ...
INCLude MESH-1 (file with mesh data)

*NODe    = N1 (value for max node    in MESH-1)
*ELEment = E1 (value for max element in MESH-1)

INCLude MESH-2 (file with mesh data)
  ...
```

During the input of the second mesh *FEAP* will add the value of `*ELEment` to each value corresponding to a element number. This generally will be the element connection records; however, it also affects the specification of element region numbers. Note that the `*NODe` command is necessary to add offsets to nodal related numbers.

```
*nod = number
```

The `*NODe` command is used to specify a base value for all subsequently input nodal quantities. The value may be reset as many times as needed to define a complete mesh. The default value is zero (0). The command often will be used in conjunction with a `*ELEment` command.

It is sometimes necessary to combine mesh data generated in two parts, each of which may number nodes and elements starting with 1,2,3, etc. In this case it will be necessary to use the `*NODe` command to increment the values of node numbers on each record.

Consider two parts of a mesh which have been created with node numbers `1` to `N1` for mesh 1 and numbers `1` to `N2` for mesh 2. These are to be combined to form a mesh containing `N1 + N2` nodes. The structure for the mesh input would be

```
FEAP * * COMBINE
  ...
INCLude MESH-1 (file with mesh data)

*NODe    = N1 (value for max node    in MESH-1)
*ELEment = E1 (value for max element in MESH-1)

INCLude MESH-2 (file with mesh data)
  ...
```

During the input of the second mesh *FEAP* will add the value of `*NODe` to each value corresponding to a node number. This could be nodal forces or the specification of node numbers on an element record. Note that the `*ELEment` command is necessary to add offsets to element related numbers.

```
angl
   node1,ngen1,angl(node1)
   node2,ngen2,angl(node2)
   <etc,,terminate with blank record>
```

The `ANGL`e command is used to specify angles (degrees) for sloping nodal boundary conditions as shown in Fig. A.1. For each node `I` to be specified a record is entered with the following information:

node	– the number of the I-node to be specified
ngen	– the increment to the next node, if generation is used, otherwise 0.
angl(node)	– value of angle new 1-coordinate makes with x(1,node).

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown above):

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each angle generated will be a linear interpolation between *node1* and *node2*.

The degrees-of-freedom associated with the sloping boundary may differ from element to element as described in the element manuals. The default will be the first two degrees-of-freedom (2 and 3-D problems) which are affected by the sloping condition. Both force and displacement values will be assumed to be given in the rotated frame. To activate the rotated boundary condition use the `BOUN`dary, `FORC`e, `DISP`lacement, etc. command.

Angle conditions may also be specified using the `EANG`le and `CANG`le commands.

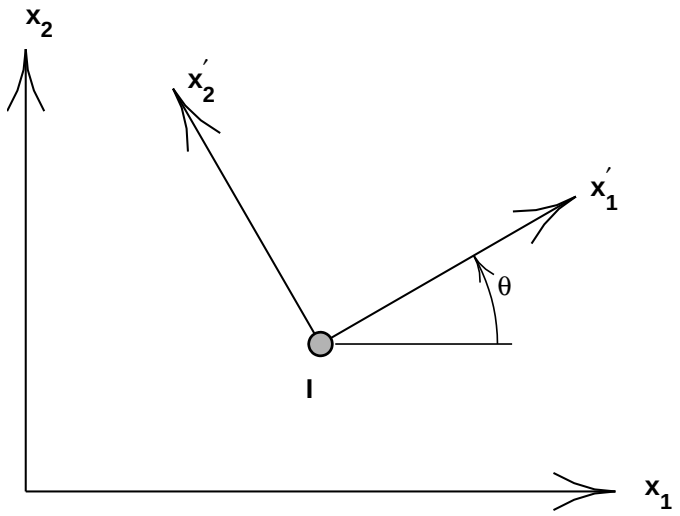


Figure A.1: ANGLe: Coordinate rotation for nodes

Example: ANGLe

As an example consider a problem in which degrees of freedom are to be defined relative to sloping axes. The statements

```
ANGLe
  1 5 30
 21 0 30
```

will define the x_1' axis to make an angle of 30^o with the x_1 axis for nodes 1, 6, 11, 16 and 21. After this command, the first two degrees of freedom will be in the x_1' and x_2' directions, respectively. Also, any specified boundary restraints, forces or displacements will also be with respect to the 30^o rotated axes.

BASE FEAP MESH COMMAND MANUAL

```
base
   node1,ngen1,(bid(i,node1),i=1,ndf)
   node2,ngen2,(bid(i,node2),i=1,ndf)
   <etc.,terminate with blank record>
```

FEAP allows for the solution of problems in which *multiple time history records* are applied as *base* motions to problems which are solved by a modal method. For each such specified point it is necessary to compute a *static* mode for correction to the dynamic states (for problems in which the base degrees of freedom all move together these are merely rigid body motions).

The `BASE` command is used to specify the values for time history records to be used at each degree of freedom. These are given later during solution of the problem as *proportional load tables* (see `COMMAND` language part of manual – `PROPortional loads` page).

For each affected base node a record is entered with the following information:

node	– the number of the node to be specified
ngen	– the increment to the next node, if generation is used, otherwise 0.
bid(1,node)	– value of 1-dof base function for *node*
bid(2,node)	– value of 2-dof base function for *node*
	etc., to *ndf* direction.

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown at top):

node1, node1+ngen1, node1+2×ngen1, ... , node2

BLENd FEAP MESH COMMAND MANUAL

```
blen (Surface in 2 or 3-D)
   surf,r-inc,s-inc,[node1,elmt1,mat],b-type
     (snode(i),i=1,4)
 blen (3-D Solid)
   soli,r-inc,s-inc,t-inc,[node1,elmt1,mat],b-type
     (snode(i),i=1,8)
```

FEAP can generate patches of a mesh using the `BLEN`ding function mesh command. Blending functions are briefly discussed in the Zienkiewicz, Taylor & Zhu: The Finite Element Method, Its Basis & Fundamentals, 6th ed, pp 169 (or 5th ed, volume 1, pp 226 or 4th ed, volume 1, pp 181). Each super node is defined by an input of the following information:

The `BLEN`d data input segment may be used to generate:

1. 3-node triangular elements in 2 or 3-D (b-type = 1 to 6).
2. 4-node quadrilateral elements in 2 or 3-D (b-type = 0).
3. 6-node triangular elements in 2 or 3-D (b-type = 7).
4. 8-node quadrilateral elements in 2 or 3-D (b-type = 8).
5. 9-node quadrilateral elements in 2 or 3-D (b-type = 9).
6. 8-node brick elements in 3-D (b-type = 10 or 0).
7. 4-node tetrahedral elements in 3-D (b-type = 11).
8. 27-node brick elements in 3-D (b-type = 12).
9. 6-node tetrahedral elements in 3-D (b-type = 13).

For surface patches the nodes and quadrilateral elements defined by `BLEN`d command is developed from a master element which is defined by an isoparametric mapping function in terms of the two natural coordinates, r (or ξ_1) and s (or ξ_2), respectively. The node numbers on the master element of each patch defined by `BLEN`d are defined from the values of the four super-nodes used to define the vertices of the blend patch region. The four vertex super-nodes are numbered in any right-hand rule sequence.

The r-direction (ξ_1) is defined along the direction of the first two super-nodes and the s-direction (ξ_2) along the direction of the first and fourth super-nodes. The vertex super-nodes are used as the end nodes which define the four edges of the blend patch. *FEAP* searches the list of edges defined by the `SIDE` command. If a match is found it is used as the patch edge. If no match is found *FEAP* will define a straight edge with linear equal increment interpolation used to define the spacing of nodes in the finite element mesh. Care must be used in defining any specified sides in order to avoid errors from this automatic generation.

For three dimensional solid patches the same technique is used; however, now it is necessary to define eight vertex super-nodes to define the blend patch. The eight nodes are numbered by any right-hand rule sequence. The r-direction and s-direction are defined in the same way as for the surface patch. The third t-direction ξ_3 is along the direction defined by the first to fifth vertex super-node.

The r-, s-, and t-increments are used in the same manner as for the `BLOCk` command. Care must be used in defining the increments along any direction which involves a multi-segment interpolation to ensure that the total number of intervals from the side definition for the multi-segment agrees with the number of increments specified with the `BLENd` command.

Examples for two and three dimensional blends are illustrated in the *FEAP* User Manual.

Since the description of the `BLENd` command depends on existence of `SNODe` and `SIDE` command data, the actual generation of nodes and elements is deferred until the entire mesh data has been defined. Thus, errors may not appear in the output file in the order data was placed in the input file.

BLOCk FEAP MESH COMMAND MANUAL

```
bloc (Line in 1,2,or3-D)
   type,r-inc,,node1,[elmt1,mat,r-skip],b-type
     1,x1,y1,z1 (only ndm coordinates required)
     2,x2,y2,z2
   etc.,blank record after all nodes are input
   bloc (Surface in 2 or 3-D)
   type,r-inc,s-inc,node1,[elmt1,mat,r-skip],b-type
     1,x1,y1,z1 (only ndm coordinates required)
     2,x2,y2,z2
   etc.,blank record after all nodes are input
   bloc (3-D Solid)
   type,r-inc,s-inc,t-inc,node1,[elmt1,mat],b-type
     1,x1,y1,z1
     2,x2,y2,z2
   etc.,blank record after all nodes are input
```

The BLOCk data input segment is used to generate:

1. 2-node line elements in 1, 2, or 3-D.
2. 4, 8, 9 or 16-node quadrilateral elements in 2 or 3-D.
3. 3 or 6-node triangles in 2 or 3-D. For the 3-node elements alternative diagonal directions may be specified as indicated below.
4. 8, 20, 27, 32 or 64-node hexahedra (bricks) in 3-D.
5. 4, 10, 11, 14, or 15-node tetrahedra in 3-D.
6. Nodes only in 1, 2 or 3-D patches.

The patch of nodes and triangular or quadrilateral elements defined by BLOCk is developed from a master element which is defined by an isoparametric 4 to 9 node mapping function in terms of the two natural coordinates, r (or ξ_1) and s (or ξ_2), respectively. The node numbers on the master element of each patch defined by BLOCk are specified according to Figure A.2. The four corner nodes of the master element must be

specified, the mid-point and central nodes are optional. The spacing between the r-increments and s-increments may be varied by an off-center placement of mid-side and central nodes. Thus, it is possible to concentrate nodes and elements into one corner of the patch generated by `BLOCk`. The mid-nodes must lie within the central-half of the r-direction and the s-direction to keep the isoparametric mapping single valued for all (r,s) points. For a line patch, the nodes and 2 node elements are defined from a 1-2 master linear line patch or a 1-3-2 master quadratic line patch. The *s-inc* parameter must be 0 for this option. For a 3-D solid the patch is described by an 8 to 27-node master solid element where the corner nodes are required and mid-edge and mid-face nodes are optional, as is the interior center node (numbering for block nodes is shown in Figures A.3, A.4 and A.5[1]).

The location of nodes on boundaries of adjacent patches should match unless a contact problem is used to determine interactions between bodies. The `TIE` command is used to merge adjacent patches.

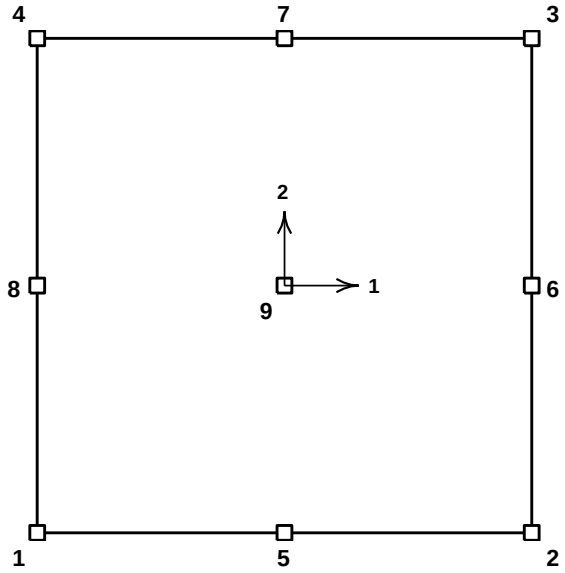


Figure A.2: BLOCk: Node Specification on 2D Master Block.

The data parameters are defined in Tables A.1 and A.2.

An alternative to specifying the *b-type* is to use the set of commands

```
BLOCk
  <CARTesian,...,SURFace>,n1,.....
     <LINE,TRIA,QUAD,TETR,BRIC,USER>,nodes
```

[1]Numbering of block nodes has changed with version 8.3 to agree with the ordering of nodes on standard brick elements. To use the old numbering issue the mesh command as `BLOCk OLD`.

type	– Master node coordinate type (*cart, pola, cyli, sphe* or *surf*).
r-inc	– Number of nodal increments to be generated along r-direction of the patch.
s-inc	– Number of nodal increments to be generated along s-direction of the patch.
t-inc	– Number of nodal increments to be generated along t-direction of the patch (N.B. Input for 3-d blocks only).
node1	– Number to be assigned to first generated node in patch. First node is located at same location as master node 1.
elmt1	– Number to be assigned to first element generated in patch.
matl	– Material identifier to be assigned to all generated elements in patch.
r-skip	– For surfaces, number of nodes to skip between end of an r-line and start of next r-line (default = 1) (N.B. Not input for 3-d block).

Table A.1: BLOCk: Block Numbering Data

```
1  x1 y1 z1
  .....
```

where `LINE` is a *line* or 1-d element; `TRIA` is a triangular element; `QUAD` is a quadrilateral element; `TETR` is a tetrahedrl element; `BRIC` is a brick or hexahedral element; and `USER` is a user defined element. Thus, for example, to define a 5×5 mesh of 6-node triangles one would use the set of commands

```
BLOCK
  CARTesian 10 10
    TRIAngle 6
       1  x1 y1
       2  x2 y2
        etc.
```

Note that the number of increments in each direction must be twice the number of desired elements.

When using the `BLOCk` command one may enter zero for the total number of nodes and elements on the `FEAP` control record. `BLOCk` will automatically generate the correct

b-type	=0: 4-node elements on surface patch; 2-node elements on a line;
	=1: 3-node triangles (diagonals in 1-3 direction of block);
	=2: 3-node triangles (diagonals in 2-4 direction of block);
	=3: 3-node triangles (diagonals alternate 1-3 then 2-4);
	=4: 3-node triangles (diagonals alternate 2-4 then 1-3);
	=5: 3-node triangles (diagonals in union-jack pattern);
	=6: 3-node triangles (diagonals in inverse union-jack pattern);
	=7: 6-node triangles (similar to =1 orientation);
	=8: 8-node quadrilaterals (*r-inc* and *s-inc* must be even numbers); N.B. Interior node generated but not used;
	=9: 9-node quadrilaterals (*r-inc* and *s-inc* must be even numbers);
	=16: 16-node quadrilaterals (*r-inc* and *s-inc* must be multiples of three);
	=10: 8-node hexahedra (bricks).
	=11: 4-node tetrahedra.
	=12: 27-node quadratic hexahedra. (r-, s-, t-inc must be even numbers)
	=13: 10-node tetrahedra. (r-, s-, t-inc must be even numbers)
	=14: 20-node quadratic hexahedra. (r-, s-, t-inc must be even numbers)
	=15: 11-node quadratic tetrahedra. (r-, s-, t-inc must be even numbers)
	=17: 14-node quadratic tetrahedra. (r-, s-, t-inc must be even numbers)
	=18: 15-node quadratic tetrahedra. (r-, s-, t-inc must be even numbers)
	=19: 64-node cubic hexahedra. (r-, s-, t-inc must be

Table A.2: BLOCk: Block Type Data

number of nodes and elements. If it is desired to use block to generate nodal coordinates only, the value of *elmt1* should be entered as a negative number.

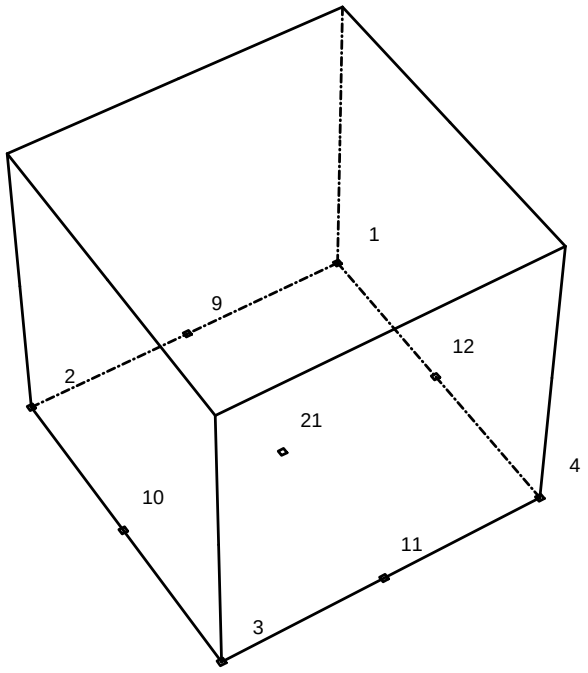


Figure A.3: BLOCk: Lower Node Specification on 3D Master Block.

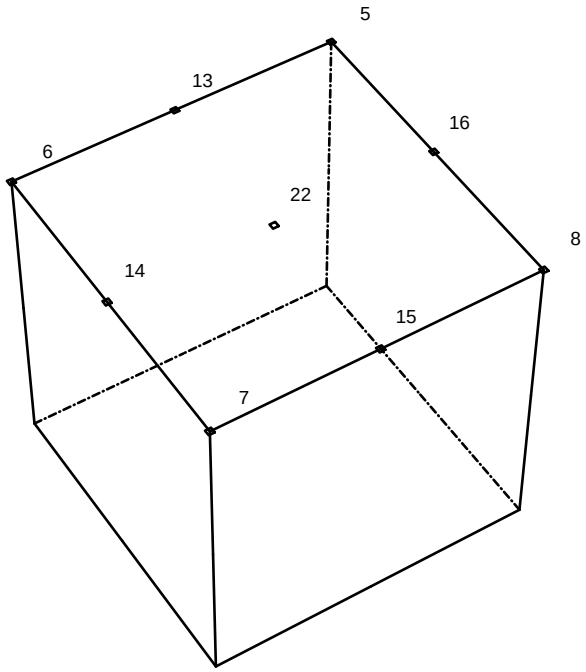


Figure A.4: BLOCk: Top Node Specification on 3D Master Block.

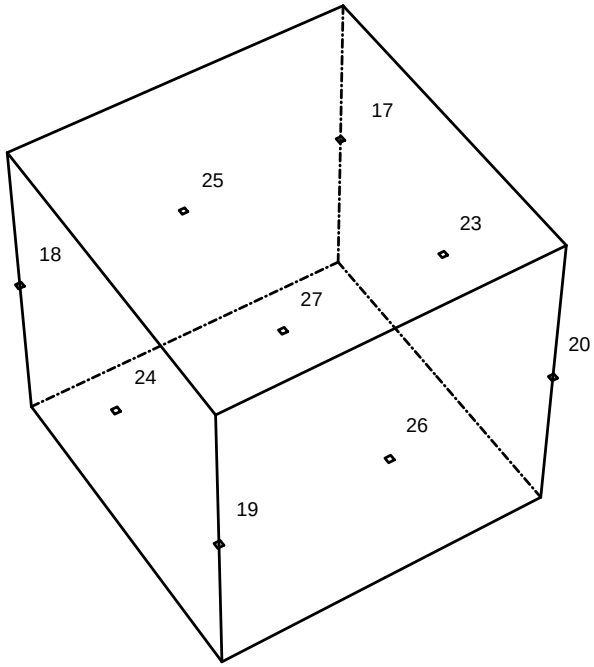


Figure A.5: BLOCk: Middle Node Specification on 3D Master Block.

```
boun
   node1,ngen1,(id(i,node1),i=1,ndf)
   node2,ngen2,(id(i,node2),i=1,ndf)
   <etc.,terminate with blank record>
```

The `BOUN`dary command is used to specify the values for the boundary restraint conditions. For each node to be specified a record is entered with the following information:

node	– the number of the node to be specified
ngen	– the increment to the next node, if generation is used, otherwise 0.
id(1,node)	– value of 1-dof boundary restraint for *node*
id(2,node)	– value of 2-dof boundary restraint for *node*
	etc., to *ndf* direction.

The boundary restraint codes are interpreted as follows:

id(i,node) $= 0$	a force will be an applied load to dof (default).
id(i,node) $\neq 0$	a displacement will be imposed to dof.

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown at top):

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each boundary restraint will be as follows:

id(i,node1) =	0 or positive	→ id(i,node1+ngen1) = 0
id(i,node1) =	negative	→ id(i,node1+ngen1) = −1

With this convention the value of a zero *id(i,node2)* will be set negative whenever the value of *id(i,node1)* starts negative. Accordingly, it is necessary to assign a positive value for the restraint code to terminate a generation sequence (e.g., when it is no longer desired to set a dof to be restrained). Alternatively, an i-dof may be eliminated for all nodes by using the generation sequence:

node	ngen	dofs				
		1	...	i	...	ndf
1	1	0	...	-1	...	0
numnp	0	0	...	+1	...	0

Subsequent records may then be used to assign values to other degree-of-freedoms.

Boundary condition restraints may also be specified using the `EBOU`nd or `CBOU`nd commands.

Example: BOUNdary

Consider a problem which has 3 degrees of freedom at each node. The sequence of records:

```
BOUNdary conditions
  1 4 1 -1 0
 13 0 0  1 1
```

will define boundary conditions for nodes 1, 5, 9 and 13 and the restraint codes will have the following values

node	DOFS		
	1	2	3
1	1	-1	0
5	0	-1	0
9	0	-1	0
13	0	1	1

Any degree of freedom with a non-zero boundary code will be *restrained*, whereas a degree of freedom with a zero boundary code will be *unrestrained*. Restrained degrees of freedom may have specified non-zero (generalized) *displacements* whereas unrestrained ones may have specified non-zero (generalized) *forces*.

```
btem
   nodes,r-inc,s-inc,t-inc,node1,[r-skip]
     1,t1
     2,t2
   etc.,until all 'nodes' records are input
```

The BTEMperature data input segment is used to generate a bloc of temperatures on a regular one, two or three dimensional patch of nodes. Temperatures specified by BTEM command are passed to the elements in the tl array (see programmers manual).

Temperatures using the BTEM command are generated by interpolating specified nodal temperatures using the standard isoparametric interpolation:

$$T^a = N_I(\boldsymbol{\xi}^a)\, T_I$$

where $N_I(\boldsymbol{\xi})$ are shape functions for the patch, $\boldsymbol{\xi}^a$ are the parent coordinates (ξ_1, ξ_2, ξ_3) at node a, and T_I are the patch temperature at patch node I.

For two dimensions, the patch of nodes defined by BTEMperature is developed from a master element which is defined by an isoparametric 4-9 node mapping function in terms of the parent coordinates ξ_1 (generates r-inc spaced values) and ξ_2 (generates s-inc spaced values). The node numbers on the master element of each patch defined by BTEM are specified according to Figure A.2 in the BLOCk manual page. The four corner nodes of the master element must be specified, the mid-point and central node are optional. For this case *t-inc* is set to 0.

For three dimensions the patch is an 8-27 node brick where the first 8-nodes are at the corners and the remaining nodes are mid-edge, mid-face, and interior nodes. The first 8-nodes must be specified. The block master nodes are numbered as shown in Figures A.3, A.4 and A.5 in the BLOCk manual page.

The data parameters are defined as:

nodes -- Number of master nodes needed to define the patch.
r-inc -- Number of nodal increments to be generated along ξ_1 or r-direction of the patch.
s-inc -- Number of nodal increments to be generated along ξ_2 or s-direction of the patch.
t-inc -- Number of nodal increments to be generated along ξ_3 or t- direction of the patch (default = 0).
node1 -- Number to be assigned to first node in patch (default = 1). First node is located at same location as master node 1.
r-skip -- Number of nodes to skip between end of an r-line and start of next r-line (may be used to interconnect blocks side-by-side) (default = 1)

```
cang
   node,(x(i),i=1,ndm),angle
   linear
     1,x1,y1,angle1
     2,x2,y2,angle2
   quadratic
     1,x1,y1,angle1
     2,x2,y2,angle2
     3,x3,y3,angle3
   surface
     1,x1,y1,z1,angle1
     2,x2,y2,z2,angle2
     3,x3,y3,z3,angle3
     4,x4,y4,z4,angle4
   cartesian
   pola,x0,y0
   gap,value
   <etc.,terminate with a blank record>
```

The angle of a sloping boundary condition may be set using the reference coordinates for a single *node*, a *linear* line, a *quadratic* line or a *surface* patch. The input values are saved in a file(s) and searched *after* the entire mesh is specified. The data is order dependent with data defined by `ANGL`e processed first, `EANG`le processed second and the `CANG`le data processed last. The value defined last is used for any analysis. The data input by `CANG` replaces previously assigned values. After use files are deleted automatically. Coordinate systems for the global and rotated axes are shown in Fig. A.6.

For a single *node*, the data to be supplied during the definition of the mesh consists of:

node	– Defines inputs to be for a *node*
x(1)	– Value of coordinates to be used during search
...	(a tolerance of about 1/1000 of mesh size is
x(ndm)	used during search, coordinate with smallest distance within tolerance is assumed to have specified value).
angle	– Value of the angle (in degrees)

At execution, the node(s) within the tolerance will have their values set to the sloping condition. For nodes with sloping conditions, the degrees-of-freedom are expressed with respect to the rotated frame instead of the global frame. For three dimensional problems the 3-direction coincides with the x3-direction.

For two dimensional problems it is possible to specify a segment to which the rotation angle is applied. The segment may be specified as a *linear* or a *quadratic* line. For the *linear* segment the angle is given together with the coordinates of the ends. These are specified as:

```
CANGle
  LINEar
    1,x1,y1,angle1
    2,x2,y2,angle1
```

For *quadratic* segments the ends *(x1,y1)* and *(x2,y2)* together with an intermediate point *(x3,y3)* are used. The quadratic segment is given as:

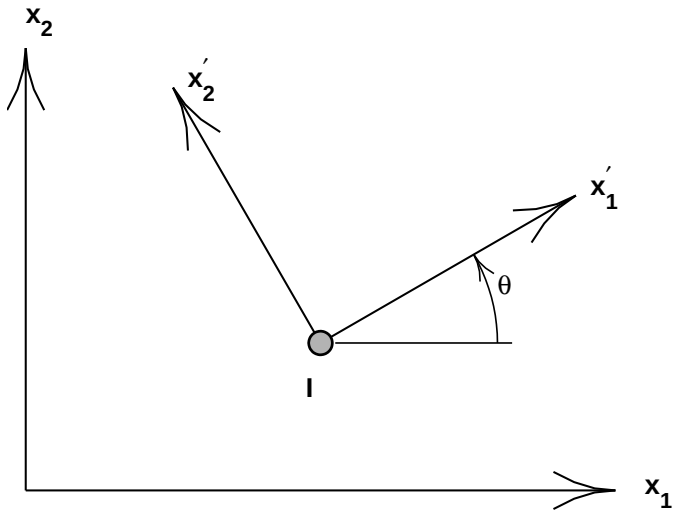


Figure A.6: CANGle: Coordinate rotation for nodes

```
CANGle
  QUADratic
    1,x1,y1,angle1
    2,x2,y2,angle2
    3,x3,y3,angle3
```

For three dimensional problems it is possible to specify the segment to which the boundary conditions are applied. The segment is specified as a *surface*. The data is specified as:

```
CANGle
  SURFace
    1,x1,y1,z1,angle1
    2,x2,y2,z2,angle2
    3,x3,y3,z3,angle3
    4,x4,y4,z4,angle4
```

The surface type (`LINEar, QUADratic, SURFace` may be repeated several times within one `CANGle` data set. Alternatively, up to 99 data sets may be used. Each data set should terminate with a blank record.

By default the specification of coordinates is given by their Cartesian values. It is possible to specify the input coordinates in a cylindrical coordinate system in which `xi = ri`, `yi =` θ_i and `zi = zi`. The inputs are given as

```
CANGle
  POLAr x0 y0 z0
  LINEar
    1 r1 theta1 z1 angle1
    2 r2 theta2 z2 angle2
```

To return to a Cartesian mode the command `CARTesian` is given before a segment type data set.

The program assigns a search region and attempts to find the elements and the nodes to which the specified segments are associated. It is possible that no segment is located (an error message will appear in the output file). To expand the search region a *gap* can be specified as:

```
CANGle
```

```
GAP,value
 (type and coordinate inputs)
```

The gap-value is a coordinate distance within which nodes are assumed to lie on the specified segment type. The value should be less than dimensions of typical elements or erroneous nodes will be found by the search. It is suggested that the computed boundary conditions be checked graphically to ensure that they are correctly identified (e.g., use PLOT,MESH and PLOT,BOUNdary to show the locations of conditions).

The value of the angle at individual nodes may be input using the CANGle command. In a two dimensional problem the input is given as

```
CANGle
  NODE  x y angle
```

In a three dimensional problem the data set is given as

```
CANGle
  NODE  x y z angle
```

where angle denotes a rotation in the x-y plane. Several NODE statements may be included in each CANGle data set. Each data set should terminate with a blank record.

The node *closest* to this point will be selected. This can be sensitive to roundoff if two nodes are at *equal* distances from the specified point (the GAP option is not active during a nodal search). Users should check (using graphics plot mode) that the correct node(s) are selected.

Example: CANGle

In a two-dimensional problem a rotated coordinate system of 45^o for a node located *close* to the coordinates $x_1 = 0$ and $x_2 = 5$ is desired. The data may be specified without needing to know a number for the node using the commands:

```
CANGle
  NODE  0 5 45
```

```
cbas
   gap,value
   node,(x(i),i=1,ndm),(bid(j),j=1,ndf)
   <etc.,terminate with a blank record>
```

FEAP allows for the solution of problems in which *multiple time history records* are applied as *base* motions to problems which are solved by a modal method. For each such specified point it is necessary to compute a *static* mode for correction to the dynamic states (for problems in which the base degrees of freedom all move together these are merely rigid body motions).

The specified base funcions may be set using the reference coordinates for a *node*. The input values are saved in files and searched after the entire mesh is specified. After use files are deleted. The data is order dependent with data defined by `BASE` processed first, `EBASe` processed second and the `CBASe` processed last. The value defined last is used for any analysis.

For each *node*, the data to be supplied during the definition of the mesh consists of:

node	– Defines inputs to be for a *node*.
x(1)	– value of coordinates to be used during search
...	(a gap of 1/1000 of mesh size is used during
x(ndm)	search, coordinate with smallest distance within
	gap is assumed to have specified value).
bid(1)	– base function on 1-dof
bid(2)	– base function on 2-dof
...	
bid(ndf)	

To expand the search region a *gap-value* can be specified as:

```
GAP,value
```

The `gap-value` is a coordinate distance within which nodes are assumed to lie on the specified segment. The value should be less than dimensions of typical elements or erroneous nodes will be found by the search.

For Example:

```
cbas
  node 1.0 1.0 1 1 2
           ! end with blank record
```

assigns the base functions to the node located at (1.0,1.0).

```
cbou,[set,add]
   node,(x(i),i=1,ndm),(ibc(j),j=1,ndf)
   linear,(ibc(j),j=1,ndf)
     1,x1,y1
     2,x2,y2
   quadratic,(ibc(j),j=1,ndf)
     1,x1,y1
     2,x2,y2
     3,x3,y3
   surface,(ibc(j),j=1,ndf)
     1,x1,y1,z1
     2,x2,y2,z2
     3,x3,y3,z3
     4,x4,y4,z4
   cartesian
   pola,x0,y0
   gap,value
   <etc.,terminate with a blank record>
```

The boundary restraint conditions may be set using the reference coordinates for a single *node*, a *linear* line, a *quadratic* line or a *surface* patch. The input values are saved in files and searched after the entire mesh is specified. The data is order dependent with data defined by `BOUN`ary processed first, `EBOU`dary processed second and the `CBOU`dary data processed last. The value defined last is used for any analysis. After use files are deleted automatically.

The `CBOU` command may be used with two options. Using the `CBOU SET` option replaces all previously defined conditions at any node by the pattern specified. This is the default mode. Using the `CBOU ADD` option accumulates the specified boundary conditions with previously defined restraints.

For a single *node*, the data to be supplied during the definition of the mesh consists of:

node	– Defines inputs to be for a *node*.
x(1)	– Value of coordinates to be used during search
...	(a tolerance of about 1/1000 of mesh size is
x(ndm)	used during search, coordinate with smallest distance within tolerance is assumed to have specified value).
ibc(1)	– Restraint conditions for all nodes with value
ibc(2)	of search. (0 = active dof, >0 or <0 denotes
...	a fixed dof).
ibc(ndf)	

For two dimensional problems it is possible to specify the segment to which the boundary conditions are applied. The segment may be specified as a *linear* or a *quadratic* line. For the *linear* segment the boundary condition pattern are given together with the coordinates of the ends. These are specified as:

```
CBOUndary
  LINEar,(ibc(i),i=1,ndf)
    1,x1,y1
    2,x2,y2
```

For *quadratic* segments the ends *(x1,y1)* and *(x2,y2)* together with an intermediate point *(x3,y3)* are used. The *quadratic* segment is given as:

```
CBOUndary
  QUADratic,(ibc(i),i=1,ndf)
    1,x1,y1
    2,x2,y2
    3,x3,y3
```

For three dimensional problems it is possible to specify the segment to which the boundary conditions are applied. The segment is specified as a *surface*. The data is specified as:

```
CBOUndary
  SURFace,(ibc(i),i=1,ndf)
    1,x1,y1,z1
    2,x2,y2,z2
    3,x3,y3,z3
    4,x4,y4,z4
```

The surface type (`LINEar, QUADratic, SURFace` may be repeated several times within one `CANGle` data set. Alternatively, up to 99 data sets may be used.

By default the specification of coordinates is given by their Cartesian values. It is possible to specify the input coordinates in a cylindrical coordinate system in which `xi = ri`, `yi =` θ_i and `zi = zi`. The inputs are given as

```
CBOUndary
  POLAr x0 y0 z0
  LINEar (ibc(i),i=1,ndf)
    1 r1 theta1 z1 angle1
    2 r2 theta2 z2 angle2
```

To return to a Cartesian mode the command `CARTesian` is given before a segment type data set.

The program assigns a search region and attempts to find the elements and the nodes to which the specified segments are associated. It is possible that no segment is located (an error message will appear in the output file). To expand the search region a *gap* can be specified as:

```
CBOUndary
  GAP,value
   (type and coordinate inputs)
```

The `gap-value` is a coordinate distance within which nodes are assumed to lie on the specified segment. The value should be less than dimensions of typical elements or erroneous nodes will be found by the search. It is suggested that the computed boundary conditions be checked graphically to ensure that they are correctly identified (e.g., use `PLOT,MESH` and `PLOT,BOUN` to show the locations of conditions).

The value of boundary restraint codes at individual nodes may be input using the `CBOU` command. In a two dimensional problem the input is given as

```
CBOUndary
  NODE  x y (ibc(i),i=1,ndf)
```

In a three dimensional problem the data set is given as

```
CANGle
  NODE  x y z (ibc(i),i=1,ndf)
```

where `ibc` denotes the list of boundary restrainit codes Several `NODE` statements may be included in each `CBOU` data set.

The node *closest* to the specified point will be selected. This can be sensitive to roundoff if two nodes are at *equal* distances from the specified point (the `GAP` option is not active during a nodal search). Users should check (using graphics plot mode) that the correct node(s) are selected.

```
cdam
   gap,value
   node,(x(i),i=1,ndm),(d(j),j=1,ndf)
   <etc.,terminate with a blank record>
```

Specified nodal dampers may be input using the reference coordinates for a *node*. The input values are saved in files and searched after the entire mesh is specified. After use files are deleted.

For a *node*, the data to be supplied during the definition of the mesh consists of:

node	– Defines inputs to be for a *node*.
x(1)	– value of coordinates to be used during search
...	(a gap of 1/1000 of mesh size is used during
x(ndm)	search, coordinate with smallest distance within
	gap is assumed to have specified value).
c(1)	– damper on 1-dof
c(2)	– damper on 2-dof
...	
c(ndf)	

To expand the search region a *gap-value* can be specified as:

```
GAP,value
```

The `gap-value` is a coordinate distance within which nodes are assumed to lie on the specified segment. The value should be less than dimensions of typical elements or erroneous nodes will be found by the search.

For Example (2-d problem):

```
cdam
  node 1.0 30.0 5.0 10.0
           ! end with blank record
```

places a damper at coordinate (1, 20) with value 5 for dof 1 and 10 for dof 2. Units are in values for the problem.

```
cdis,[set,add]
   gap,value
   node,(x(i),i=1,ndm),(d(j),j=1,ndf)
   <etc.,terminate with a blank record>
```

The specified displacement boundary conditions may be set using the reference coordinates for a *node*. The input values are saved in files and searched after the entire mesh is specified. After use files are deleted. The data is order dependent with data defined by `DISP`lacement processed first, `EDIS`placement processed second and the `CDIS`placement data processed third and data specified by the `CSUR`face processed last. The value defined last is used for any analysis.

The `CDIS` command may be used with two options. Using the `CDIS,SET` option replaces all previously defined values at any node by the pattern specified. This is the default mode. Using the `CDIS,ADD` option accumulates the specified value with previously defined values.

For a *node*, the data to be supplied during the definition of the mesh consists of:

node	– Defines inputs to be for a *node*.
x(1)	– value of coordinates to be used during search
...	(a gap of 1/1000 of mesh size is used during
x(ndm)	search, coordinate with smallest distance within
	gap is assumed to have specified value).
d(1)	– displacement on 1-dof
d(2)	– displacement on 2-dof
...	
d(ndf)	

To expand the search region a *gap-value* can be specified as:

```
GAP,value
```

The `gap-value` is a coordinate distance within which nodes are assumed to lie on the specified segment. The value should be less than dimensions of typical elements or erroneous nodes will be found by the search. It is suggested that the computed

boundary conditions be checked graphically to ensure that they are correctly identified (e.g., use `PLOT,MESH` and `PLOT,BOUN` to show the locations of conditions).

While it is possible to specify both the force and the displacement applied to a node, only one can be active during a solution step. The determination of the active value is determined from the boundary restraint condition value. If the boundary restraint value is zero and you use one of the force-commands a force value is imposed, whereas, if the boundary restraint value is non-zero and you use one of the displacement-commands a displacement value is imposed. (See `BOUN`dary, `CBOU`ndary, or `EBOU`ndary pages for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value. It is not possible, to specify a displacement by using the combination of a force-command with a non-zero boundary restraint value, as it was in earlier releases of *FEAP*.

Only those values of the `CDIS`placement-command are regarded whose directions have a non-zero boundary restraint value. All other displacement values are variable.

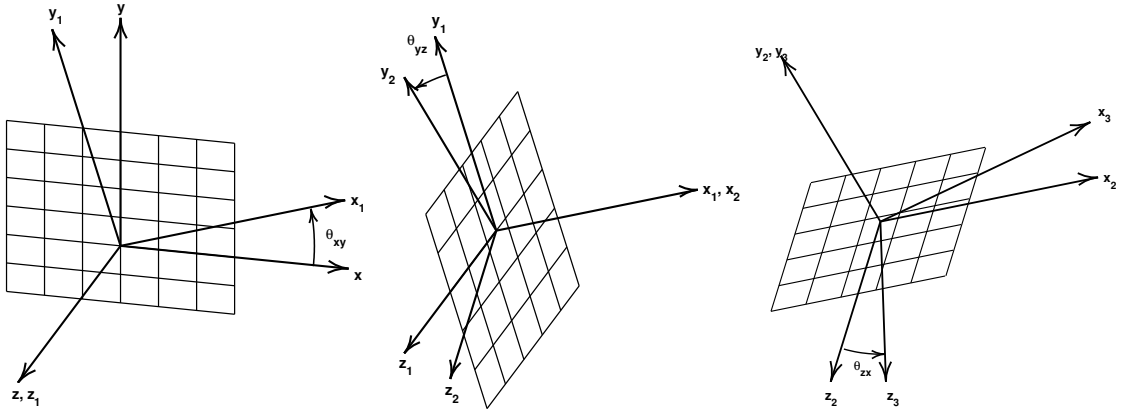
For Example:

```
cang
  node,1.0,1.0,30.0
           ! end with blank record
cbou
  node,1.0,1.0,0,1
           ! end with blank record
cdis
  node,1.0,1.0,0.1,0.1
           ! end with blank record
```

Here the first displacement value is not considered. There is a displacement of the node with the coordinates (1.0,1.0). The direction of the displacement is 120 degrees and the value is 0.1. The displacement in the 30 degree direction is variable.

```
ceul
   node (x(i),i=1,3),theta_xy,theta_yz,theta_zx
   <etc,,terminate with blank record>
```

For three-dimensional problems, the `CEULer` command is used to specify the Euler angles (degrees) for nodal boundary conditions oriented as shown in Fig. A.7. For each node `I` to be specified a record is entered with the following information:



(a) Rotation θ_{xy} (b) Rotation θ_{yz} (c) Rotation θ_{zx}

Figure A.7: CEULer: Euler angle rotations for nodes.

node	– character name 'node'
x(i)	– coordinate for component "i"
theta_xy(node)	– value of rotation angle about z-axis
theta_yz(node)	– value of rotation angle about x-axis
theta_zx(node)	– value of rotation angle about y-axis

The degrees-of-freedom associated with the sloping boundary may differ from element to element as described in the element manuals. The default will be the first three degrees-of-freedom (3-D problems) which are affected by the sloping condition. Only displacement values will be assumed to be given in the rotated frame. To activate the rotated boundary condition use the `BOUN`dary, `DISP`lacement, etc. command.

```
cfor,[set,add]
   gap,value
   node,(x(i),i=1,ndm),(f(j),j=1,ndf)
   <etc.,terminate with a blank record>
```

The specified force boundary conditions may be set using the reference coordinates for a *node*. The input values are saved in files and searched after the entire mesh is specified. After use files are deleted. The data is order dependent with data defined by `FORCe` processed first, `EFORce` processed second and the `CFORce` data processed last. The value defined last is used for any analysis.

The `CFORce` command may be used with two options. Using the `CFOR,SET` option replaces all previously defined forces at any node by the pattern specified. This is the default mode. Using the `CFOR,ADD` option accumulates the specified forces with previously defined values.

For a *node*, the data to be supplied during the definition of the mesh consists of:

node	– Defines inputs to be for a *node*.
x(1)	– value of coordinates to be used during search
...	(a gap of 1/1000 of mesh size is used during
x(ndm)	search, coordinate with smallest distance within
	gap is assumed to have specified value).
f(1)	– force on 1-dof
f(2)	– force on 2-dof
...	
f(ndf)	

To expand the search region a *gap-value* can be specified as:

```
GAP,value
```

The `gap-value` is a coordinate distance within which nodes are assumed to lie on the specified segment. The value should be less than dimensions of typical elements or erroneous nodes will be found by the search. It is suggested that the computed loads be checked graphically to ensure that they are correctly identified (e.g., use `PLOT,MESH` and `PLOT,LOAD` to show the locations of conditions).

While it is possible to specify both the force and the displacement applied to a node, only one can be active during a solution step. The determination of the active value is determined from the boundary restraint condition value. If the boundary restraint value is zero and you use one of the force-commands a force value is imposed, whereas, if the boundary restraint value is non-zero and you use one of the displacement-commands a displacement value is imposed. (See `BOUN`dary, `CBOU`ndary, or `EBOU`ndary pages for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value.

```
cmas
   gap,value
   node,(x(i),i=1,ndm),(d(j),j=1,ndf)
   <etc.,terminate with a blank record>
```

Specified nodal mass may be input using the reference coordinates for a *node*. The input values are saved in files and searched after the entire mesh is specified. After use files are deleted.

For a *node*, the data to be supplied during the definition of the mesh consists of:

node	– Defines inputs to be for a *node*.
x(1)	– value of coordinates to be used during search
...	(a gap of 1/1000 of mesh size is used during
x(ndm)	search, coordinate with smallest distance within
	gap is assumed to have specified value).
m(1)	– mass on 1-dof
m(2)	– mass on 2-dof
...	
m(ndf)	

To expand the search region a *gap-value* can be specified as:

```
GAP,value
```

The `gap-value` is a coordinate distance within which nodes are assumed to lie on the specified segment. The value should be less than dimensions of typical elements or erroneous nodes will be found by the search.

For Example (2-d problem):

```
CMASs
  NODE 1.0 30.0 5.0 10.0
           ! end with blank record
```

places a mass at coordinate (1, 30) with value 5 for dof 1 and 10 for dof 2. Units are in values for the problem.

```
cpro
   node,(x(i),i=1,ndm),(pnum(i),i=1,ndf)
   linear (pnum(i),i=1,ndf)
     1,x1,y1
     2,x2,y2
   quadratic (pnum(i),i=1,ndf)
     1,x1,y1
     2,x2,y2
     3,x3,y3
 surface (pnum(i),i=1,ndf)
     1,x1,y1,z1
     2,x2,y2,z2
     3,x3,y3,z3
     4,x4,y4,z4
   cartesian
   pola,x0,y0
   gap,value
   <etc.,terminate with a blank record>
```

The proportional loading number to be applied to nodal forces and displacements may be input using this command. The input values are saved in a file(s) and searched after the entire mesh is specified. The data is order dependent with data defined by FPROportional processed first, EPROportional processed second and the CPROportional data processed last. The value defined last is used for any analysis.

For a single *node*, the data to be supplied during the definition of the mesh consists of:

node	– Defines inputs to be for a *node*
x(1)	– Value of coordinates to be used during search
...	(a tolerance of about 1/1000 of mesh size is
x(ndm)	used during search, coordinate with smallest distance within tolerance is assumed to have specified value).
pnum(1,node)	– Proportional load number of dof-1
pnum(2,node)	– Proportional load number of dof-2
	etc., to *ndf* directions

At execution, the node(s) within the tolerance will have their values set to the proportional load numbers given.

For two dimensional problems it is possible to specify a segment to which the proportional load numbers are to be applied. The segment may be specified as a *linear* or a *quadratic* line. For the *linear* segment the angle is given together with the coordinates of the ends. These are specified as:

```
LINEar (pnum(i),i=1,ndf)
  1,x1,y1
  2,x2,y2
```

For *quadratic* segments the ends *(x1,y1)* and *(x2,y2)* together with an intermediate point *(x3,y3)* are used. The quadratic segment is given as:

```
QUADratic (pnum(i),i=1,ndf)
  1,x1,y1
  2,x2,y2
  3,x3,y3
```

For three dimensional problems it is possible to specify the segment to which the proportional load numbers are applied. The segment is specified as a *surface*. The data is specified as:

```
SURFace (pnum(i),i=1,ndf)
  1,x1,y1,z1
  2,x2,y2,z2
  3,x3,y3,z3
  4,x4,y4,z4
```

The program assigns a search region and attempts to find the elements and the nodes to which the specified segments are associated. It is possible that no segment is located (an error message will appear in the output file). To expand the search region a *gap* can be specified as:

```
GAP,value
```

The `gap-value` is a coordinate distance within which nodes are assumed to lie on the specified segment. The value should be less than dimensions of typical elements or erroneous nodes will be found by the search. It is suggested that the computed

boundary conditions be checked graphically to ensure that they are correctly identified (e.g., use `PLOT,MESH` and `PLOT,BOUN`dary to show the locations of conditions).

The *polar* option may be used to set the origin *(x0,y0)* of a polar coordinate system. Coordinates entered after *polar* will be assumed to be radius and angle. The *cartesian* option resets the coordinate system to a cartesian frame.

Example: CFORce

In a two dimensional problem it is desired to have a time variation for the force applied to the node nearest to the coordinates $x_1 = 10$ and $x_2 = 5$ which is different in the two directions. To prescribe the data it is necessary to define three different command sets. The first defines the *magnitude* of the two forces at the node. This may be given as:

```
CFORce
  NODE 10 5  8.5  -6.25
```

in which $F_1 = 8.5$ and $F_2 = -6.25$. The second command set describes the *numbers* for proportional loading factors which will multiply each of the forces. These may be given as:

```
CPROportional
  NODE 10 5  2  3
```

where 2 is the proportional loading number 2 and 3 that for 3. Finally, during solution mode the proportional loads must be given. This is best included in a `BATCh` solution mode as:

```
BATCh
  PROP,,2
END
 data for proportional load 2 (see PROP in solution commands)
```

and

```
BATCh
  PROP,,3
END
 data for proportional load 3 (see PROP in solution commands)
```

Failure to specify correctly any of the above will usually result in an error.

COORdinates FEAP MESH COMMAND MANUAL

```
coor,<all>
   node1,ngen1,(x(i,node1),i=1,ndm)
   node2,ngen2,(x(i,node2),i=1,ndm)
   <etc.,terminate with blank record>
```

The COORdinate command is used to specify the values for nodal coordinates. For each node to be specified a record is entered with the following information:

node	– Number of node to be specified
ngen	– Increment to next node, if generation is used, otherwise 0.
x(1,node)	– Value of coordinate in 1-direction
x(2,node)	– Value of coordinate in 2-direction
	etc., to 'ndm' directions

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown above):

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values generated for each coordinate will be a linear interpolation between *node1* and *node2*.

The COORdinate values may be input in a polar or spherical coordinate system and converted to cartesian values later using the POLAr or SPHErical commands.

Nodal coordinates may also be generated using the BLOCk and the BLENd commands.

Example: COORdinate

The set of commands:

```
COORdinates
  1 1  0.0  0.0
 11 0 10.0  5.0
```

will generate 11 nodes equally spaced along the straight line connecting the points (0, 0) and (10, 5). The nodes will be numbered from 1 to 11.

The use of the `COORd ALL` form requires all coordinate records to be created (e.g., by a mesh generation program) in full numerical form. The input will be performed without parsing and without any offset by a `*NOD` value.

```
csti
   gap,value
   node,(x(i),i=1,ndm),(d(j),j=1,ndf)
   <etc.,terminate with a blank record>
```

Specified nodal stiffness to ground may be input using the reference coordinates for a *node*. The input values are saved in files and searched after the entire mesh is specified. After use files are deleted.

For a *node*, the data to be supplied during the definition of the mesh consists of:

node	– Defines inputs to be for a *node*.
x(1)	– value of coordinates to be used during search
...	(a gap of 1/1000 of mesh size is used during
x(ndm)	search, coordinate with smallest distance within
	gap is assumed to have specified value).
k(1)	– stiffness on 1-dof
k(2)	– stiffness on 2-dof
...	
k(ndf)	

To expand the search region a *gap-value* can be specified as:

```
GAP,value
```

The `gap-value` is a coordinate distance within which nodes are assumed to lie on the specified segment. The value should be less than dimensions of typical elements or erroneous nodes will be found by the search.

For Example (2-d problem):

```
csti
  node 1.0 30.0 5.0 10.0
           ! end with blank record
```

places a stiffness to ground at coordinate (1, 20) with value 5 for dof 1 and 10 for dof 2. Units are in values for the problem.

```
csur
   linear
     1,x1,y1,p1
     2,x2,y2,p2
   quadratic
     1,x1,y1,p1
     2,x2,y2,p2
     3,x3,y3,p3
   surface
     1,x1,y1,z1,p1
     2,x2,y2,z2,p2
     3,x3,y3,z3,p3
     4,x4,y4,z4,p4
   disp,udof
   flux,tdof
   normal
   tangential,sdof
   traction,sdof
   polar,x0,y0
   cartesian
   gap,value
   <terminate with a blank record>
```

A mesh may be generated in *FEAP* in which it is desired to specify distributed loading, displacements or thermal flux on parts of the body. For two dimensional problems it is possible to specify the surface to which the boundary condition is applied using the `CSUR`face command (The command is for Coordinate specified SURfaces.). The input values are saved in files and searched after the entire mesh is input (i.e., after the `END` mesh command. After use files are deleted. The data is order dependent with data defined by other options. Surface data is always generated last.

The type of input to be generated is set using the `DISPlacement`, `FLUX`, `NORMal`, or `TANGential` options. These specify that inputs will be a specific flux (`tdof` specifies the degree of freedom for temperature – default is 1), displacement component (`udof`),

normal traction component (`sdof`), or tangential component (`sdof`), respectively. The default is `NORMal` loading. For displacement, tangential and traction inputs the component to be generated is specified immediately after the command type.

A two-dimensional surface may be specified as a `LINEar` or a `QUADratic` line. For the linear surface the values at the ends `p1`, `p2` are given together with the end coordinates `(x1,y1)` and `(x2,y2)`. These are specified as:

```
LINEar
  1,x1,y1,p1
  2,x2,y2,p2
```

For quadratic line surfaces the ends (nodes 1 and 2) together with an intermediate point are used. Thus it is possible to have quadratic variation of the values. The quadratic surface is given as:

```
QUADratic
  1,x1,y1,p1
  2,x2,y2,p2
  3,x3,y3,p3
```

For three dimensional problems it is possible to specify the surface area to which the quantities are applied. The area is specified as `SURFace` followed by the coordinate data specified as:

```
SURFace
  1,x1,y1,z1,p1
  2,x2,y2,z2,p2
  3,x3,y3,z3,p3
  4,x4,y4,z4,p4
```

The program assigns a search region and attempts to find the elements and the nodes to which the specified surfaces are associated. It is possible that no surface is located (an error message will appear in the output file). To expand the search region a `GAP` can be specified as:

```
GAP,value
```

The `gap-value` is a coordinate distance within which nodes are assumed to lie on the specified surface. The value should be less than dimensions of typical element or

erroneous surfaces will be found by the search. It is suggested that the computed loads be checked graphically to ensure that they are correctly identified (e.g., use `PLOT,MESH` and `PLOT,LOAD` to show the locations of computed loads).

The `POLAr` option may be used to set the origin of a polar coordinate system. Coordinates entered after `POLAr` will be assumed to be radius and angle. The `CARTesian` option resets the coordinate system to a cartesian frame. The default mode is `CARTesian`.

The nodes 1, 2 (and 3 and 4 if required) must be input in the right order. The normal vector of the surface has to point outward from the surface as defined by a right-hand rule.

```
damp
 node1,ngen1,(c(i,node2),i=1,ndf)
 node2,ngen2,(c(i,node2),i=1,ndf)
 <etc.,terminate with blank record>
```

The DAMPer command is used to specify the values for linear nodal dampers to earth. For each node on which non-zero values are to be specified a record is entered with the following information:

node	– Number of the node to be specified
ngen	– Increment to the next node, if generation is used, otherwise 0.
c(1,node)	– Value of the damper in 1-dof
c(2,node)	– Value of the damper in 2-dof
	etc., to *ndf* directions

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown at top):

$$node1,\ node1+ngen1,\ node1+2\times ngen1,\ \ldots.\ ,\ node2$$

The values for each damper will be a linear interpolation between the *node1* and *node2* values.

Example: DAMPer

A damping in the vertical direction is to be specified for node 15. The damping factor is 1250 and given for this case by the commands:

```
DAMPer
  15 0 0 1250
```

The first zero indicates that no generations are to follow. The second zero indicates no damping for the horizontal (1^{st}) direction.

DEBUg FEAP MESH COMMAND MANUAL

```
debug,,ndebug
debug,on,ndebug
debug,off
```

Use of the `DEBUg,ON,ndebug` or `DEBU,,ndebug` command enables internal prints controlled by the `DEBU`g parameter in `common /debugs/ ndebug,debug`.

The `ndebug` parameter is provided to allow setting of different levels for displaying prints. The debug print option is disabled using a `DEBUg,OFF` command

```
disp
   node1,ngen1,(d(i,node1),i=1,ndf)
   node2,ngen2,(d(i,node2),i=1,ndf)
   <etc.,terminate with blank record>
```

The `DISP`lacement command is used to specify the values for nodal boundary displacements. For each node to be specified a record is entered with the following information:

node	– Number of node to be specified
ngen	– Increment to next node, if generation is used, otherwise 0.
d(1,node)	– Value of displacement for 1-dof
d(2,node)	– Value of displacement for 2-dof
	etc., to *ndf* directions

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown at top):

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each displacement will be a linear interpolation between the *node1* and *node2* values for each degree-of-freedom.

While it is possible to specify both the force and the displacement applied to a node, only one can be active during a solution step. The determination of the active value is determined from the boundary restraint condition value. If the boundary restraint value is zero and you use one of the force-commands a force value is imposed, whereas, if the boundary restraint value is non-zero and you use one of the displacement-commands a displacement value is imposed. (See `BOUN`dary, `CBOU`ndary, or `EBOU`ndary pages for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value. It is not possible, to specify a displacement by using the combination of a force-command with a non-zero boundary restraint value, as it was in the last releases of *FEAP*. For further information see the `CDIS`placement page.

Displacement conditions may also be specified using the `EDIS` and `CDIS` commands.

```
eang
   i-coor,xi-value,angle
   <etc.,terminate with a blank record>
gap value
```

The sloping boundary condition angle may be set along any set of nodes which has a constant value of the *i-coordinate direction* (e.g., 1-direction (or x), 2-direction (or y), etc.). A search for a node within a *gap* is made to set the conditions. By default the gap is taken as $[max(x_i) - min(x_i)] * 10^{-3}$; however this value may be reset by inserting (after the `EANG` command and before the list of coordinate locations) the statement:

```
GAP value
```

In this case the search will be made for values within the distance specified by `value`.

The data to be supplied during the definition of the mesh consists of:

i-coor	– Direction of coordinate (i.e., 1 = x, 2 = y, etc.)
xi-value	– Value of i-direction coordinate to be used during search (a tolerance of 1/1000 of mesh size is used during search, any coordinate within the gap is assumed to have the specified value).
angle	– Value 1-direction makes with x1-direction in degrees.

For nodes with sloping conditions, the degrees-of-freedom are expressed with respect to the rotated frame 1-2 instead of the global frame x1-x2 (x-y). For three dimensional problem the 3-direction coincides with the x3-direction (z).

Angle conditions may also be specified using the `EANG`le and `CANG`le commands. The data is order dependent with data defined by `ANGL`e processed first, `EANG`le processed second and the `CANG`le data processed last. The value defined last is used for any analysis.

Example: EANGle

All the nodes located on the $x_3 = z = 0$ plane are to have degrees of freedom specified relative to a rotated coordinate system (about the x_3-axis). This is not a common case but may be specified using the command set:

```
EANGle
  3 0.0 40.0
```

where 40.0 is the angle (in degrees) of the rotation. Rotation is defined by right-hand screw rule.

If the default tolerance is too small to find all the desired nodes for the $x_3 = z = 0.0$ plane the command set

```
EANGle
  GAP 1.e-5
  3 0.0 40.0
```

may be used. In this case all the nodes which satisfy $z = 0.0 \pm 10^{-5}$ will be included in the search.

```
ebas
   i-coor,xi-value,(bid(j),j=1,ndf)
   <etc.,terminate with a blank record>
gap value
```

FEAP allows for the solution of problems in which *multiple time history records* are applied as *base* motions to problems which are solved by a modal method. For each such specified point it is necessary to compute a *static* mode for correction to the dynamic states (for problems in which the base degrees of freedom all move together these are merely rigid body motions).

The values of boundary displacement conditions may be set along any set of nodes which has a constant value of the *i-coordinate direction* (e.g., 1-direction (or x), 2-direction (or y), etc.). A search for a node within a *gap* is made to set the conditions. By default the gap is taken as $[max(x_i) - min(x_i)] * 10^{-3}$; however this value may be reset by inserting (after the `EBASe` command and before the list of coordinate locations) the statement:

```
GAP value
```

In this case the search will be made for values within the distance specified by `value`.

The data to be supplied during the definition of the mesh consists of:

i-coor	– Direction of coordinate (i.e., 1 = x, 2 = y, etc.)
xi-value	– Value of i-direction coordinate to be used during search (a tolerance of 1/1000 of mesh size is used during search, any coordinate within the gap is assumed to have the specified value).
bid(1)	– Base load function on dof 1
bid(2)	– Base load function on dof 2
...	
bid(ndf)	

For further information see the `BASE` page.

Proportional load functions may also be specified using the `BASE` and `CBASe` commands. The data is order dependent with data defined by `BASE` processed first, `EBASe` processed

second and the `CBASe` data processed last. The value defined last is used for any analysis.

Example: EBASe

All the nodes located on the $x_3 = z = 0$ plane are to have the base load functions set. These may be set using the commands

```
EBASe
  3 0.0  1 1 2
```

If the default tolerance is too small to find all the desired nodes for the $x_3 = z = 0.0$ plane the command sets

```
EBASe
  GAP 1.e-5
  3 0.0  1 1 2
```

may be used. In this case all the nodes which satisfy $z = 0.0 \pm 10^{-5}$ will be included in the search.

```
ebou,[set,add]
   i-coor,xi-value,(ibc(j),j=1,ndf)
   <etc.,terminate with a blank record>
gap value
pola x0 y0
```

The boundary restraint conditions may be set along any set of nodes which has a constant value of the *i-coordinate direction* (e.g., 1-direction (or x), 2-direction (or y), etc.). A search for a node within a *gap* is made to set the conditions. By default the gap is taken as $[max(x_i) - min(x_i)] * 10^{-3}$; however this value may be reset by inserting (after the `EBOU` command and before the list of coordinate locations) the statement:

```
GAP value
```

In this case the search will be made for values within the distance specified by `value`.

The option `POLAr x0 y0` performs subsequent transformations in cylindrical coordinates centered at `x0 y0`. The direction parameter should be 1 and *xi-value* is interpretted as radius. The z-coordinate is ignored during the search

The coordinate data to be supplied during the definition of the mesh consists of:

i-coor	– Direction of coordinate (i.e., 1 = x, 2 = y, etc.)
xi-value	– Value of i-direction coordinate to be used during search (a tolerance of 1/1000 of mesh size is used during search, any coordinate within the gap is assumed to have the specified value).
ibc(1)	– Restraint conditions for all nodes with value of
ibc(2)	search.(0 = boundary code remains as previously set
...	> 0 denotes a fixed dof, < 0 resets previously
ibc(ndf)	defined boundary codes to 0.)

The option `POLAr x0 y0` performs subsequent transformations in cylindrical coordinates centered at `x0 y0`. The direction parameter should be 1 and *xi-value* is interpretted as radius. The z-coordinate is ignored during the search

The `EBOU` command may be used with two options. Using the `EBOU,SET` option replaces previously defined conditions at any node by the pattern specified. Using the

`EBOU,ADD` option accumulates the specified boundary conditions with previously defined restraints. The default mode is `ADD`. Boundary restraint conditions may also be specified using the `BOUN` and `CBOU` commands. The data is order dependent with data defined by `DISP` processed first, `EDIS` processed second and the `CDIS` data processed last. The value defined last is used for any analysis.

Example: Cartesian

All the nodes located on the $x_3 = z = 0.1$ plane are to have restraints on the 3^{rd} and 6^{th} degrees of freedom. This may be specified using the command set:

```
EBOUndaray
  3 0.1 0 0 1 0 0 1
```

where non-zero values indicate a *restrained* degree of freedom and a *zero* an unrestrained degree of freedom. Non-zero displacements may be specified for restrained dof's and non-zero forces for unrestrained dof's.

If the default tolerance is too small to find all the desired nodes for the $x_3 = z = 0.1$ plane the command set

```
EBOUndaray
  GAP 1.e-5
  3 0.1 0 0 1 0 0 1
```

may be used. In this case all the nodes which satisfy $z = 0.1 \pm 10^{-5}$ will be included in the search.

Example: Cylindrical/Polar

All nodes located at a radius $r = 0.75$ and centered at $x_0 = 0.2$ and $y_0 = 0.5$ may be assigned fixed values using the command sequence

```
EBOUndaray
  POLAr 0.2 0.5
   1 0.75 1 1 1
```

```
edis
   i-coor,xi-value,(d(j),j=1,ndf)
   <etc.,terminate with a blank record>
gap value
```

The values of boundary displacement conditions may be set along any set of nodes which has a constant value of the *i-coordinate direction* (e.g., 1-direction (or x), 2-direction (or y), etc.). A search for a node within a *gap* is made to set the conditions. By default the gap is taken as $[max(x_i) - min(x_i)] * 10^{-3}$; however this value may be reset by inserting (after the `EDIS` command and before the list of coordinate locations) the statement:

```
GAP value
```

In this case the search will be made for values within the distance specified by `value`.

The data to be supplied during the definition of the mesh consists of:

i-coor	– Direction of coordinate (i.e., 1 = x, 2 = y, etc.)
xi-value	– Value of i-direction coordinate to be used during search (a tolerance of 1/1000 of mesh size is used during search, any coordinate within the gap is assumed to have the specified value).
d(1)	– Value of displacement for dof's
d(2)	
...	
d(ndf)	

While it is possible to specify both the force and the displacement applied to a node, only one can be active during a solution step. The determination of the active value is determined from the boundary restraint condition value. If the boundary restraint value is zero and you use one of the force-commands a force value is imposed, whereas, if the boundary restraint value is non-zero and you use one of the displacement-commands a displacement value is imposed. (See `BOUN`dary, `CBOU`ndary, or `EBOU`ndary pages for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value. It is not possible, to specify

a displacement by using the combination of a force-command with a non-zero boundary restraint value, as it was in the last releases of *FEAP*. For further information see the `CDIS`placement page.

Displacement conditions may also be specified using the `DISP` and `CDIS` commands. The data is order dependent with data defined by `DISP` processed first, `EDIS` processed second and the `CDIS` data processed last. The value defined last is used for any analysis.

Example: EDISplacement

All the nodes located on the $x_3 = z = 0$ plane are to have a vertical displacement (2^{nd} dof) of -0.25 units. This may be set using the commands

```
EDISplacement
  3 0.0  0.0 -0.25
```

In addition it is necessary to specify boundary restraint codes for the nodes to which the condition is to be applied. A simple way to do this is to use the command set:

```
EBOUndary
  3 0.0  0 1
```

Of course the horizontal (1^{st}) dof could be restrained for any of the nodes also.

If the default tolerance is too small to find all the desired nodes for the $x_3 = z = 0.0$ plane the command sets

```
EBOUndaray
  GAP 1.e-5
  3 0.0 0.0 -0.25

EBOUndary
  GAP 1.e-5
  3 0.0  0 1
```

may be used. In this case all the nodes which satisfy $z = 0.0 \pm 10^{-5}$ will be included in the search.

```
efor,[set,add]
   i-coor,xi-value,(f(j),j=1,ndf)
   <etc.,terminate with a blank record>
gap value
```

The values of boundary force conditions may be set along any set of nodes which has a constant value of the *i-coordinate direction* (e.g., 1-direction (or x), 2-direction (or y), etc.). A search for a node within a *gap* is made to set the conditions. By default the gap is taken as $[max(x_i) - min(x_i)] * 10^{-3}$; however this value may be reset by inserting (after the `EFOR` command and before the list of coordinate locations) the statement:

```
GAP value
```

In this case the search will be made for values within the distance specified by `value`.

The data to be supplied during the definition of the mesh consists of:

i-coor	– Direction of coordinate (i.e., 1 = x, 2 = y, etc.)
xi-value	– Value of i-direction coordinate to be used during search (a tolerance of 1/1000 of mesh size is used during search, any coordinate within the gap is assumed to have the specified value).
f(1)	– Value of force for dof's
f(2)	
...	
f(ndf)	

While it is possible to specify both the force and the displacement applied to a node, only one can be active during a solution step. The determination of the active value is determined from the boundary restraint condition value. If the boundary restraint value is zero and you use one of the force-commands a force value is imposed, whereas, if the boundary restraint value is non-zero and you use one of the displacement-commands a displacement value is imposed. (See `BOUN`dary, `CBOU`ndary, or `EBOU`ndary pages for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value.

The `EFOR` command may be used with two options. Using the `EFOR,SET` option replaces previously defined forces at a node by the pattern specified. Using the `EFOR,ADD` option accumulates the forces with previously defined values. The default mode is `ADD`.

Force conditions may also be specified using the `FORC`e and `CFOR`ce commands. The data is order dependent with data defined by `FORC`e processed first, `EFOR`ce processed second and the `CFOR`ce data processed last. The value defined last is used for any analysis.

Example: EFORce

All the nodes located on the $x_3 = z = 10$ plane are to have a common specified horizontal force value. (Note that this is not a common case as end nodes on equally spaced intervals would have different values from other nodes.) This may be specified using the command set:

```
EFORce
  3 10.0 -12.5
```

where -12.5 is the value of each force

If the default tolerance is too small to find all the desired nodes for the $x_3 = z = 10.0$ plane the command set

```
EFORce
  GAP 1.e-5
  3 10.0 -12.5
```

may be used. In this case all the nodes which satisfy $z = 10.0 \pm 10^{-5}$ will be included in the search.

ELEMent FEAP MESH COMMAND MANUAL

```
elem,<all>
   nelm1,ngen1,matl1,(ix(i,nelm1),i=1,nen)
   nelm2,ngen2,matl2,(ix(i,nelm2),i=1,nen)
   <etc.,terminate on blank record>
 elem node=n mate=m type=t
 elem,old
   nelm1,matl1,(ix(i,nelm1),i=1,nen),ngen1
   nelm2,matl2,(ix(i,nelm2),i=1,nen),ngen2
   <etc.,terminate on blank record>
```

The ELEMent command is used to specify values of nodal numbers which are attached to an element. The command may appear more than once during mesh inputs. It may also be combined with BLOCk and BLENd inputs to generate elements in a mesh. For each element to be specified by an ELEMent command, a record is entered with the following information:

nelm	– Number of the element to be specified
ngen	– Value to increment each node-i value when generation is used (default = 1).
matl	– Material identifier (*id* of material) for the element, this will determine the element type.
ix(1,nelm)	– Node-1 number attached to element.
ix(2,nelm)	– Node-2 number attached to element. etc., to *nen* nodes.

Element inputs must be in increasing values for *nelm*. If gaps occur in the input order generation is performed, the element number sequence will be in increments of 1 from *nelm1* to *nelm2*; the nodes which are generated for each intermediate element will be as follows:

```
ix(i,nelm1+1) = ix(i,nelm1) + ngen1
```

except

```
ix(i,nelm1+1) = 0  whenever ix(i,nelm1) = 0
```

The program assumes that any zero value of an *ix(i,nelm)* indicates that no node is attached at that point.

Input terminates whenever a blank record is encountered.

ADVICE: When the number of elements on the control record is input as zero *FEAP* attempts to compute the number of elements in the mesh. The number computed is the largest number input by an `ELEMent` input or during a `BLOCk` and `BLENd` generation. During `ELEMent` input it is necessary to input the last element in generation sequences.

When analyses are performed using different types of elements the `ELEMent` command has optional parameters which may be used to simplify the input specification. For example in a 3-d analysis using 27-node brick `SOLId` elements in which the boundary is loaded by 9-node `PRESsure` elements specification for the solid elements requires two records due to the limit of 16-items per record for *FEAP* inputs. However, for the pressure elements the input for 9-nodes will all be on the first record and the second record becomes an inconvenience. This may be avoided by splitting the element inputs into two groups using the commands:

```
ELEMents NODEs=27
  e1 g1 m1 n1 n2 n3      ....  n13
     n14 n15   .....           n27
  e2 g2 m1 n1 n2 n3      ....  n13
     n14 n15   .....           n27
     etc.
! End of solid elements
ELEMents NODEs=9
  e1 g1 m1 n1 n2 n3      ....  n9
  e1 g2 m2 n1 n2 n3      ....  n9
     etc.
! End of pressure elements
```

Another option that may be added to the `ELEMent` record is the material number to be used for the group. This may be added to the above `NODE` parameter or given alone. The form is

```
ELEMents MATE=ma
```

where `ma` is either a parameter of a numeric value.

A third parameter that describes the element type may also be assigned using

Type	Description
`line`	Line elements
`tria`	Triangular elements
`quad`	Quadrilateral elements
`tetr`	Tetrahedron elements
`hexa`	Hexahedron elements
`wedg`	Wedge elements
`pyra`	Pyramid elements
`poin`	Point elements

Table A.3: ELEMent: Different types for elements

```
ELEMent TYPE=t
```

where `t` is a 4-character name describing the type of element input. This can assist *FEAP* in distinguishing the difference between elements with the same number of nodes. The current options for `t` are:

An example is:

```
ELEMents NODEs=6 TYPE=triangle
```

The use of the `ELEMent ALL` form requires all element records to be created (e.g., by a mesh generation program) in full numerical form. The input will be performed without parsing and without any offset by a `*ELEment` or `*AUTo` value.

END FEAP MESH COMMAND MANUAL

```
end
```

The last mesh command must be `END`. This terminates the mesh input and returns to the control program, which may then perform additional tasks on the data or `STOP` execution.

Immediately following the `END` mesh command any additional data required to manipulate the mesh (e.g., `TIE`, `LINK`, `ELINk`, `PARTition` `ORDEr`, `RIGId` and `JOINt` should be given prior to initiation of a problem solution using `BATCh` and/or `INTEractive`.

```
epro
   i-coor,xi-value,(pnum(i),i=1,ndf)
   <etc.,terminate with a blank record>
gap value
```

The proportional loading number to be applied to nodal forces and displacements may be input using this command. The number may be set along any set of nodes which has a constant value of the *i-coordinate direction* (e.g., 1-direction (or x), 2-direction (or y), etc.). A search for a node within a *gap* is made to set the conditions. By default the gap is taken as $[max(x_i) - min(x_i)] * 10^{-3}$; however this value may be reset by inserting (after the `EPRO` command and before the list of coordinate locations) the statement:

```
GAP value
```

In this case the search will be made for values within the distance specified by `value`.

The data to be supplied during the definition of the mesh consists of:

i-coor	– Direction of coordinate (i.e., 1 = x, 2 = y, etc.)
xi-value	– Value of i-direction coordinate to be used during search (a tolerance of 1/1000 of mesh size is used during search, any coordinate within the gap is assumed to have the specified value).
pnum(1,node)	– Proportional load number of dof-1
pnum(2,node)	– Proportional load number of dof-2
	etc., to *ndf* directions

For nodes with sloping conditions, the degrees-of-freedom are expressed with respect to the rotated frame 1-2 instead of the global frame x1-x2 (x-y). For three dimensional problem the 3-direction coincides with the x3-direction (z).

Proportional load numbers may also be specified using the `FPROp` and `CPROp` commands. The data is order dependent with data defined by `FPROp` processed first, `EPROp` processed second and the `CPROp` data processed last. The value defined last is used for any analysis.

```
ereg
   elem1,ngen1,reg1
   elem2,ngen2,reg2
   <etc.,terminate with blank record>
```

The EREGion command is used to specify the region number for elements. For each element to be specified a record is entered with the following information:

elem	– Number of the element to be specified
ngen	– Increment to the next element, if generation is used, otherwise 0.
reg	– Region number to be assigned

When generation is performed, the element number sequence will be

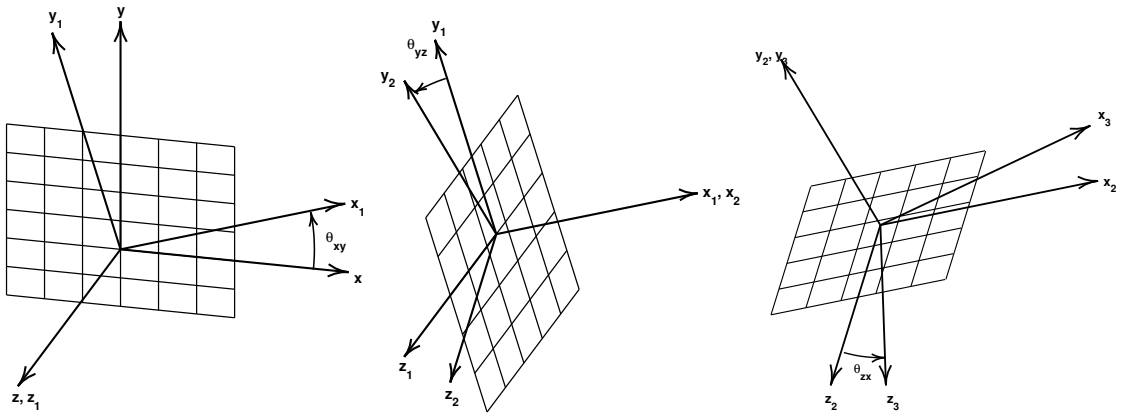
elem1, elem1+ngen1, elem1+2×ngen1, ... , elem2

The generated element are assigned to *reg1*.

Region numbers may also be assigned to element groups using the REGIon mesh command.

```
eule
   node1,ngen1,angl(node1)
   node2,ngen2,angl(node2)
   <etc,,terminate with blank record>
```

For three-dimensional problems, the `EULEr` command is used to specify the Euler angles (degrees) for nodal boundary conditions oriented as shown in Fig. A.8. For each node `I` to be specified a record is entered with the following information:



(a) Rotation θ_{xy} (b) Rotation θ_{yz} (c) Rotation θ_{zx}

Figure A.8: EULEr: Euler angle rotations for nodes.

node	– the number of the I-node to be specified
ngen	– the increment to the next node, if generation is used, otherwise 0.
theta_xy(node)	– value of rotation angle about z-axis
theta_yz(node)	– value of rotation angle about x-axis
theta_zx(node)	– value of rotation angle about y-axis

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown above):

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each angle generated will be a linear interpolation between *node1* and *node2*.

The degrees-of-freedom associated with the sloping boundary may differ from element to element as described in the element manuals. The default will be the first three degrees-of-freedom (3-D problems) which are affected by the sloping condition. Only displacement values will be assumed to be given in the rotated frame. To activate the rotated boundary condition use the `BOUN`dary, `DISP`lacement, etc. command.

`expl mate ma it` `expl elem`

FEAP permits solution of transient problems using a mix of implicit time integration and explicit time integration parts. Unless the entire solution is by an explicit method all elements are integrated using the specified implicit method. Parts of the mesh may be designated to be solved by an explicit method using one of two forms. All elements associated to a particular material set may be integrated by the explicit method using the mesh command

```
EXPLicit MATErial ma it
```

where `ma` is the material set number and `it` the number of iterations for which the explicit part is updated. If a lumped (diagonal) mass is used for both the tangent and the residual convergence occurs in 1 iteration and, thus, the value of `it` may be omitted. However, if the residual is updated using a consitent mass additional numbers of iterations may be specified (for *FEAP* this may occur for isogeometric formulations).

The second option is to specify a specific list of elemenhts to be integrated by the explicit method. Currently, all elements must be specified independently, hence the form of the input is

```
EXPLicit ELEMent
  n1 i1
  n2 i2
    ... etc.
    ! Terminate with a blank line
```

```
filt mate ma
filt region re
filt off
```

FEAP permits inputs to filter input commands such that they are associated with elements that are defined by a single *material set number* or a single *region number*.

Material set numbers are the value set during input of `ELEMent`, `BLOCk`, or `BLENd` command sets. The filtering applies to subsequent commands that are based on edge or coordinate values (e.g., `EFORce`, etc.).

To associate filtering based on a material set. the command is given by

```
FILTer MATErial 2

EBOUndary
  2 40.0  1 0

FILTer OFF
```

This set would look at all elements belonging to material set 2 and apply a boundary condition `(0, 1)` to any node in the set that had a coordinate `y = 40.0`.

To associate filtering based on `REGIon` definitions the filtering command set is initiated by

```
FILTer REGIon re
```

where `re` is the number of a `REGion`.

`flex`

FEAP permits portions of a mesh to be declared as a rigid body. During the generation of the mesh it is necessary to designate which elements will belong to a rigid body and which elements remain flexible. By default all elements are flexible. However, if a group of elements has been declared to be rigid (using the `RIGId` command) it is then necessary to insert a flexible command before generating additional flexible elements. This is accomplished by inserting a record `FLEXible` before groups of elements which will remain deformable.

Example:

```
FLEXible
ELEMents
  1, ....
        ! Blank terminator
```

The command may also be inserted before a `BLOCk` or `BLENd` command and may be used as many times as necessary. By default all elements are flexible.

Note: It is also necessary to use the `RIGId` mesh manipulation command to activate the rigid bodies and to assign additional parameters. See also the `JOINt` command for methods to interconnect rigid bodies.

```
forc
   node1,ngen1,(f(i,node1),i=1,ndf)
   node2,ngen2,(f(i,node2),i=1,ndf)
   <etc.,terminate with blank record>
```

The FORCe command is used to specify the values for nodal boundary forces. For each node to be specified a record is entered with the following information:

node	– Number of node to be specified
ngen	– Increment to next node, if generation is used, otherwise 0.
f(1,node)	– Value of force for 1-dof
f(2,node)	– Value of force for 2-dof
	etc., to *ndf* directions

When generation is performed, the node number sequence for *node1-node2* sequence shown at top will be:

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each force will be a linear interpolation between the *node1* and *node2* values for each degree-of-freedom.

While it is possible to specify both the force and the displacement applied to a node, only one can be active during a solution step. The determination of the active value is determined from the boundary restraint condition value. If the boundary restraint value is zero and you use one of the force-commands a force value is imposed, whereas, if the boundary restraint value is non-zero and you use one of the displacement-commands a displacement value is imposed. (See BOUNdary, CBOUndary, or EBOUndary pages for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value.

Force conditions may also be specified using the EFORce and CFORce commands. The data is order dependent with data defined by FORCe processed first, EFORce processed second and the CFORce data processed last. The value defined last is used for any analysis.

Example: FORCe

A concentrated force is to be applied to nodes 10 and 15. The force at node 10 has values of 100.0 in the horizontal direction and 0 in the vertical direction; whereas the force at node 15 has a magnitude of 200 and makes an angle of 60^o with the horizontal axis. These two forces may be specified using the command set:

```
FORCe
  10  0  100.0          0.0
  15  0  200*cosd(60) 200*sind(60)
```

Note the use of the built-in functions available in *FEAP* to compute the horizontal and vertical components.

```
fpro
   node1,ng1,(pnum(i,node1),i=1,ndf)
   node2,ng2,(pnum(i,node2),i=1,ndf)
   <etc.,terminate with blank record>
```

The FPROportional factors command is used to specify the proportional load numbers for forced nodal conditions. For each node a record is entered with the following information:

node	– Number of node
ng	– Generator increment to next node
pnum(1,node)	– Proportional load number of dof-1
pnum(2,node)	– Proportional load number of dof-2
	etc., to *ndf* directions

The proportional load numbers are interpreted as follows:

pnum(i,node) = 0	dof-i uses sum of specified proportional load factors
pnum(i,node) not 0	dof-i uses specified proportional load based on order of solution inputs prop (default = 1. if prop not used).

As a default all *pnum* values are set to zero (0) and individual proportional load factors to 1.

Generation is performed similar to FORCe input. Thus

```
FPROportional
   1 5 0 1
  21 0 1 2
```

would generate nodes 6, 11, 16 with proportional load number 1 assigned to the second degree of freedom; node 21 would have proportional load 1 for the first degree of freedom and 2 for the second degree of freedom.

Proportional loading numbers may also be specified using the EPROp and CPROp commands. The data is order dependent with data defined by FPROp processed first, EPROp processed second and the CPROp data processed last. The value defined last is used for any analysis.

```
glob
   equation,,g_eq g_part
   plane stress
   plane strain
   axisymmetric <torsion>
   small
   finite
   temp,dof,value
   refe,node,(x(i),i=1,ndm)
   refe,vect,(v(i),i=1,ndm)
   omega,<cycle>,omega
   omega,node,(x(i),i=1,ndm)
   omega,vector,(n(i),i=1,ndm)
   scale,<coor,time,mass>,value)
   scale,disp,(v(i),i=1,ndf)
   scale,elem,j,(v(i),i=1,j)
```

The `GLOBal` command is used to set parameters which apply to all elements.

The option*equation* designates the number of global equations in a problem and has two prameters: *g_eq* the number of equations and *g_part* the partition to which the equations belong.

Use of *plane stress* sets all 2-d elements to compute properties based on the plane stress assumption; use of *plain strain* sets the properties for plane strain condition; and *axisymmetric* sets the geometry to an axisymmetric condition. If the ¡*torsion*¿ option is added three degrees of freedom are required, with the third degree of freedom used for the tangential motion required to add the torsion effects.

The option *small* designates a small deformation solution option for all elements (this is the default mode); whereas, the option *finite* designates a finite deformation solution mode (at present only the two dimensional solid element supports this option - in displacement mode).

The *temperature dof* option designates the global degree of freedom (i.e., the *value* dof) which is to be used by the solid and structural element to extract the temperatures for use in computing thermal strains. This is used for coupled thermo-mechanical solutions

in which the temperatures are computed using a thermal element (e.g., the *thermal* element type specified by the MATErial set command).

The *reference node* option defines a coordinate location to be used to orient the cross section of three dimensional FRAMe elements. The 2-axis is directed from the center of the beam toward the node location. The *reference vector* option defines a vector to be used to orient the cross section of three dimensional FRAMe elements. A cross product of the vector with the axis of the frame element defines the 1-axis of the cross section. The 2-axis is then constructed by another cross product between the 1-axis and the frame element axis.

Rotating three dimensional bodies may be analyzed using the global OMEGA commands. The command without an option specifies the rotational valocity in radians/time and with the CYCLe option in cycles/time. The command with option NODE defines a location of a fixed point and the VECTor option the orientation of the fixed axis. Each element then computes the body force and, for finite deformation problems, the tangent matrix for the rotational effects.

Most global parameters may be superseded by specifying a different condition during input of MATErial commands.

The global SCALE option may be used when developing modules that require careful scaling of the equations to avoid severe numerical solution difficulties. At this time none of the elements included in the release have scaling. The scaling options are for COORdinates for use in scaling the nodal dimensions of an element; TIME for scaling the transient terms in an element; MASS for scaling the mass or density in an element. These commands have the form

```
GLOBal
  SCALe <COOR,TIME,MASS> value
```

The SCALe command may be repeated as many times as needed and input is terminated by a blank record. It is also possible to scale the degrees of freedom for each node using the command

```
GLOBal
  SCALe DISP (v(i),i=1,ndf)
```

Finally, internal values in an element may be scaled by using the command

```
GLOBal
  SCALe ELEMent j (v(i),i=1,j)
```

Users may also scale values by input of parameters within a material set. Recent interest in multiphysics problems involving formulations of differing types have shown some need for scaling to help in convergence.

```
idis
   node1,ngen1,(d(i,node1),i=1,ndf)
   node2,ngen2,(d(i,node2),i=1,ndf)
   <etc.,terminate with blank record>
```

The IDISplacement command is used to specify the values for imaginary parts of nodal boundary displacements. For each node to be specified a record is entered with the following information:

node	– Number of node to be specified
ngen	– Increment to next node, if generation is used, otherwise 0.
d(1,node)	– Value of displacement for 1-dof
d(2,node)	– Value of displacement for 2-dof
	etc., to *ndf* directions

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown at top):

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each displacement will be a linear interpolation between the *node1* and *node2* values for each degree-of-freedom.

While it is possible to specify both the force and the displacement applied to a node, only one can be active during a solution step. The determination of the active value is determined from the boundary restraint condition value. If the boundary restraint value is zero and you use one of the force-commands a force value is imposed, whereas, if the boundary restraint value is non-zero and you use one of the displacement-commands a displacement value is imposed. (See BOUNdary, CBOUndary, or EBOUndary pages for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value. It is not possible, to specify a displacement by using the combination of a force-command with a non-zero boundary restraint value, as it was in the last releases of *FEAP*. For further information see the CDISplacement page.

```
ifor
   node1,ngen1,(f(i,node1),i=1,ndf)
   node2,ngen2,(f(i,node2),i=1,ndf)
   <etc.,terminate with blank record>
```

The IFORce command is used to specify the imaginary part of values for nodal boundary forces. For each node to be specified a record is entered with the following information:

node	– Number of node to be specified
ngen	– Increment to next node, if generation is used, otherwise 0.
f(1,node)	– Value of force for 1-dof
f(2,node)	– Value of force for 2-dof
	etc., to *ndf* directions

When generation is performed, the node number sequence for *node1-node2* sequence shown at top will be:

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each force will be a linear interpolation between the *node1* and *node2* values for each degree-of-freedom.

While it is possible to specify both the force and the displacement applied to a node, only one can be active during a solution step. The determination of the active value is determined from the boundary restraint condition value. If the boundary restraint value is zero and you use one of the force-commands a force value is imposed, whereas, if the boundary restraint value is non-zero and you use one of the displacement-commands a displacement value is imposed. (See BOUNdary, CBOUndary, or EBOUndary pages for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value.

Example: IFORce

A concentrated force is to be applied to node 10. The force at node 10 has values of (100.0, i 50) in the horizontal direction and (0, i 0) in the vertical direction. These force components may be specified using the command set:

```
FORCe
  10  0  100.0  0.0
     ! Blank termination record
IFORce
  10  0   50    0.0
     ! Blank termination record
```

```
incl,filename
```

The INCLude command may be used to access data contained in a file called *filename*. This permits the data to be separated into groups which may be combined to form the problem data. Thus, if all the coordinate numerical data is in a file called COOR.DAT it may be combined into the mesh by using the command sequence:

```
COORdinates
  INCLude,COOR.DAT
         !blank terminator
```

This is particularly useful when data is generated by another program.

Another use is for cases in which multiple executions are to be performed using a different value for some parameter. Placing the problem data in a file named Example.prb (without the definition for the parameter) and using the sequence:

```
PARAmeter
  n=2
         !blank terminator
INCLude,Example.prb
         !blank terminator
PARAmeter
  n=4
         !blank terminator
INCLude,Example.prb
         !blank terminator
```

permits two executions for different values of the parameter n.

```
lbou,[set,add]
me,dof
```

A boundary restraint condition on degree of freedom `dof` of *element equations* may be set for element `me` using the mesh command set

```
LBOUndary
  me dof
     ....
    ! Blank record
```

Multiple records may be included in each set. The set is terminated by a blank record. The value of the degree of freedom must be specified between `1` and `ndl`, where `ndl` is the number of element equations in the element material set. The number of element equations for material set `ma` is specified by the `ELEMent` statement, *i.e.*,

```
MATErial ma
   ....
   ELEMent,,ndl
```

```
lfor[set,add]
   line (v(i),i=1,ndm)
   end1 (x(i),i=1,ndm),(f(j),j=1,ndf)
   end2 (x(i),i=1,ndm),(f(j),j=1,ndf)
```

The specified discrete force conditions may be set using the reference coordinates for a *line segment*. The input values are saved in files and searched after the entire mesh is specified. After use files are deleted. The data is order dependent with data defined by `FORC`e processed first, `EFOR`ce processed second, `CFOR`ce data processed third and `LFOR`ce data processed last. The value defined last is used for any analysis.

The `LFOR`ce command may be used with two options. Using the `CFOR,SET` option replaces all previously defined forces at any node by the pattern specified. This is the default mode. Using the `LFOR,ADD` option accumulates the specified forces with previously defined values.

For a *line*, the data to be supplied during the definition of the mesh consists of:

```
LFORce
  LINE v_1 v_2 v_3 ! ndm components
  END1 (x1_i, i=1,ndm), (f1_i, i=1,ndf)
  END2 (x2_i, i=1,ndm), (f2_i, i=1,ndf)
```

where

node	– Defines inputs to be for a *node*.
x(1)	– end point value of segment coordinates
...	– repeat components to
x(ndm)	– number of mesh spatial dimension.
f(1)	– force on 1-dof
f(2)	– force on 2-dof
...	
f(ndf)	

Only straight line segments may be considered. Nodes along the line segement are located and nodal forces are linearly interpolated along the segment. It may be necessary to modify end values to account for proper termination conditions.

```
load,<prop,pnum>
load end
```

This command is used to define the start and end of a load group. A load group is a set of nodal force and/or displacement values applied to the mesh. For nodal forces may be defined by any combination of the commands `FORCe`, `EFORce`, `CFORce` or `CSURface`. For displacements the nodal forces may be defined by any combination of the commands `DISPl`, `EDISpl`, `CDISpl` or `CSURface`.

A load group is defined by the set of commands

```
LOAD PROPld PNUM
   .... Force and or displacement inputs
LOAD END
```

In addition to the definition of the load group it is necessary to define the proportional load function (see the command `PROP` in the Solution Command section of the manual).

Specifying a load group by

```
LOAD
   .... Force and or displacement inputs
LOAD END
```

results in the group being multiplied by the sum of all defined proportional load functions. by `value`.

Example:

The set of commands

```
LOAD PROP 1
  FORCe
    10 0  0.0 -10.0

LOAD END
LOAD PROP 2
  FORCe
```

```
    10 0  2.0   5.0

LOAD END
```

results in the force set

$$\mathbf{F}(t) = \begin{Bmatrix} 0.0 \\ -10.0 \end{Bmatrix} p_1(t) + \begin{Bmatrix} 2.0 \\ 50.0 \end{Bmatrix} p_2(t)$$

being appied to node 10. The functions $p_1(t)$ and $p_2(t)$ are defined by the solution commands `PROP`, for example

```
BATCH
  PROP,,1
END
 2  2
0,1   1000,1
```

defines $p_1(t) \equiv 1.0$ for $0 \leq t \leq 1000$ and

```
BATCH
  PROP,,2
END
 0
```

defines $p_2(t) = t$.

```
loop ni
```

The `LOOP` command *must* be used in conjunction with a matching `NEXT` command.

A `LOOP-NEXT` pair is used to repeat the execution of a set of data input commands `ni` times. The `LOOP` appears first, followed by one or more commands then a `NEXT` command. The loop-next commands may be nested to a depth of 8. That is,

```
LOOP n1
  LOOP n2
    LOOP n3
       etc. to 8-levels
    NEXT
  NEXT
NEXT
```

is permitted.

This feature is particularly useful when parts of a mesh are repetitive. Translations, reflections, stretching and rotations may be accomplished by specifying a `TRANsform` command within the `LOOP-NEXT` pair.

MANUal FEAP MESH COMMAND MANUAL

```
manu,level
```

The `MANU`al command will set the *level* of help commands shown when the command `HELP` is given in an interactive solution mode. The levels are: 0 = basic; 1 = intermediate; 2 = advanced; 3 = expert. The default level is 0.

```
mass
   node1,ngen1,(m(i,node1),i=1,ndf)
   node2,ngen2,(m(i,node2),i=1,ndf)
   <etc.,terminate with blank record>
```

The `MASS` command is used to specify the values for nodal point masses. For each node on which non-zero values are to be specified a record is entered with the following information:

node	– Number of the node to be specified
ngen	– Increment to the next node, if generation is used, otherwise 0.
m(1,node)	– Value of the mass in 1-dof
m(2,node)	– Value of the mass in 2-dof
	etc., to *ndf* directions

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown at top):

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each mass will be a linear interpolation between the *node1* and *node2* values.

Example: MASS

A concentrated mass is to be specified at the end of a cantilever beam. The node number at the end is 101. The mass has both translational and rotational effects, necessitating the specification of the (principal) inertia tensor at the end. For a horizontal beam a rectangular block with side lengths a, b and c in the x_1, x_2 and x_3 directions is assumed. Mass density is r.

Parameters specify the values for the a, b, c and r as:

```
PARAmeters
  a = ...
  b = ...
  c = ...
  r = ...
  m = r*a*b*c
  i1 = m*(b*b + c*c)/12
  i2 = m*(c*c + a*a)/12
  i3 = m*(a*a + b*b)/12
```

The data for the concentrated mass effect is then given by the commands:

```
MASS
  101 0  m  m  m i1 i2 i3
```

```
mate,ma,<output label>
   type,iel,<id,(idf(i),i=1,ndf)>
   <parameters element type>
```

The MATErial set command is used to specify the parameters for each unique material set number *ma* in the analysis, as well as to specify the element type associated with the material set parameters.

The parameter *type* denotes the element formulation to be employed. *FEAP* includes a library of elements for thermo-mechanical analyses. The included types are:

ACOUstic	Continuum acoustic element (1, 2 or 3-D).
SOLId	Continuum solid mechanics element (1, 2 or 3-D).
THERmal	Continuum thermal element (1, 2 or 3-D).
FRAMe	2-Node frame element (2 or 3-D).
TRUSs	2-Node truss element (1, 2, or 3-D).
PLATe	Plate bending element (2-D).
SHELl	Shell element. (2 and 3-D)
MEMBrane	Membrane element. (3-D)
ROBIn	Robin boundary condition. (1, 2, or 3-D)
WINKler	Winkler boundary condition. (1, 2, or 3-D)
GAP	n-d Gap element.
PRESsure	Pressure load element (dead or follower). (2 or 3-D)

Users may also add their own elements and access by setting *type* to USER and the parameter *iel* to the number of the element module added (between 1 and 50).

The parameter *id* is the material identifier. Defined during element generation using ELEMent or BLOCk commands. If *id* is less than or equal to zero it defaults to the value of the *ma* parameter. Material sets with the same *id* number are associated to each element which designate this *id* number, thus, an element can be associated with more than one material set.

The *idf* parameters are used to assign active degrees of freedom. Default: idf(i) = i, i=1,ndf.

The *MATE*rial command may also be used to provide a material identification label for the *FEAP* output file.

Example: MATErial

The material properties for a 2-d thermo-mechanical problem with degree-of-freedoms u_1, u_2, T are given by:

```
MATE,1,Cam shaft material model: Aluminum mechanical
  SOLId,,1,1,2,3    ! properties for solid analysis
    ELAStic,,200.0d09,0.3
              ! terminate set 1
MATE,2,Cam shaft material model: Aluminum thermal
  THERmal,,1,3,0,0  ! properties for thermal analysis
    FOURier,,50
              ! terminate set 2
```

The *Cam shaft material model: Aluminum mechanical* will appear in the output file before the first material parameter values printed from the element routine. Note, that two material sets have the same material identifier, consequently the element connection list belonging to this identifier will be processed twice - once for the mechanical and once for the thermal. For the mechanical element the local dofs 1, 2, and 3 will map to global dofs 1, 2, and 3; for the thermal element local dof 1 will map to global dof 3. The mechanical element will not form residual or tangent terms for the 3-dof; however, it is used to extract the temperature used to calculate the thermal strains. This temperature degree of freedom must be designated for the material set using a `TEMP`erature command (or globally, using the `GLOB`al,`TEMP`erature command).

The specific parameters to be input are described in the user manual for the elements included with *FEAP*. For `USER` elements the data is set by the programmer of each module.

MBOUndary FEAP MESH COMMAND MANUAL

```
mbou,[set,add]
ma,dof
```

A boundary restraint condition on degree of freedom `dof` may be set for all nodes in an element that belongs to material set `ma` using the command set

```
MBOUndary
  ma dof
     ....
    ! Blank record
```

Multiple records may be included in each set. The set is terminated by a blank record.

```
next
```

The `NEXT` command *must* be used in conjunction with a `LOOP` command.

A `LOOP-NEXT` pair is used to repeat the execution of a set of commands. The `LOOP` appears first, followed by one or more commands then a `NEXT` command. The loop-next commands may be nested to a depth of 8. That is,

```
LOOP n1
  LOOP n2
    LOOP n3
       etc. to 8-levels
    NEXT
  NEXT
NEXT
```

is permitted.

See `LOOP` command for additional information.

NOPArse FEAP MESH COMMAND MANUAL

```
nopa
```

The `NOPA`rse command may be used to enforce no parsing of the input data. *FEAP* data may be input in either direct numerical form or in parameter or expression form. In the former case the data need not be parsed in order to compute the value of the data entry. When large amounts of data are to be processed the program can be forced to ignore parsing using the `NOPA`rse command and thus perform more efficiently. If subsequent data must be parsed, a `PARS`e command may be required to produce the correct results.

```
nopr
```

The use of the `NOPRint` command will discontinue placing information in the *FEAP* output file of most subsequent mesh data (material data printed in each element will always be output). The use of `PRINt` will cause the mesh information to again be reported in the output file. The default value is `PRINt` at the start of each problem execution.

```
para
   x = expression
```

The use of the PARAmeter command may be used to assign values to letter parameters. A letter parameter is defined immediately following the PARAmeter command (several may follow terminating with a blank record) according to the following:

```
x = expression
```

where x may be any of the single letters (*a*-*z*), any group of two letters (*aa*-*zz*), or any letter and a numeral (*a0*-*z9*) followed by the equal sign. The expression may be any set of numbers (floating point numbers should contain an E or a D exponent format so they will not be interpreted as integer constants!) or one or two letter constants together with any of the arithmetic operations +, -, *, /, or ^. The expression is processed left to right and can contain one set of parentheses to force groupings. Examples are:

```
a  = 3.
bb = 14/3.45
f  = a + 3.23/bb
c  = f + 1.03e-04*a/bb
d1 = (f + 1.03e-04)*a/bb
                ! blank terminator
```

In interactive mode of execution, the current set of parameter values may be output by entering *list* while in PARAmeter input mode. After listing, input of additional parameters may be continued. It is possible to use expressions containing the parameters while in any input mode.

An input file may contain multiple PARAmeter commands. The values for parameters may be reset as needed. If an expression requires more than one set of parentheses a parameter may be used to temporarily hold the value for one set of parentheses and then reset. For example,

```
a = cos( (2*n-1)*p/l )
```

is not legal because of the nested parentheses, but may be replaced by

```
a = 2*n-1
a = cos(a*p/l)
```

which is legal. Note the reuse and replacement of the a parameter. The list of functions permitted in expressions is defined in the user manual.

PARSe FEAP MESH COMMAND MANUAL

```
pars
```

The PARSe command may be used to enforce parsing of the input data. *FEAP* data may be input in either direct numerical form or in parameter or expression form. In the latter case the data must be parsed in order to compute the value of the data entry. When large amounts of data are to be processed the program can be forced to ignore parsing using the NOPArse command. If subsequent data must be parsed, a PARSe command may be required to produce the correct results.

```
peri,<hill,kirk>
  <mech,imech> <prop n_u>
  <ther,ither> <prop n_t>
  file file_name
```

For some problems in stress analysis of solids it is desirable to specify non-zero nodal displacements that result from applying a non-zero displacement gradient. For such situations the displacement of the node is computed from

$$\mathbf{u}_a = \mathbf{G}\,\mathbf{x}_a$$

where $\mathbf{u}_a$ and $\mathbf{x}_a$ are the displacement and coordinate of node a; and $\mathbf{G}$ is a *specified* gradient computed

$$\mathbf{G} = \begin{bmatrix} \dfrac{\partial u_1}{\partial x_1} & \dfrac{\partial u_1}{\partial x_2} & \dfrac{\partial u_1}{\partial x_3} \\ \dfrac{\partial u_2}{\partial x_1} & \dfrac{\partial u_2}{\partial x_2} & \dfrac{\partial u_2}{\partial x_3} \\ \dfrac{\partial u_3}{\partial x_1} & \dfrac{\partial u_3}{\partial x_2} & \dfrac{\partial u_3}{\partial x_3} \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix}$$

The displacement gradient is input using the commands

```
PERIodic <HILL,KIRK>
  MECHanical PROP n_u
  G_11    G_12    G_13
  G_21    G_22    G_23
  G_31    G_32    G_33
```

Another useful application is to impose *periodic* boundary conditions in which the constraint for two nodes is given as

$$u_i^+ = u_i^- + \sum_{j=1}^{d} G_i j (x_j^+ - x_j^-) \;\text{ with }\; x_j^+ = x_j^- \quad j \neq i$$

where d is the mesh dimension. Such periodic cases are useful in evaluating the behavior of micro-scale models under specified strain histories. Generally, the constraint can only be used for meshes in which the geometry is rectangular and the boundary nodes satisfy

the above constraint. The boundary constraints for the problem can then be specified by specifying fixed displacement conditions for the corner nodes and `ELINk` conditions for the parallel boundaries (see Sec. 11.2).

A similar behavior may be imposed for thermal problems where temperature T has the condition

$$T^+ = T^- + \sum_{j=1}^{d} G_j(x_j^+ - x_j^-) \quad \text{with} \quad x_j^+ = x_j^- \quad j \neq i$$

in which $G_j = \partial T/\partial x_j$ is a constant thermal gradientl.. The input is specified as

```
PERIodic <HILL,KIRK>
  THERmal PROP n_t
  G_1    G_2    G_3
```

It is also possible to solve the problem using an incremental form in which temperature, strain and deformation gradient are defined by

$$\nabla T = \frac{\partial T}{\partial \mathbf{x}} + \bar{G}$$

$$\boldsymbol{\varepsilon} = \left(\frac{\partial \mathbf{u}}{\partial \mathbf{x}}\right)^{(s)} + \bar{\mathbf{G}}^{(s)}$$

and

$$\mathbf{F} = \frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \bar{\mathbf{G}}$$

respectively. In the above the barred quantities are the imposed gradients.

The above are input as

```
PERIodic <HILL,KIRK>
  ITHErmal PROP n_t
  G_1    G_2    G_3
```

or

```
PERIodic <HILL,KIRK>
  IMEChanical PROP n_u
  G_11    G_12    G_13
  G_21    G_22    G_23
  G_31    G_32    G_33
```

In the latter, the distinction between small and finite deformation is based on the constitutive model used.

```
pola
   node,node1,node2,inc
   all
   <terminate with blank record>
```

The POLAr command may be used to convert any coordinates which have been specified in polar (or cylindrical) form, to cartesian coordinates. The conversion is performed using the following relations:

radius	= x(1,node)	– input value
theta	= x(2,node)	– input value in degrees
x(1,node)	= x_0 + radius × cos(theta)	
x(2,node)	= y_0 + radius × sin(theta)	
x(3,node)	= z_0 + x(3,node)	– 3-D only

The values for x_0, y_0, and z_0 are specified using the SHIFt command (default values are zero). A sequence of nodes may be converted by specifying non-zero values for *node1*, *node2*, and *inc*. The sequence generated will be:

node1, node1+inc, node1+2×inc, ... , node2

Several records may follow the POLAr command. Execution terminates with a blank record.

The option *all* perform the operation on all currently defined nodes.

PRINt FEAP MESH COMMAND MANUAL

```
prin
```

The use of the `PRINt` command will cause the description of most information produced during the mesh description to be placed in the *FEAP* output file. The use of `NOPRint` will discontinue the output of mesh information (except for data printed in elements). The default value is `PRINt`.

```
reac,<filename>
```

The use of the `REAC`tion command permits the retrieval of reaction data which was saved using the `REAC,FILE nnnn` command in the command language execution phase of the program. This option is useful when changing boundary conditions from displacement to force or when elements have been deleted. The command is specified as:

```
REACtion Rfile.reacnnnn
```

where `nnnn` is replaced by the appropriate character string beginning with `0000`.

READ FEAP MESH COMMAND MANUAL

```
read,<filename>
```

The use of the `READ` command permits the retrieval of mesh data which was processed by a `SAVE` command. For example, consider the following data in an input file.

```
save,Imatl
  mate,1
    user,1
      e,n,r,2,2,2
      1,0,0,0,0,0
            !end of material data
save,end
```

During the mesh input the data is processed normally, with the current values of the parameters e, n, r, used to describe the inputs. When the `SAVE,end` command is encountered, a file named *Imatl* is written to the current directory. The use of a

```
READ,Imatl
```

command will cause *FEAP* to reinput the commands which were saved, using the current values for e, n, r. These may be reset using a `PARA`meter command.

```
regi,nreg
```

The `REGIon` command sets the current region number to *nreg*. The default value is 0. Regions may be used to separate parts of the mesh for which use of a `TIE` command is to connect. Alternatively, regions may be used during execution to `ACTIvate` or `DEACtivate` parts of the mesh during execution.

```
 rese angl
rese disp
rese forc
rese <boun>
```

The `RESE`t command may be used to reinitialize mesh values to zero. The options are:

1. Specification of

   ```
   RESEt ANGLe
   ```

   sets all nodal angles to zero.

2. Specification of

   ```
   RESEt DISPlacement
   ```

   sets all nodal displacement components to zero.

3. Specification of

   ```
   RESEt FORCe
   ```

   sets all nodal force components to zero.

4. Specification of

   ```
   RESEt BOUNdary
   ```

   or simply

   ```
   RESEt
   ```

   sets all the boundary condition codes zero (i.e., to have no restraints imposed on any degree-of-freedom). Thus, all the degrees-of-freedom become unknowns for the problem.

These commands are useful when boundary conditions are to be changed from *displacement* to *force* states during execution. After the use o any `RESE`t command, new values for the deleted quantity may be specified using any of the available mesh input forms for the quantity.

```
rfor
   node1,ngen1,(f(i,node1),i=1,3)
   node2,ngen2,(f(i,node2),i=1,3)
   <etc.,terminate with blank record>
```

The RFORce command is used to specify the values for nodal boundary follower forces that have a purely radial or tangential component as shown in Fig. A.9.

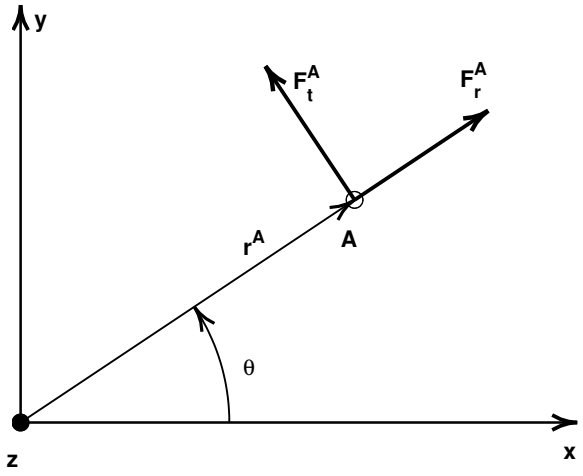


Figure A.9: RFORce: Radial and tangential follower forces at node-A.

For each node to be specified a record is entered with the following information:

node	– Number of node to be specified
ngen	– Increment to next node, if generation is used, otherwise 0.
f(1,node)	– Value of r-force
f(2,node)	– Value of θ-force
f(3,node)	– Value of z-force

For 2-dimensional problems the z-force component is not input. When generation is performed, the node number sequence for *node1-node2* sequence shown at top will be:

```
node1, node1+ngen1, node1+2*ngen1, .... , node2
```

The values for each force component will be a linear interpolation between the *node1* and *node2* values for each degree-of-freedom.

This form is intended for use with finite deformation problems in which forces are to be applied in a radial and/or tangential direction throughout the solution process. During the solution the radius is computed from:

$$
\begin{aligned}
x^A &= X^A + u^A \\
y^A &= Y^A + v^A \\
r^A &= [(x^A)^2 + (y^A)^2]^{1/2} \\
\theta^A &= tan^{-1}(y^A/x^A)
\end{aligned}
$$

While it is possible to specify both a force and a displacement at a node, only one can be active during any solution step. The determination of the active value is determined from the boundary restraint condition value. If the boundary restraint value is zero and a force value is imposed, whereas, if the boundary restraint value is non-zero and a displacement value is imposed. (See `BOUN`dary, `CBOU`ndary, or `EBOU`ndary pages for setting boundary restraint conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value using the `MESH` solution command.

Force conditions may also be specified using the `FORC`e, `EFOR`ce and `CFOR`ce commands.

`rigi,<nrbody>`

FEAP permits portions of a mesh to be declared as a rigid body. During the generation of the mesh it is necessary to designate which elements will belong to a rigid body. This is accomplished by inserting a record `RIGId,nrbody` before each group of elements which will belong to rigid body number `nrbody`.

Example:

```
RIGId nrbody
  ELEMents
    1,.....
                ! Blank terminator
```

To specify an additional rigid body another `RIGId` command may be given. The command for a `nrbody` may be given more than once. The command may also be inserted before a `BLOCk` or `BLENd` command.

The `FLEX`ible command is used to designate element groups as deformable. By default all elements are flexible.

Note: It is also necessary to use the `RIGId` mesh manipulation command to activate the rigid bodies and to assign additional parameters. See also the `JOINt` command for methods to interconnect rigid bodies.

SAVE FEAP MESH COMMAND MANUAL

```
save,<filename>
save,end
```

The use of the `SAVE` command permits the saving of mesh data for future retrieval by a `READ` command. For example, consider the following data in an input file.

```
SAVE,Imatl
  MATE,1
    USER,1
      e,n,r,2,2,2
      1,0,0,0,0,0
                ! end of material data
save,end
```

During the mesh input the data is processed normally, with the current values of the parameters e, n, r, used to describe the inputs. When the `SAVE,end` command is encountered, a file named *Imatl* is written to the current directory. The use of a `READ,Imatl` command will cause *FEAP* to reinput the commands which were saved using the current values for the e, n, r.

```
sblo,nsblk
   surf
     nodes,r-inc,s-inc,t-inc,node1,[elmt1,mat,r-skip]
      1,x1,y1,z1,th1 (only ndm coordinates required)
      2,x2,y2,z2,th2
      (etc.,until all nodes records are input).
   Repeat for next surf until 'nsblk' patches are defined.
```

The SBLOck data input segment is used to generate a regular three dimensional mesh of nodes for a set of *nsblk* surface patches. Alternatively, nodes together with 8-node brick elements may be generated based upon the set of three dimensional surfaces.

Each patch of nodes/elements defined by SBLOck is developed from a master surface element which is defined by an isoparametric 4-9 node mapping function in terms of the natural coordinates r and s. The node numbers on the master element of each patch defined by SBLOck are specified according to Figure A.2 in the BLOCk manual page. The four corner nodes of the master element must be specified, the mid-point and central node are optional. The three-dimensional mesh of nodes is constructed by erecting normals to the *surface* patch, each specified by a thickness, *th1*, at each surface i-node. The normals between patches are averaged for all patch nodes with the same coordinates to produce a continuous three dimensional mesh.

The spacing between the r-increments and s-increments may be varied by a proper specification of the mid-side and central nodes. Thus, it is possible to concentrate nodes and elements into one corner of the patch generated by SBLOck. The mid-nodes must lie within the central half of the r-direction or s-direction to keep the isoparametric mapping single valued for all (r,s) points. The thickness nodes are generated for a t-increment.

Patches may be interconnected, in a restricted manner, by using the *r-skip* parameter judiciously. In addition, the TIE command may be used to connect any nodes which have the same coordinates.

The data parameters are defined as:

nodes – Number of master nodes needed to define the patch.

r-inc – Number of nodal increments to be generated along r-direction of the patch.

s-inc – Number of nodal increments to be generated along s-direction of the patch.

t-inc – Number of nodal increments to be generated along t-direction of the patch (thickness).

node1 – Number to be assigned to first generated node in patch (default = 1). First node is located at same location as master node 1. Last generated node [i.e., node1 - 1 + (r-inc + 1) * (s-inc + 1)] is located at same location as master node 3.

elmt1 – Number to be assigned to first element generated in patch; if zero no elements are generated (default = 0)

matl – Material number to be assigned to all generated elements in patch (default = 1)

r-skip – Number of nodes to skip between end of an r-line and start of next r-line (may be used to interconnect blocks side-by-side) (default = 1)

```
shif
   x0,y0,z0
```

The SHIFt command is used to specify the values for the origin of polar and spherical coordinate transformations (used by commands POLAr, SPHErical, or BLOCk). The input of *x0*, *y0*, and, for three dimensional problems, *z0* are in cartesian values based on the reference mesh coordinate distances.

SIDE FEAP MESH COMMAND MANUAL

```
side
   type1,(is(i,side1),i=1,nn)
   type2,(is(i,side2),i=1,nn)
   <etc.,terminate with blank record>
```

Currently, *FEAP* uses the `SIDE` command to generate patches of a mesh using the *blending function* option and to determine contact surfaces. Blending functions are briefly discussed in the Zienkiewicz, Taylor & Zhu: The Finite Element Method, Its Basis & Fundamentals, 6th ed, pp 169 (or 5th ed, volume 1, pp 226 or 4th ed, volume 1, pp 181). Each super node is defined by an input of the following information:

It is necessary to define only those edges which are not straight or which have interpolations which generate non-equal spacing on a straight edge. `SIDE` commands *should not* be placed inside `LOOP-NEXT` pair commands, that is

```
LOOP,n
  SNODe
    ....
  SIDEs
    .......
  BLEND
    ....
NEXT
```

requires renumbering of the `BLENd` block nodes, where as

```
SNODe
  ...
SIDEs
  ...
LOOP,n
  BLEND
    .......
NEXT
```

permits the blend to use the same super node number for each generated block.

There are four options for generating the side description as indicated in the following table:

type	Type of interpolation
cart	- Lagrange interpolation in cartesian coordinates.
pola	- Lagrange interpolation in polar coordinates.
segm	- Straight multi-segment interpolation.
elli	- Lagrange interpolation in elliptical coordinates.

For Lagrange interpolation in cartesian coordinates the list of values defining the connected super nodes are given according to the following:

is	Type of interpolation
1	- End 1 super-node number.
2	- End 2 super-node number.
3	- Intermediate node nearest End 1.
$\cdots$	- etc. for remaining internal nodes.

For Lagrange interpolation in polar or elliptical coordinates the list of values is input as above, followed by the super-node number defining the location of the origin for the polar radius.

For straight multi-segment interpolations the inputs are given as:

is	Type of interpolation
1	- End 1 super-node number
2	- Number of equal increments to next node
3	- Intermediate node nearest End 1
4	- Number of equal increments to next node
5	- Next intermediate node
$\cdots$	- etc. for remaining internal nodes
nn	- End 2 super-node number

In addition to the side definitions it is necessary to define the super-node locations using the mesh command `SNOD`e. Finally, the mesh command `BLEN`d must be specified for each mesh patch to be created.

```
sloa
   iel,ns,nv,nl
   (ixl(i),i=1,ns),(p(i),i=1,nv)
   <etc.,terminate with blank record>
```

The `SLOAd` command is used to specify the values for surface loading quantities. Only traction quantities are considered (e.g., no surface displacement distributions may be specified by 'sloa'). The nodal values for the loads are determined by each element subprogram (i.e., in 'elmt**' with the isw = 7). Data is specified as follows:

iel	– Element subprogram which generates surface loads (only one routine may be given for a problem).
ns	– Number of nodes on surface of element.
nv	– Number of parameters defining distributed loading.
nl	– Loading type (generally only one type is currently included in elements and 'nl' is ignored - default may be 0).
ixl(i)	– List of nodes on element surface.
p(i)	– List of parameters defining loading.

No generation is permitted in the current implementation. A maximum of 8 items can appear on each record. If more than 8 items are required continue on the next record.

Before attempting to use this option users should see also the `CSURface` command for specifying distributed loads which do not change with deformation. Also, the element type `PRESsure` should be considered for cases where the pressure loading remains normal to deformed configuration. If these are not adequate for a users needs it is then necessary to write a new element which includes an option under `ISW = 7` to compute the loading.

SNODes FEAP MESH COMMAND MANUAL

```
snod
   snode1,(x(i,snode1),i=1,ndm)
   snode2,(x(i,snode2),i=1,ndm)
   <etc.,terminate with blank record>
```

The SNODe command is used to specify the values for nodal coordinates of *super nodes*. Currently, *FEAP* uses super nodes to generate patches of a mesh using the *blending function* option and to determine contact surfaces. Blending functions are briefly discussed in the Zienkiewicz, Taylor & Zhu: The Finite Element Method, Its Basis & Fundamentals, 6th ed, pp 169 (or 5th ed, volume 1, pp 226 or 4th ed, volume 1, pp 181). Each super node is defined by an input with the following information:

snode	- Number of super node to be specified.
x(1,snode)	- Value of coordinate in 1-direction.
x(2,snode)	- Value of coordinate in 2-direction.
	etc., to 'ndm' directions.

Super nodes must be numbered from 1 to the number needed to describe the *sides* and *blend patches*. The position of each super node is specified in cartesian coordinate components. No generation is performed for missing node numbers. SNODe commands *should not* be placed inside LOOP-NEXT pair commands, that is

```
LOOP,n
  SNODe
    ....
  SIDEs
    .......
  BLEND
    ....
NEXT
```

requires renumbering of the BLENd block nodes, whereas

```
SNODe
  ...
SIDEs
```

```
  ...
LOOP,n
  BLEND
    .......
NEXT
```

permits the blend to use the same super node number for each generated block. Location of all super nodes may be graphically displayed using the `PLOT,SNOD`e command.

In addition to the supernodes it may be necessary to define the sides of blend patches using the mesh command `SIDE`. Also, the mesh command `BLEN`d must be given for each mesh patch to be created.

```
sphe
   node1,node2,inc
   <terminate with blank record>
```

The SPHErical command may be used to convert any coordinates which have been specified in spherical form, to cartesian coordinates. The conversion is performed using the following relations:

radius	= x(1,node)	– input value
theta	= x(2,node)	– input value in degrees
phi	= x(3,node)	– input value in degrees
x(1,node)	= x_0 + radius × cos(theta) × sin(phi)	
x(2,node)	= y_0 + radius × sin(theta) × sin(phi)	
x(3,node)	= z_0 + radius × cos(phi)	

The values for x_0, y_0 and z_0 are specified by the SHIFt command (default values are zero). A sequence of nodes may be converted by specifying non-zero values for *node1*, *node2*, and *inc*. The sequence generated will be:

node1, node1+inc, node1+2×inc, ... , node2

Several records may follow the SPHErical command. Execution terminates with a blank record.

```
stif
   node1,ngen1,(k(i,node2),i=1,ndf)
   node2,ngen2,(k(i,node2),i=1,ndf)
   <etc.,terminate with blank record>
```

The `STIF`fness command is used to specify the values for linear nodal stiffness (i.e., spring) to earth. For each node for which non-zero values are to be specified a record is entered with the following information:

node	– Number of the node to be specified
ngen	– Increment to the next node, if generation is used, otherwise 0.
k(1,node)	– Value of the stiffness in 1-dof
k(2,node)	– Value of the stiffness in 2-dof
	etc., to *ndf* directions

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown at top):

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each stiffness will be a linear interpolation between the *node1* and *node2* values.

Example: STIFfness

A spring (point stiffness) relative to a fixed condition (earth) may be specified by the commands

```
STIFfness
  15 0 500 1250
```

This inserts a diagonal stiffness of 500 in the horizontal (1^{st} dof) direction and 1250 in the vertical direction at node 15. Note that application of a boundary restraint (using `BOUN`, etc.) for either degree of freedom will result in the stiffness being ignored.

```
stru pair
 stru two
 stru
   elem1,u_1,u_2,u_3,<v_1,v_2,v_3>
   elem2,u_1,u_2,u_3,<v_1,v_2,v_3>
   <etc.,terminate with blank record>
```

The `STRUcture` command is used to specify the values for structure vectors associated with specific elements. The structure vectors are currently restricted to use with the `fiber` material models. Options for single fibers in each element and pairs of fibers are controlled by the second parameter on the `STRUcture` command record. Thus if there are two fibers in each element the input is given as

```
STRUcture PAIR
  e_a  ua _1 ua _2 ua _3   va _1 va _2 va _3
  e_b  ub _1 ub _2 ub _3   vb _1 vb _2 vb _3
     etc.
    ! Blank termination record
```

If an element has only one fiber then the input is given as

```
STRUcture
  e_a  ua _1 ua _2 ua _3
  e_b  ub _1 ub _2 ub _3
     etc.
    ! Blank termination record
```

No generation is provided and ALL elements with fibers must be specified individually.

TBOUndary FEAP MESH COMMAND MANUAL

`tbou`

Boundary restraint conditions may be set for all the unknowns in the problem using the single command

`TBOUndary`

This is sometimes referred to as the Taylor boundary conditions for representative volume elements.

```
temp
   node1,ngen1,t(node1)
   node2,ngen2,t(node2)
   <etc.,terminate with blank record>
```

The `TEMP`orary command is used to specify the values for nodal quantities that may be required in an analysis. For example if the values of temperature are known they may be stored in the `TEMP` array and used in a stress analysis. For each node to be specified a record is entered with the following information:

node	– Number of the node to be specified.
ngen	– Increment to the next node, if generation is used, otherwise 0.
t(node)	– Value of quantity at node.

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown at top):

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each generated value will be a linear interpolation between *node1* and *node2*.

The data is provided as

```
TEMP = name
  node1 gen1 val1
  node2 gen2 val2
    ...
  ! Terminate with a blank line
```

The value for `name` us used to describe the quantity on the output and by default is `Temperature`.

TITLe FEAP MESH COMMAND MANUAL

```
titl,<on>
titl,off
```

The `TITL`e,*off* command is used to suppress the print of headers on output pages produced by *FEAP*. It may be toggled on by entering the command with no parameter. This is provided to produce outputs devoid of header information every few lines, thus, the outputs are more readily usable by other programs or data conversions.

`tran,<inc>`
T_{11} , T_{12} , T_{13}
T_{21} , T_{22} , T_{23}
T_{31} , T_{32} , T_{33}
x_0 , y_0 , z_0

The `TRAN`sformation command defines a coordinate transformation to be applied to input values. After specification of the command the input nodal coordinates $\boldsymbol{x}_{input}$ are transformed to global nodal coordinates, $\boldsymbol{x}$ using

$$\mathbf{x} = \mathbf{T}\,\mathbf{x}_{input} + \mathbf{x}_0 \ .$$

Thus the $\boldsymbol{x}$ correspond to the nodal values after applying the transformation and become the values used for the analysis.

Two options exist to define the transformation: (a) a direct specification of the translation and rotation transformation arrays; (b) an incremental specification of the arrays.

Example: Direct specification TRANsformation

A rectangular block of nodes and elements of size 20×20 units is to be generated in two dimensions in a rotated coordinate frame (30^o relative to x_1 axis). The commands may be given as

```
TRANsform
  cosd(30)  sind(30)  0
 -sind(30)  cosd(30)  0
     0          0      1
     0          0      0

BLOCk
  CARTesian n1 n2
    1  0  0
    2 20  0
    3 20 10
    4  0 10
```

After the generation it is best to enter an identity transformation to prevent any spurious later effects. That is enter the set

```
TRANsform
     1  0  0
     0  1  0
     0  0  1
     0  0  0
```

before ending the mesh generations.

Example: Incremental specification TRANsformation

Another block of elements may be input in which the transformation is given as:

$$\begin{aligned} \mathbf{T}_{new} &= \mathbf{T}_{inc}\,\mathbf{T}_{old} \\ \mathbf{x}_{0,new} &= \mathbf{T}_{inc}\,\mathbf{x}_{0,old} + \mathbf{x}_{inc}\ . \end{aligned}$$

In this case the new coordinates are given as:

$$\mathbf{x} = \mathbf{T}_{new}\,\mathbf{x}_{input} + \mathbf{x}_{0,new}\ .$$

Thus specification of a second block of nodes as:

```
TRANsform,INCrement
  cosd(30)  sind(30)  0
 -sind(30)  cosd(30)  0
     0         0      1
     0         0      0

BLOCk
  CARTesian n1 n2
    1  0  0
    2 20  0
    3 20 10
    4  0 10
```

after the first block given above will create a block rotated by 60^o. This option may be used very conveniently with `LOOP-NEXT` commands to replicate a block of nodes and elements, each rotated and/or translated by an equal incremental amount.

```
tria
   node1,ngen1,(t_1(node1),i=1,3),(t_2(node1),i=1,3),(t_3(node1),i=1,3)

   node2,ngen2,(t_1(node2),i=1,3),(t_2(node2),i=1,3),(t_3(node2),i=1,3)

   <etc,,terminate with blank record>
```

For three-dimensional problems, the `TRIAd` command is used to specify an orthonormal vector triad for nodal boundary conditions oriented as shown in Fig. A.8. For each node `I` to be specified a record is entered with the following information:

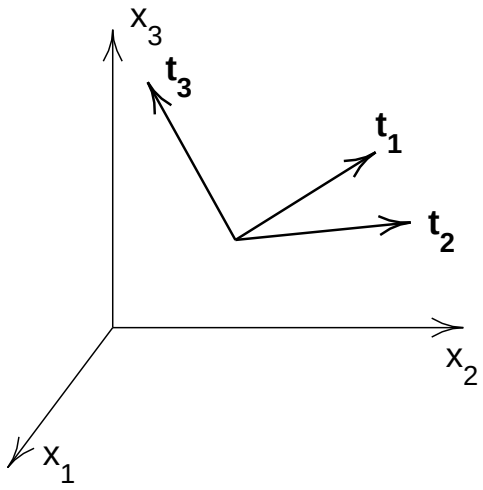


Figure A.10: TRIAd: Triad of orthonormal vectors.

node	– the number of the I-node to be specified
ngen	– the increment to the next node, if generation is used, otherwise 0.
t_1x(node)	– value of compontent x for vector 1
t_1y(node)	– value of compontent y for vector 1
t_1z(node)	– value of compontent z for vector 1
t_2x(node)	– value of compontent x for vector 2
t_2y(node)	– value of compontent y for vector 2
t_2z(node)	– value of compontent z for vector 2
t_3x(node)	– value of compontent x for vector 3
t_3y(node)	– value of compontent y for vector 3
t_3z(node)	– value of compontent z for vector 3

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown above):

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each triad generated will be a linear interpolation between *node1* and *node2*.

The vector unknowns for the transformed nodes will be given in terms of the components expressed in terms of the triad. Accordingly,

$$\mathbf{u} = \sum_{j=1}^{3} \widehat{u}_j \, \mathbf{t}_j$$

Boundary conditions and nodal forces & displacements will also be interpreted in terms of the triads.

The degrees-of-freedom associated with the sloping boundary may differ from element to element as described in the element manuals. The default will be the first three degrees-of-freedom (3-D problems) which are affected by the sloping condition. Only displacement values will be assumed to be given in the rotated frame. To activate the rotated boundary condition use the `BOUN`dary, `DISP`lacement, etc. command.

```
trib
   type,n-inc,[node1,elmt1,mat]
   1,x1,y1,z1 (only ndm coordinates required)
   2,x2,y2,z2
   etc.,until all nodes records are input.
```

The `TRIBlock` data input segment is used to generate a triangular two dimensional patch of nodes and 3-node triangular elements.

The patch of nodes/elements defined by `TRIBlock` is developed from a master element which is defined by an isoparametric 3-6 node mapping function in terms of the natural coordinates *L1*, *L2* and *L3*. The first three node numbers on the master element of each patch defined by `TRIBlock` are the vertex nodes of the master patch. The additional nodes are 4 - midside of edge 1-2; 5 - midside of edge 2-3; and 6 - midside of edge 3-1. The vertex nodes must be specified. Midside nodes are optional.

The node spacing may be varied by a proper specification of the mid-side nodes. Thus, it is possible to concentrate nodes/elements into one corner of the patch generated by `TRIBlock`. The mid-nodes must lie within the central-half of each edge to keep the isoparametric mapping single valued for all points.

Patches may be interconnected using the `TIE` command will merge any nodes which have the same coordinates.

The data parameters are defined as:

type – Type of block (cartesian, polar, spherical)
n-inc – Number of nodal increments to be generated along each side of the patch.
node1 – Number to be assigned to first generated node in patch (default = 1).
elmt1 – Number to be assigned to first element generated in patch; if zero no elements are generated (default = 0).
matl – Material number assigned to all generated elements in patch (default = 1)

Appendix B

Mesh manipulation manual pages

After the mesh is initially defined *FEAP* has options which may be used define additional features. These features include the ability to merge parts of the mesh generated as blocks and blends as well as linking the degrees of freedom from one node to those of another. It is also possible to define interconnections between rigid bodies and activate the rigid body options. The following pages summarize the commands which are available to manipulate the mesh data. Currently these include:

1. Command to merge mesh parts:

   `TIE`

2. Commands to link degree of freedoms from one node to another:

   `LINK, CLINk,  ELINk`

3. Command to define partitions of problem:

   `PARTition`

4. Command to set order of transients for degree of freedoms:

   `ORDEr`

5. Commands to define rigid body conditions:

   `RIGId, JOINts, RBOUndary, RDISplacement, RLOAd`

6. Command to define master-slave behavior for small deformation analysis:

   `MASTer`

Users may also add their own mesh manipulation commands by writing a subprogram `UMANIn`. See the *FEAP* Programmer Manual for more information.

```
clin
   mate,m1,x2,(idl(i),i=1,ndf)
   regi,m1,x2,(idl(i),i=1,ndf)
   node,(idl(i),i=1,ndf)
   <terminate with a blank record>
gap tol_value
```

A mesh may be generated in *FEAP* in which it is desired to let the some or all of the degree-of-freedom values for nodes located at the same point to share the same displacement unknown.

To use the `CLINk` option the complete mesh must first be defined. The data statements are placed after the mesh `END` command and before the `BATCh`, or `INTEractive` command for defining a solution algorithm. The data for a `CLINk` consists of the the type of search (*i.e.*, `MATErial`, `REGIon` or `NODE`) and the degree-of-freedom list will cause the program to search for all conditions that are to be connected together. A link will be performed for all zero values in the degree-of-freedom list.

The input data is interpreted as follows:

Example: The command

```
CLINk
  MATErial m1  m2 1 1 0
```

will link the third degree of freedom for all nodes between material set `m1` and `m2` which have identical coordinates to within the *gap tolerance*.

A tolerance on coordinate differences (i.e., gap) is used to make the search. By default the tolerance is 0.0001 (10^{-4}); however, when coordinates have values for which this in not appropriate the search gap may be reset by inserting the data `GAP` command anywhere after the `CLINk` command. For example:

```
CLIN
  GAP  tol_value
  REGIon r1 r1  dof list
```

will set the search gap for nodes to `tol_value` and search on regions.

ELINk FEAP MESH MANIPULATION COMMAND MANUAL

```
elin
   dir,x1,x2,(idl(i),i=1,ndf)
   <terminate with a blank record>
gap value
```

A mesh may be generated in *FEAP* in which it is desired to let the some or all of the degree-of-freedom values for nodes located at two constant values for a coordinate direction share the same displacement unknown. For example, in repeating structures the value of the dependent variable along two equally spaced edges should be the same. In a finite element model it is necessary to specify the repeating condition by linking the degree-of-freedoms at theses nodes to the same unknown in the equations. The `ELINk` command may be used for this purpose.

To use the `ELINk` option the complete mesh must first be defined. After the `END` command for the mesh definition and before the `BATCh`, or `INTEractive` command for defining a solution algorithm, the use of a `ELINk` statement together with the direction, *dir*, the values of two coordinates, *x1* and *x2*, for the direction, and the link pattern for the degrees-of-freedom will cause the program to search for all conditions that are to be connected together. A connection is performed whenever the coordinates for the directions other than *dir* are the same.

The input data is interpreted as follows:

dir	– Coordinate direction for edge.
x1	– Coordinate value for direction *dir*.
x2	– Coordinate value for direction *dir*.
idl(1)	– Linking condition, 0 = link, non-zero do not link dof 1.
idl(2)	– Linking condition, 0 = link, non-zero do not link dof 2.
	etc. for *ndf* degree of freedoms.

Example: The command

```
ELIN
  1  0.0  10.0  0  1
```

will link the first degree of freedom for all nodes which have *x1* equal to 0.0 or 10.0 and all the other xi are the same.

A tolerance on coordinate differences (i.e., gap) is used to make the search. By default the tolerance is 0.0001 (10^{-4}); however, when coordinates have values for which this in not appropriate the search gap may be reset by inserting the data `GAP` command anywhere after the `ELINk` command. For example:

```
ELIN
  ...
  GAP  value
  ...
```

would read data for which the default value is used and then after the `GAP` command the gap would be reset to *value*.

```
join
   type,body1,body2,x1,y1,z1,<x2,y2,z2>
   <terminate with a blank record>
```

Rigid bodies may be interconnected by joint restraints. The specification of joints is initiated using a `JOINt` mesh manipulation command. Immediately following the `JOINt` command the list of joint types and their association to rigid bodies must be specified. All joint types involve two rigid bodies in which *body1* is the number of one rigid body and *body2* the number of the other rigid body. Currently, the joint types are

a. Ball and Socket: A ball and socket joint constrains two rigid bodies to have the same position at some specified location. Only translational motion is restrained, thus permitting the two bodies to rotate freely relative to the restraint point. A ball and socket joint is specified as:

```
BALL,body1,body2,x1,y1,z1
```

where *(x1, y1, z1)* are the reference coordinates for the constraint point.

b. Revolute: A revolute joint constrains the rotation to be about some specified axis described by two points in the reference configuration of the mesh. A revolute joint is specified as:

```
REVOlute,body1,body2,x1,y1,z1,x2,y2,z2
```

where *(x1, y1, z1)* and *(x2, y2, z2)* are the reference coordinates of two points defining the direction of the axis about which rotations take place. A revolute joint is created by combining a *BALL and socket* type with a *REVOlute* type.

c. Slider: A slider joint permits two objects to translate relative to a specified axis while also permitting rotation about the axis. The axis may rotate in space during the solution. The slider joint is specified as:

```
SLIDer,body1,body2,x1,y1,z1,x2,y2,z2
```

where *(x1, y1, z1)* and *(x2, y2, z2)* are the reference coordinates of two points defining the axis on which sliding take place.

d. Translator: A translator joint permits two objects to translate relative to a specified axis without any relative rotation of the bodies about the axis. The axis may rotate in space during the solution. The translator joint is specified as:

```
TRANslator,body1,body2,x1,y1,z1,x2,y2,z2
```

where *(x1, y1, z1)* and *(x2, y2, z2)* are the reference coordinates of two points defining the axis on which sliding take place.

e. Plane: A plane joint constrains a rigid body to slide on a specified plane. The plane joint is specified as:

```
PLANe,body1,body2,x1,y1,z1,x2,y2,z2
```

where *(x1, y1, z1)* and *(x2, y2, z2)* are the reference coordinates of two points defining the normal to the plane on which sliding take place.

LINK FEAP MESH MANIPULATION COMMAND MANUAL

```
link
   node1,node2,inc1,inc2,(idl(i),i=1,ndf)
   <terminate with a blank record>
```

A mesh may be generated in *FEAP* in which it is desired to let the some or all of the degree of freedom values at more than one node share the same displacement unknown. For example, in repeating structures the value of the dependent variable will be the same at each repeating interval. In a finite element model it is necessary to specify the repeating condition by linking the degree of freedoms at theses nodes to the same unknown in the equations. The `LINK` command is used for this purpose.

To use the `LINK` option the complete mesh must first be defined. After the `END` command for the mesh definition and before the `BATCh` or `INTEractive` command for defining a solution algorithm, the use of a `LINK` statement together with the list of affected nodes and degree of freedoms will cause the program to search for all conditions that are to be connected together.

The input data is interpreted as follows:

node1	– Node on one body to be linked.
node2	– Node on other body to be linked.
inc1	– Increment to generate additional nodes for node1.
inc2	– Increment to generate additional nodes for node2.
idl(1)	– Linking condition, 0 = link, non-zero do not link dof 1.
idl(2)	– Linking condition, 0 = link, non-zero do not link dof 2.
	etc. for *ndf* degrees of freedom.

Generation is accomplished by giving a pair of records. A generation terminates whenever one of the sequences is reached. For example:

```
LINK
  5,105,3,5,1,0,1
  15,140,,,1,1,0
```

will generate the sequence of links

Node 1	Node 2	Link Codes
5	105	1 0 1
8	110	1 0 1
11	115	1 0 1
14	120	1 0 1
15	140	1 1 0

Termination of input occurs with a blank record.

Whenever it is desired to only connect *node1* to *node2*, *inc1* and *inc2* need not be specified (they may be blank or zero).

MANUal FEAP MESH MANIPULATION COMMAND MANUAL

```
manu,level
```

The `MANU`al command will set the *level* of help commands shown when the command `HELP` is given in an interactive solution mode. The levels are: 0 = basic; 1 = intermediate; 2 = advanced; 3 = expert. The default level is 0.

MASTer-slave FEAP MESH MANIPULATION COMMAND MANUAL

```
master
 slave (xm(i),i=1,ndf) (xs(i),i=1,ndf) (rlink(i),i=1,ndf)
 surface (xm(i),i=1,ndf) idir (rlink(i),i=1,ndf)
 gap value
```

The `MASTer` command is used for small deformation problems in which it is desired to express the response of the degrees-of-freedom (DOF) at a set of nodes (called *slave* nodes) in terms of the DOF at one node (called the *master* node). It is possible to keep some DOF at the slave nodes active using the `rlink` pattern. A non-zero value in the `rlink` set keeps the DOF of the slave active. Multiple slave nodes may be associated with a single master node by repeating a `slave` option with the same master coordinates `xm` and different slave coordinates `xs`.

Example: A 2-d problem with 2-DOF per node.

```
MASTer ! x_m  x_s  rlink
  SLAVe  5,5  0,3  1 0
  SLAVe  5,5  3,2  1 0
               ! blank terminator
```

has two slave nodes (located at 0,3 and 3,2) associated with one master node located at 5,5. The first DOF for the two slaves is to remain active and independent of the response at the master. The second DOF for the three nodes has the same solution value.

Alternatively, the `surface` option assigns as slaves all nodes which have the same `xs(idir)` values as the `xm(idir)` coordinate value.

Example: A 3-d problem with 3-DOF per node.

```
MASTer !   x_m    idir rlink
  SURFace  5,5,3.5  3  1 1 0
               ! blank terminator
```

will find all nodes which have an x_3 coordinate equal to `3.5` and assign them as slave nodes. Only the u_3 displacement will be slaved to the master node displacement. This will produce a surface which moves as a plane in the 3-direction. The slave node degrees

of freedom will be associated with the master node closest to the specified coordinates of the master point specified (e.g., $x_1 = 5,\ x_2 = 5,\ x_3 = 3.5$ in the example above).

```
orde
   ord_1,ord_2, ...  ord_ndf
```

Problems may be solved in *FEAP* where each degrees-of-freedom (DOF) is associated with an ordinary differential equation (ODE)of order-p. In the current implementation only ODE's of order zero (0), one (1), and two (2) may be considered. By default all the DOF will be associated with the highest order ODE associated with the specified `TRAN`sient solution command (e.g., the Newmark option will have the DOF associated with a second order ODE). To assign the DOF to different order ODE it is necessary to insert a `ORDE`r command between the mesh `END` command and the first problem solution command `BATC`h or `INTE`ractive.

The `ORDE`r command is followed by a record which denotes the order of the ODE for each DOF.

As an example consider the solution of a thermo-mechanical problem in which the global DOF are ordered as two displacements (u_1 and u_2) and the temperature (T). A transient solution is to be performed in which the displacements are associated with a quasi-static behavior (no inertia loads) and the temperature to a first order ODE. The data to make this assignment is given by:

```
ORDEr
  0  0  1
```

The specification of a `TRANsient,BACK` algorithm may then be used in the solution process. In the solid (and/or truss) elements the inertial effects will be ignored. Similarly, solution of a transient mechanical and thermal problem can be performed by using the `TRANsient,NEWMark` algorithm with the order command:

```
ORDEr
  2  2  1
```

PARTition FEAP MESH MANIPULATION COMMAND MANUAL

```
      part
         part_1 list of active/inactive dof's              part_2 list of
active/inactive dof's                ...                  part_n list of
active/inactive dof's
```

Problems may be solved in *FEAP* where all degrees-of-freedom (DOF) are treated together or where they are split into *partitions*. By default the DOF's are all in a single partition (called partition 1). To assign the DOF to different partitions it is necessary to insert a `PARTition` command between the mesh `END` command and the first problem solution command `BATCh` or `INTEractive`.

The `PART` command is followed by a set of records which denote the active dof's in the partition. An active dof is indicated by a non-zero number and an inactive one by a zero (0) or blank field. Admissible partition numbers range from one (1) to four (4).

As an example consider the solution of a thermo-mechanical problem in which the global DOF are ordered as two displacements (u_1 and u_2) and the temperature (T). A solution is to be performed in which the displacements are assigned to partition number 2 and the temperature to partition number 1. The data to make this assignment is given by:

```
PARTition
  0  0  1
  1  1  0
    ! Blank record to terminate input
```

Note that a DOF can belong to more one partition; thus, general staggering algorithms can be considered. For a case which also permits a fully monolithic solution (all dof's active) the data is given as

```
PARTition
  0  0  1
  1  1  0
  1  1  1
    ! Blank record to terminate input
```

where now the third partition includes all dof's.

During solution the partitions are activated by the `PARTition` solution command (see solution command section D).

```
rbou
   body,comp_1,comp_2, ...  ,comp_ndf
   < terminate with blank record >
```

Rigid bodies may have some of their degrees-of-freedom restrained by boundary condition codes. These may be specified for each rigid body. Users may specify resultants applied to each body using the `RLOAd` command which must appear in the data file after the `END` mesh command and before the first `BATCh` or `INTEractive` solution command. A fixed DOF has a non-zero restraint code and an active DOF has a zero restraint code.

Example:

```
RBOUnd
  2  1 1 1 0 0 0
```

specifies that rigid body 2 (assumed a 3-D problem) has all of its translation DOF fixed and can rotate freely about its center of mass.

Rigid bodies may be interconnected using joints (see `JOINt` mesh manipulation command). They may also be loaded and restrained at their center of mass (see `RLOAd` and `RDISplacement` mesh manipulation of commands).

`rigi,<nrbdof,npart,neqrb>`

FEAP permits portions of a mesh to be declared as a rigid body. The parts of a rigid body are associated with individual element sets defined by the `RIGId` mesh command during inputs. The material properties for each element are used to compute inertial properties for each rigid body. At least one of the materials must have a non-zero density or an error will result.

Each rigid body has a set number of equations. For two dimensions each body has three degrees-of-freedom (DOF) (2-translations, and 1-rotation); in three dimensions there are six DOF (3-translations and 3-rotations). The number may be changed using the `nrbdof` parameter.

Rigid bodies are associated with a partition during the solution process (default is partition 1). The associated partition may be changed by specifying a specific value for the `npart` parameter.

Different options are available to perform the rotational updates in three dimensions by specifying the `neqrb` option.

At present it is recommended to use the rigid body option to solve problems with only one partition and accept the default values for the number of rigid body DOF, partition, and equation update method.

The rigid body option is initiated using the mesh manipulation command `RIGId`, which must appear in the data file after the `END` mesh command and before the first `BATCh` or `INTEractive` solution comomand.

Rigid bodies may be interconnected using joints (see `JOINt` mesh manipulation command). They may also be loaded and restrained at their center of mass (see `RLOAd` and `RDISplacement` mesh manipulation of commands).

```
rloa
   body,comp_1,comp_2, ...  ,comp_ndf
```

Rigid bodies may be loaded by forces specified at nodes in an identical manner as for any deformable body. These forces will be transformed to a resultant and couple at the center of mass of each body. Alternatively, users may specify resultants applied to each body using the `RLOAd` command which must appear in the data file after the `END` mesh command and before the first `BATCh` or `INTEractive` solution command.

Rigid bodies may be interconnected using joints (see `JOINt` mesh manipulation command). They may also be loaded and restrained at their center of mass (see `RLOAd` and `RDISplacement` mesh manipulation of commands).

```
rpro
   body,comp_1,comp_2, ...  ,comp_ndf
   < terminate with blank record >
```

Rigid bodies may have some their loads or displacements specified by non- zero values that vary in time. The associated proportioinal load to each degree of freedom on the rigid body may be specified using the `RPROp` command which may be specified for each rigid body separately.. Users may specify resultants applied to each body using the `RPROp` command which must appear in the data file after the `END` mesh command and before the first `BATCh` or `INTEr`active solution command. A fixed DOF has a non-zero restraint code and an active DOF has a zero restraint code.

Example:

```
RPROp
  2  3 1 2 0 0 0
```

specifies that rigid body 2 (assumed a 3-D problem) has its translation DOF nonzero values associated to the proporional load number "3" for dof "1"; "1" for dof "2" and "2" for dof "3".

WARNING: Any loads or displacements on a degree of freedom that has a zero value above will be scaled by the sum of all proportional load factors.

To specify loads or displacements see `RLOAd` and `RDISplacement` mesh manipulation of commands.

SLINk FEAP MESH MANIPULATION COMMAND MANUAL

```
slin
   dir,x1,x2,(idl(i),i=1,ndf)
   <terminate with a blank record>
gap value
```

A mesh may be generated in *FEAP* in which it is desired to let all of the surface degree-of-freedom values for nodes located at a constant value for a coordinate direction share the same displacement unknown with a node or surface at a different value of the same coordinate direction.

To use the `SLINk` option the complete mesh must first be defined. After the `END` command for the mesh definition and before the `BATCh`, or `INTEractive` command for defining a solution algorithm, the use of a `SLINk` statement together with the direction, *dir*, the values of two coordinates, *x1* and *x2*, for the direction, and the link pattern for the degrees-of-freedom will cause the program to search for all conditions that are to be connected together. A connection is performed whenever the coordinates for the directions other than *dir* are the same.

The input data is interpreted as follows:

dir	– Coordinate direction for edge.
x1	– Coordinate value for direction *dir*.
x2	– Coordinate value for direction *dir*.
idl(1)	– Linking condition, 0 = link, non-zero do not link dof 1.
idl(2)	– Linking condition, 0 = link, non-zero do not link dof 2.
	etc. for *ndf* degree of freedoms.

Example: The command

```
SLIN
  1  0.0  10.0  0  1
```

will link the first degree of freedom for all nodes which have *x1* equal to 0.0 or 10.0.

A tolerance on coordinate differences (i.e., gap) is used to make the search. By default the tolerance is 0.0001 (10^{-4}); however, when coordinates have values for which this

in not appropriate the search gap may be reset by inserting the data `GAP` command anywhere after the `SLINk` command. For example:

```
SLIN
  ...
  GAP  value
  ...
```

would read data for which the default value is used and then after the `GAP` command the gap would be reset to *value*.

TIE FEAP MESH MANIPULATION COMMAND MANUAL

```
tie
tie gap gtol
tie line n1 n2
tie coor x1 x2 x3
tie edge dir x-dir
tie mate n1 n2
tie node n1 n2
tie regi n1 n2
```

A mesh may be generated by *FEAP* in which there is more than one node with the same coordinates (to within a tolerance). The `TIE` command may be used after the mesh `END` command to *merge* these nodes so that the same values of the solution will be produced at specified nodes which have the same initial coordinates. Current options include:

gap	– Set search gap fraction
line	– [Currently not documented]
coor	– Search all nodes connect all with same coordinate
edge	– Search all nodes with same coordinates on an edge
mate	– Search material identifiers *n1* and *n2* (*n1* can equal *n2*)
node	– Search node list between nodes *n1* and *n2*
regi	– Search regions *n1* and *n2* (*n1* can equal *n2*)

To use the `TIE` option the *complete mesh* must first be defined. After the `END` command for the mesh definition and before the first `BATCh` or `INTEractive` command for defining a solution algorithm, use of a `TIE` statement will cause the program to perform tie options as described below. Multiple tie commands may be given.

Use of the `TIE` command without additional parameters will search all nodes and join those which have coordinates with the same values (to within a small tolerance). Use of `TIE EDGE` *i,value* (with i between 1 and ndm) will tie nodes with common coordinates which are on the plane defined with an x_i coordinate equal to *value*. Similarly, the use of the *coordinate*, *material*, *node* or *region* parameter will result in searches based on these identifiers.

The coordinate option permits a search as

```
TIE COOR x1 x2 x3
```

and all nodes with these coordinates (`x3` need not be given for two dimensional problems) will be merged and have the same solution.

Use of the material limiter as

```
TIE MATE m1 m2
```

will limit the merge to only materials `m1` and `m2` (which may be the same material if desired).

Use of the region limiter as

```
TIE REGI r1 r2
```

will limit the merge to only regions `r1` and `r2` (which may be the same region if desired).

FEAP will set a default relative tolerance (`tol`) for the search based upon the number of nodes in the problem. Nodes which have coordinates which satisfy

$$|x_i^\alpha - x_i^\beta| \leq \text{gtol}\,|x_i^{max} - x_i^{min}| \quad \text{for } \alpha \neq \beta$$

Currently, the default tolerance is set as

$$\text{gtol} = \frac{1}{1000\,\sqrt{\text{NUMNP}}}$$

The default value may be replaced by a user specified value using the command structure

```
TIE GAP gtol
```

where `gtol` is a mesh fraction to limit the search.

When nodes are connected any specified, restrained boundary condition will be assigned to all interconnected nodes. Thus, it is only necessary to specify restrained boundary conditions and loadings for one of the nodes.

TITLe FEAP MESH MANIPULATION COMMAND MANUAL

```
titl,<on>
 titl,off
```

The `TITLe`,*off* command is used to suppress the print of headers on output pages produced by *FEAP*. It may be toggled on by entering the command with no parameter. This is provided to produce outputs devoid of header information every few lines, thus, the outputs are more readily usable by other programs or data conversions.

Appendix C

Contact manual pages

FEAP can treat some contact problems. The standard features included in the current release are summarized on the following pages. Currently, the contact may only be point to point for small deformation problems where a point is interpretted as a node on each surface. The second option is a node to segment penalty method. This option can consider the interaction between surfaces which undergo large motions and sliding. The implementation is limited to applications in which segments are the boundaries of low order elements (3-node triangles and 4-node quadrilaterals in two dimensions; 4-node tetrahedra and 8-node hexahedra in three dimensions). Both frictional and frictionless options exist for both formulations.

`cont <on,off,debug>`

The solution of contact problems is initiated by including a definition of the *surfaces*, interaction *pairs*, and interface *material* property descriptions. *FEAP* can solve two and three dimensional problems in which mechanical interactions can occur on specified boundary parts. For small deformation situations in which the nodes at the two boundary segments align a *node-to-node* strategy may be used. For cases in which the nodes do not align a *node-to-surface* solution strategy is available. In addition, a *node-to-rigid surface* capability exists.

In addition to specifying the contact surface data it may also be necessary to specify information about the contact solution strategy as part of the command language steps.

The contact surface data must include the definition of at least two surfaces (`SURF`ace command) which are expected to interact during the analysis as well as at least one pair set (`PAIR` command) which describes which surface is the master surface and which surface is the slave surface. The solution algorithm is implemented such that slave nodes interact with master facets. A facet is the boundary of an element. The pair data also defines methods to be used in searching for interactions, imposing a constraint to prevent penetration, and tolerances to be used. Optional data includes description of surface material property data (`MATE`erial command). If no material command is included the surfaces are assumed to be *smooth* and *frictionless*. The definition of a smooth surface is one with no asperities - a finite element mesh usually has small discontinuities in slope between contiguous elements. These discontinuities can lead to significant errors during large sliding and in some cases loss of contact due to search errors.

The contact data sets are terminated by an `END` command.

END FEAP CONTACT COMMAND MANUAL

```
end
```

The last contact data input command must be `END`. This terminates the input of contact surface definitions and returns to the control program, which may then perform additional tasks on the data or `STOP` execution.

Immediately following the contact `END` command any additional data required to manipulate the mesh (e.g., `TIE`, `LINK`, `ELINk`, `PARTition` `ORDEr`, `RIGId` and `JOINt` can be given prior to initiation of a problem solution using `BATCh` and/or `INTEractive`.

```
mate number
   standard
     friction coulomb value                user coulomb value
```

The MATErial command is used to define properties for contact interaction. Only one option is currently available: the standard option denoted by the data STANdard. If the MATErial command is not included as part of the contact data a standard model without friction is assumed. Friction may be added by including the FRICion command record with the parameters COULomb and a *value* of the frictional coefficient.

PAIR FEAP CONTACT COMMAND MANUAL

```
pair number
   nton s_surf m_surf
   ntor s_surf m_surf
   ntos s_surf m_surf
   tied s_surf m_surf q_surf
   solm s_type k_norm k_tang
   deta d_type k_norm k_tang
   mate m_type m_s m_m
   augm a_type m_s m_m
   tole none t_1 t_2 t_3                    thic cons t_1
```

The `PAIR` command is used to define which two surfaces are to be considered for contact detections. The pair `number` is used as a reference value only. Two types of contact algorithms are available: `NTON` considers interactions between a node on the slave surface and a node on the master surface (point-to-point contact); `NTOS` considers interactions between a node on the slave surface and a contact facet on the master surface. The slave surface reference number is specified by `s-surf` and the master surface reference number by `m-surf`.

The `NTON` solution method may only be used for contacts which occur in a coordinate direction. In addition, nodes on one contact surface must align with nodes on the other contact surface, restricting application to problems which have small deformation on the contact surface. This solution mode is similar to that which can be performed using a `GAP` element. Thus, for situations which involve very few contacting nodal pairs users should consider use of the gap element instead of a general contact surface.

The `NTOS` solution method permits large deformations on the contact surface. In addition large sliding can be accommodated, however, with node to surface treatments the sliding occurs on element surfaces and thus may be *non-smooth*. The contact surface for the *slave* side may be defined either as segments or as points. The *master* surface must be defined as by segments. In two dimensions these are line segments and in three dimensions they are surface facets. The node to surface treatment is effective only with low order elements - in two dimensions these are 3-node triangles or 4-node quadrilaterals and in three dimensions these are 4-node tetrahedra or 8-node bricks. Use of higher order elements with quadratic (or higher) displacements on boundaries should not be employed in conjunction with this contact type.

The NTOR solution method permits large deformation contact analysis between a *deformable slave surface* and a *rigid master surface*. Currently, only frictionless contact is permitted. The rigid surface may be a plane or circular surface for 2-d problems and may be a plane, cylinder or sphere in 3-d problems. The rigid surface is defined as described in the SURFace command page of the contact manual.

The TIED solution method permits large deformation analysis in which the surfaces defined by the *slave surface*, s_surf, and the *master surface*, m_surf, are tied together using a *dual mortar* Lagrange multiplier method. The only solution option currently available is SOLM LAGM. The parameter q_surf defines the number of quadrature points used to compute integrals over parts of each segment defined by a continuous interpolation on both surfaces.

The SOLM command defines the solution method to be used to impose the contact constraint. Currently a penalty method and a Lagrange multiplier method are implemented, consequently, s-type may be either PENA or LAGM with the parameters k-norm and k-tang having the values for the normal and tangential if required penalty parameters, respectively. An augmented solution strategy may be employed in combination with the penalty method using the AUGM option. This can be robust in that moderate values of the 'penalty parameters' may then be employed, thus reducing ill conditioning of the tangent matrix.

The MATErial command defines the material models to be used for the slave and master surface definitions. If only the first is given the slave and master are assumed to have the same properties.

The TOLErance command defines tolerances to be used during the solution phase. Three tolerance values are used:

Parameter	Description
t_1	Initial penetration check.
t_2	Gap opening considered to be contact.
t_3	Extension to contact facet in contact.

The THICkness command defines a constant non-zero gap allowed between the master and slave surfaces. Only one option exists: CONStang with a value specified by t_1. For example, this allow the contact to compensate for half thickness of a shell in contact between a solid and shell.

For node to surface treatments (NTOS) the PAIR command may be used twice for each contact surface pair, thus providing a *two pass* implementation of the constraint. Accordingly, the pair commands may be given as

```
PAIR 1
```

```
    NTOS n1 n2
    ...

  PAIR 2
    NTOS n2 n1
    ...
```

where `n1` and `n2` define the two surfaces which may interact in a contact mode.

WARNING: Contact is in a development stage and documentation is incomplete at this time.

SURFace FEAP CONTACT COMMAND MANUAL

```
surf number
   type <number>
     facet
        facet_data
     < terminate with blank record >
     bloc btype (td(i),i=1,n)
        1 x_1 y_1 z_1 (ndm reqd)
        2 x_2 y_2 z_2 (ndm reqd)
        < etc.  for number required >
     blen btype (td(i),i=1,n)
        side_type (id(i),i=1,m)
        < etc.  for data required >
     function type (td(i),i=1,n)
```

The SURFace command is used to define simply connect surfaces which will be considered for possible contact. It is necessary to have at least two surfaces which will be considered as a *contact pair* during solution steps. The surface `number` is used as a reference value to define pairs. The *type* data may be `LINE`, `TRIA`ngle, `QUAD`rilateral, `POIN`t or `RIGI`d. A `LINE` type is used for 2-dimensional problems to define contact facets which are either straight (*number = 2*) (default) or quadratic (*number = 3*) edges. A `TRIA` type is used for 3-dimensional problems to define surface facets which are 3-node triangles (default). A `QUAD` type is used for 3-dimensional problems to define surface facets which are 4-node quadrilaterals (default). The `POIN` type is used to define single slave nodes (default = 1). Points may be used in any dimension. Finally, the `RIGId` type defines a surface which retains its original shape throughout the analysis but may change in location or size according to a specified value. No *number* is required for this type.

FEAP permits a *two pass* solution strategy in which the slave and master definitions are switched in the second pass. For this class of problems each surface must be defined by appropriate boundary facets. The point type may not be used in a two pass solution strategy.

Once the type of surface facet is determined the facet data defining the individual surface elements is given. Facet data may be input using `FACE`t, `BLOC`k, `BLEN`d, or `REGI`on options. The `FACE`t option inputs a list of nodal connections for each facet. The `facet` data is given as:

nfac	Facet number.
ngen	Generator increment for facet nodes.
node-1	Global node number 1 for facet.
node-2	Global node number 2 for facet.
	etc. until proper number specified.

Generation is performed as described for elements in the mesh `ELEM` command.

The `BLOC` option is input in an identical manner as described for mesh blocks. The data sets are grouped as:

```
BLOCk SEGMent
  1  x1 y1 z1
  2  x2 y2 z2
     etc. for required  number of block nodes
```

Other `BLOC`k options exist to define the coordinate system to use, gap for the search for nodes, and region to restrict the search. The data options are:

Option	Data	Description
`GAP`	value	Gap value for search.
`CART`	-	Cartesian coordinate system.
`POLAr`	$x_0\, y_0\, z_0$	Polar coordinates centered at $x_0\, y_0\, z_0$.
`REGIon`	number	Region number to restrict search.

The `BLEN`d option is input in an identical manner as described for `SIDE`s in mesh blending. The data sets are grouped as:

```
BLENd SEGMent
  stype (list(i),i=1,n)
```

The *stype* options are `CART`, `POLA`, `SEGM`, and `ELLI` (see `SIDE` mesh manual page for *list* data required for each). Other `BLEN`d options exist to define the gap for the search for nodes:

Option	Data	Description
`GAP`	value	Gap value for search.

The `REGI`on type is used to generate point surfaces only. All nodes which are referenced by any element in the region *number* are assigned to the contact surface.

The RIGId type is used to specify a *master* surface which is not associated to any elements. Instead the surface is defined by a *function* giving its original shape and a motion amount by which the surfaces changes. Currently the function types are given by the command:

```
FUNCtion shape (list(i),i=1,n)
```

where shape may be: CYLInder; SPHEre; or CARTesian. A cylindrical or cartesian surface may be used in two or three dimensional analyses and a spherical one in three dimensional analysis. The cylindrical and spherical surfaces have the data lists:

List	Description
1	+1 for outward normal to cylinder,
	-1 for inward normal to cylinder.
2	R: Initial radius.
3	u: Displacement amplitude.
4	i:Proportional load number.

The deformed radius of the cylinder is given by

$$r(t) = R + u\, p_i(t)$$

where $p_i(t)$ is the proportional load value at time t. The *slave* surface may be points or facets but may not be another rigid surface.

A CARTesian rigid surface is a plane at constant values of one cartesian coordinate. The motion of the surface is in the specified coordinate direction. The data list required for a cartesian surface is:

List	Description
1	$+n$: direction of normal to plane in positive direction,
	$-n$: direction of normal to plane in negative direction.
2	X_n: Initial coordinate for plane.
3	u_n: Displacement amplitude.
4	i:Proportional load number.

The deformed position of the plane is given by

$$x_n(t) = X_n + u\, p_i(t)$$

The *slave* surface may be points or facets but may not be another rigid surface.

Appendix D

Solution command manual pages

FEAP has several options which may be used to solve problems. The solution strategy is based on a *command language* approach in which users write each step using the available commands. The following pages summarize the commands currently available in *FEAP*. These include options needed to solve most problems; however, provisions are also available for users to include their own solution routines through use of `UMACRn` subprograms. Methods to write and interface user routines to the program are described in the *FEAP* Programmers Manual.

BATCh/INTEractive FEAP SOLUTION COMMAND MANUAL

```
batc
inte
xxxx,yyyy,v1,v2,v3
```

The solution algorithm used by *FEAP* to solve problems is defined by a *command language program*. The command language program may be executed in either a *batch* or an *interactive* mode using the initial command `BATCh` or `INTEractive`, respectively. By properly specifying the commands following either of these modes, a very wide range of applications may be addressed – including both linear and non-linear, as well as, steady state and transient applications.

The name for the command `xxxx` is selected from the list contained in the following pages of this appendix. The description for the options for `yyyy` and `v1`, `v2`, `v3` also may be obtained from the manual entry for each command.

```
acce,,<n1,n2,n3>
acce,all
acce,coor,dir,xi,tol
acce,list,n1
acce,node,x1,x2,x3
acce,cmpl,<n1,n2,n3>
acce,imag,<n1,n2,n3>
```

The command `ACCE`leration may be used to print the current values of the acceleration vector as follows:

1. Using the command:

   ```
   acce,,n1,n2,n3
   ```

   prints out the current acceleration vector for nodes `n1` to `n2` at increments of `n3` (default increment = 1). If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only the first nodal acceleration is reported.

2. If the command is specified as:

   ```
   acce,all
   ```

   all nodal accelerations are output.

3. If the command is specified as:

   ```
   acce,coor,dir,xi,tol
   ```

   all nodal quantities for the coordinate direction `dir` with value equal to `xi` (within the tolerance `tol`) are output. The default for `tol` is 0.01 coordinate units.

   Example:

   ```
   acce,coor,1,3.5
   ```

   prints all the nodal acceleration vector which have $x_1 = 3.5 \pm 0.01$i units.

   This is useful to find the nodal values along a particular constant coordinate line.

4. If the command is specified as:

   ```
   acce,list,n1
   ```

   all nodal quantities contained in `list` number `n1` are output (see command `LIST` for specification of the list).

   Example:

   ```
   acce,list,3
   ```

   prints the nodal accelerations contained in list number 3.

5. If the command is specified as:

   ```
   acce,node,x1,x2,x3
   ```

   the single value for the acceleration *nearest* the coordinate with values `x1, x2, x3` is output. Only coordinates up to the dimension of the mesh need be specified.

6. If the command is specified as:

   ```
   acce,cmpl,n1,n2,n3
   ```

   the current *real and imaginary* part of a complex acceleration vector for nodes `n1` to `n2` at increments of `n3` (default increment = 1) is output. If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only node one (1) is reported.

7. If the command is specified as:

   ```
   acce,imag,n1,n2,n3
   ```

   only the current *imaginary* part of a complex acceleration vector for nodes `n1` to `n2` at increments of `n3` (default increment = 1) is output. If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only node one (1) is reported.

In order to output an acceleration vector it is first necessary to specify commands language instructions to compute the desired values for a transient analysis.

```
acti,,k1,k2,k3
acti,all
```

The first form of this command activates regions `k1` through `k2` in increments of `k3`; k2 $\geq$ k1. With the key word `all`, this activates all regions. See Mesh manual for the method to define mesh `REGI`ons. See also command `DEAC`tivate.

```
arcl,<xxxx,kfl,lfl>
```

The `ARCL`ength command computes a solution using an arclength continuation method. The basic use of the command is

```
ARCLength,,kfl,lfl
```

The `kfl` options are defined as follows:

kfl = 0:	Normal plane, modified newton solution.
	N.B. kfl = 0: defaults to kfl = 2.
kfl = 1:	Updated normal plane, modified newton solution.
kfl = 2:	Normal plane, full newton solution.
kfl = 3:	Updated normal plane, full newton solution.
kfl = 4:	Displacement control, modified newton solution.
kfl = 5:	Displacement control, full newton solution.

The `lfl` options are defined as follows:

lfl = 0:	Use current values for arclength and load direction
	(Initial default is calculated by first solution step).
lfl = 1:	Change current values for arclength and load direction.

The use of the `ARCLength` command requires additional data to set parameters. For all `kfl` a single parameter, called the damping value is specified using

```
BATCh
  ARCLength,,kfl
END
 <0,1>
```

Setting as zero initiates a *rattle* check to help traverse limit points, whereas, setting as one always uses a full step.

The use of `kfl = 1,2,3` contols the loading by the applied forces, whereas use of `kfl = 4,5` uses a nodal displacement parameter as the control. For displacement control it is necessary to give the node number, the specific degree of freedom and the displacement increment as input data. This requires an additional record in the input as

```
BATCh
  ARCLength,,kfl
END
 <0,1>
 node dof d_inc
```

`ARCLength` must be called once at the beginning of the solution commands when a nonlinear problem is to be solved using this method. With this call all flags will be set to perform an arclength solution. To turn arclength off after it has been activated issue `ARCL,OFF`.

For the calculation of load deflection curves specify `PRO`Portional load using default parameters; the actual load level is computed by `ARCL`.

Branch switching

The following table of additional input options are allowed.

xxxx	`kfl`	`lfl`
add	n1	tau
check	n1	
off		

In the above table `n1` denotes the number of the eigenvector to be included with the current solution. The **tau** is a scaling factor such that

$$\boldsymbol{u} \leftarrow \boldsymbol{u} + \frac{|\, \boldsymbol{u} \,|}{|\, \boldsymbol{E}\boldsymbol{v}_{n1} \,|\ \tau} \boldsymbol{E}\boldsymbol{v}_{n1} \tag{D.1}$$

where $\boldsymbol{u}$ is the current solution and $\boldsymbol{E}\boldsymbol{v}_{n1}$ is the $n1$-eigenvector.

If a branch-switching is to be performed it is necessary to calculate the eigenvectors associated with the bifurcation load first (Use of a shift on the tangent may be necessary as the tangent may be nearly singular, see `TANG`).

The command `ARCL,CHEC`k tells whether the stability point is a limit point or a bifurcation point (where the value returned should be zero).

The branch-switching is then initiated by the command `ARCL,ADD,n1,n2` which adds the n1-eigenvector to the current displacement field as shown above. `n2` is a scaling factor. If n2 is zero a scaling factor is automatically computed using the formula

$$\tau = 100\, \frac{(\boldsymbol{u} \cdot \boldsymbol{E}\boldsymbol{v}_{n1})}{|\, \boldsymbol{u} \,|\, |\, \boldsymbol{E}\boldsymbol{v}_{n1} \,|} + 1 \tag{D.2}$$

After the addition of eigenvector n1 to the displacement field a new equilibrium state must be computed on the secondary branch. This is performed by the following commands:

```
LOOP,,N
  TANG,,1
NEXT
```

ASSIgn FEAP SOLUTION COMMAND MANUAL

```
assi name num value
```

The command `ASSIgn` is used to set a specified value into a *FEAP* array. In interactive mode, u se of the command:

```
SHOW DICTionary
```

displays on the screen the list of currently allocated main arrays along with their name, size and precision. It is possible to change the value of individual entries in an array using the `ASSIgn` command.

Example:

It is possible to reset the value of a particular coordinate entry using the command

```
ASSIgn X  num x_val
```

where `X` is the name that *FEAP* gives to the array containing all the nodal coordinates for the mesh. The above would set the value for the position `num` to `x_val`. For coordinates the array is dimensioned as

```
real*8     x(ndm,numnp)
```

where `ndm` is the spatial dimension of the mesh and `numnp` is the total number of nodal coordinates in the mesh. Thus for node `n` and degree of freedom `i` the value of `num` would be:

```
num = ndf*(n-1) + i
```

`augm`

The command `AUGM`ent is used to perform augmented Lagrangian updates to solutions. Each element computes an update to the augmented data (defined in a user element) using `isw` equal to 10.

Augmented Lagrangian updates are normally used to accurately satisfy constraints during a solution.

AUTO time step FEAP SOLUTION COMMAND MANUAL

```
auto,time,imin,imax,maxr
auto,dt,dtmin,dtmax
auto,mate
auto,off
```

The `AUTO` command provides for automatic time step control, based on time step size and iteration properties or a material response indicator.

The form based on iteration limits is given as:

```
DT,,dt_value
AUTO DT   DTMIN  DTMAX
AUTO TIME IMIN   IMAX   MAXR
```

where `dt_value` is the size of the initial solution time increment; `DTMIN` and `DTMAX` are the minimum and maximum size of any time step allowed; `IMIN` determines the *minimum* number of iterations in an optimal range; the parameter `IMAX` defines the *maximum* number of iterations; and the parameter `MAXR` is the maximum number of retries with different time increments.

When the number of iterations per step is between `IMIN` and `IMAX` the routine maintains the current time step. Whenever the iteration exceeds the upper limit the time step is reduced, whereas when the iteration falls below the lower limit the time step is increased.

The command

```
AUTO DT DTMIN DTMAX
```

limits the range of the auto time stepping for the `AUTO TIME` command so that the values of time steps remain between `DTMIN` and `DTMAX`. This range should be preceded by a command `DT,,dt` in which the value of `dt` lies between the minimum and maximum. The time steps are adjusted according to:

For an increase

$$\Delta t_{new} = \min(\Delta t_{max}, 10^{[\log(\Delta t_{old})+0.2]})$$

For a decrease

$$\Delta t_{new} = \max(\Delta t_{min}, 10^{[\log(\Delta t_{old})-0.2]})$$

Example use:

```
DT,,0.024
AUTO DT   0.001 0.25
AUTO TIME 4 9 5
   ....
LOOP,,500
  TIME,,50.0
  LOOP,,15
    TANG,,1
  NEXT
NEXT
```

attempts to keep the number of iterations between a minimum of 4 and a maximum of 9. During any one increment the time step will be reduced or increased only a maximum of 5 times.

An alternative option is to limit the time step based on an indicator returned by the constitutive equation[1]. The command is given as:

```
AUTO MATErial
```

Each material model should return a value to define the setting of the time increment (see *FEAP Programmer Manual* for additional details).

When the command

```
AUTO OFF
```

is given auto time stepping is disabled.

[1]This is for use with user models and is not implemented in any of the standard models currently included in the program

```
back,,<dtnew>
```

The use of the `BACK` command will decrement the current time by `dt`, the current time increment. In addition, the previous value of the proportional loading will be recomputed, if necessary. The value of the current time and proportional loading are reported in the output (or to the screen). The back command also will recompute the dynamic state at the old time for time integration of the equations of motion, as well as, restore the stress data base for any elements with non-linear constitutive equations which require variables other than the displacement state to compute a solution.

As an option, it is possible to specify a new time increment for integrations to be continued. The value of `dtnew` is then used to perform the updates on the solutions in the same way as if the command `DT,,dtnew` were given. See manual on `DT` command for additional details.

`base`

This option computes the specified (during mesh description) static base modes for multiply supported structures which are to be solved using modal methods. One mode for every base degree of freedom which is to be excited must be obtained. This option should be used also for any structure in which the solution is obtained (by modal methods) with a specified *displacement* history at the degree of freedom. The history of the base motion is specified by a `PROP` load command.

See specification of base patterns in the `MESH INPUT MANUAL`.

```
bfgs,<xxxx>,nits,stol,etol
```

The `BFGS` command computes a solution using a quasi-Newton method with BFGS (Broyden-Fletcher-Goldfarb-Shano) updates. The command must be called at the beginning of an analysis - prior to the computation of any solutions with a tangent matrix. It is intended for use on problems with symmetric tangents. *FEAP* computes a new tangent at the beginning of each time step. Subsequently, the program will compute up to `nits` updates before computing another tangent (default nits=15). The value of `stol` is used in connection with a line search algorithm to compute a new solution (default stol=0.8). And `etol` is the BFGS energy tolerance (default etol=tol).

A typical algorithm using BFGS is:

```
LOOP,,N
  TANG
  BFGS,,10,0.8,1.d-10
NEXT
```

In the above a tangent will be computed and factored. BFGS would then perform 10 iterations, use line search on any step in which the energy was greater than 0.8 times a previous maximum, and exit when the energy is less than 1.d-10 times the initial energy in the step.

```
broyden,,nits
```

The `BROYden` command computes a solution using a quasi-Newton method with the Broyden algorithm updates for the inverse of the tangent matrix. The command must be preceded by at least one solution so that both a tangent matrix and a non-zero solution exists. It is primarily intended for use on problems with unsymmetric tangents. For problems with symmetric tangents the BFGS method may be used.

A typical algorithm using the Broyden method is:

```
LOOP,,N
  TANG,,1
  BROY,,10
NEXT
```

In the above a tangent will be computed and factored and a solution step performed to obtain a trial value. `BROYden` would then perform 10 iterations but will exit if convergence of the residual norm is attained. The `LOOP` may be used to repeat the solution process `N` times.

CHECk mesh FEAP SOLUTION COMMAND MANUAL

```
chec
```

The `CHECk` command requests a check of the mesh consistency. It is necessary for elements to have checking capability for the `isw = 2` option in order for `CHECk` to report results. Typical tests include jacobian tests at nodes, tests on node sequencing, etc.

If the jacobian is negative at all nodes the nodal sequencing has been in put in reverse order and should be resequenced. The 4-node solid elements contained in *FEAP* will attempt the resequencing automatically; however, the error is not corrected in the data input file so that it is necessary to use the check command each time the problem is executed.

```
comm,text
```

The `COMM`ent command permits a 15 character message (`text` option) to be displayed on the screen during batch solutions. This can assist in monitoring the progress of large problems to ensure that desired actions are being taken.

```
comp,,<n1,n2,n3>
comp,all
comp,coor,dir,xi,tol
comp,list,n1
comp,node,x1,x2,x3
comp,cmpl,<n1,n2,n3>
comp,imag,<n1,n2,n3>
```

The command `COMPonent` may be used to output the current values for the active degree of freedom component of the solution *generalized displacement* vector. The active component is set using the command

```
SET COMPonent dof
```

where `dof` is the component number in the solution so output. By default `dof = 1`.

Component output may then be performed as follows:

1. Using the command:

   ```
   comp,,n1,n2,n3
   ```

   prints out the current solution component for nodes `n1` to `n2` at increments of `n3` (default increment = 1). If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only the first nodal solution is reported.

2. If the command is specified as:

   ```
   comp,all
   ```

   all nodal solutions are output.

3. If the command is specified as:

   ```
   comp,coor,dir,xi,tol
   ```

all nodal quantities for the coordinate direction `dir` with value equal to `xi` (within the tolerance `tol`) are output. The default for `tol` is 0.01 coordinate units.

Example:

```
comp,coor,1,3.5
```

prints all the nodal components which have $x_1 = 3.5 \pm 0.01$i units.

This is useful to find the nodal values along a particular constant coordinate line.

4. If the command is specified as:

   ```
   comp,list,n1
   ```

   all nodal quantities contained in `list` number `n1` are output (see command `LIST` for specification of the list).

   Example:

   ```
   comp,list,3
   ```

   prints the nodal solutions contained in list number 3.

5. If the command is specified as:

   ```
   comp,node,x1,x2,x3
   ```

   the single value for the component *nearest* the coordinate with values `x1, x2, x3` is output. Only coordinates up to the dimension of the mesh need be specified.

6. If the command is specified as:

   ```
   comp,cmpl,n1,n2,n3
   ```

   the current *real and imaginary* part of a complex solution vector for nodes `n1` to `n2` at increments of `n3` (default increment = 1) is output. If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only node one (1) is reported.

7. If the command is specified as:

   ```
   comp,imag,n1,n2,n3
   ```

   only the current *imaginary* part of a complex solution vector for nodes `n1` to `n2` at increments of `n3` (default increment = 1) is output. If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only node one (1) is reported.

In order to output a solution vector it is first necessary to specify command language instructions to compute the desired values, e.g., for displacements perform a static or transient analysis.

```
cont,chec
cont,noch
cont,fric
cont,nofr
cont,pena,n,pen
cont,off
cont,on
```

The `CONT`act command may be used to activate and deactivate the contact logic during command language solutions using the `ON` and `OFF` options, respectively. The default mode is `ON`. It is necessary to describe the surfaces which may come into contact during the analysis when specifying the mesh data (i.e., the FEAP CONTACT USERS MANUAL). The contact logic may be skipped during execution of a command language program (even though the contact surfaces are defined) by using the `CONT,OFF` command.

When the command `CONT,CHEC`k is encountered in a solution sequence the program will determine which slave nodes are in contact with a master surface and readjust the profile of the equations of the *tangent* matrix. During each `TANG` or `UTAN` command no check on contact is performed.

During execution it is possible to reset the value of the penalty parameter on any contact pair, `n`, to a value of `pen`. This permits the adjustment of the penalty parameter from a smaller to larger value during iterations. For problems in which large deformations occur the convergence to a solution may lead to a large number of iterations when large penalty parameters are involved. On the other hand, the use of a lower penalty parameter may result in unacceptable large penetrations across the contact surface. In these situations, it is recommended that the penalty parameter be adjusted to larger values during the iteration process in each load. Similarly, the friction may be included or excluded using the `CONT,FRIC` or `CONT,NOFR` commands, respectively.

`conv,<on,off>`

The `CONVerge` solution command may be used in a solution to output the residual and energy norms to the *log* file (i.e., the file whose name begins with `Lxxx`). The command is included in the solution commands as:

```
BATCh
  CONVerge ON
    ... Commands to execute
END
```

The `ON` paramter is optional. It is also possible to discontinue collecting the data by inserting a command

```
CONVerge OFF
```

CXSOlve FEAP SOLUTION COMMAND MANUAL

```
cxso,,freq
```

This command is used to solve the set of damped linear equations given as

$$\mathbf{M}\,\mathbf{a} + \mathbf{C}\,\mathbf{v} + \mathbf{K}\,\mathbf{d} = \mathbf{f}\,\exp(i\,\omega\,t)\ , \tag{D.3}$$

where ω is specified in radians by `freq`. One solution is found for each frequency.

```
cyli,x1,x2,x3
```

The `CYLIndrical` solution command sets the form of the output variables (`CONT, VELOcity, ACCEleration, STREss` and `ESTRess` to a cylindrical coordinate mode. The form of plots for these variables is also in a cylindrical coordinate frame. In cylindrical mode subsequent outputs are interpreted as:

- 1 = radial (r) value
- 2 = tangential (θ) value
- 3 = axial (z) value.

Use the command `RECTangular` to return to a rectangular cartesian coordinate mode.

The values `x_1, x_2, x_3` are cartesian values for the origin of the cylindrical coordinate system.

Another option for a output mode is `SPHErical`.

DAMPing matrix FEAP SOLUTION COMMAND MANUAL

```
damp <unsy>
```

The command `DAMP`ing is used to compute a *damping* matrix. Each element computes a contribution to the damping in the array `S` when `isw` is 9. The damping matrix is stored in a sparse form and is symmetric if the command is issued as

```
DAMPing
```

alternatively, it may be unsymmetric if the command is given as

```
DAMPing UNSYmmetric
```

It should be noted that the current release version of *FEAP* does not use the damping matrix. However, user developed versions have included the damping matrix to compute the complex modes and frequencies of non-proportionally damped systems.

As an alternative option, Rayleigh damping is included in the small deformation *elements* for use in transient and modal solutions.

DATA FEAP SOLUTION COMMAND MANUAL

```
data,xxxx
```

During command language execution it is sometimes desirable to progressively change parameters, e.g., the time step size or the solution tolerance accuracy. This could become cumber- some and require an excessive number of commands if implemented directly. Accordingly, the `DATA` command may be used in instances when the time step or tolerance is to be varied during a `LOOP` execution. The permissible values for `xxxx` are `TOL` and `DT`. The actual values of the tolerance or time step size are given after the `END` statement using the data inputs specified in the `TOL` or `DT` manuals. For example, to vary time steps during a loop the commands:

```
LOOP,time,3
  DATA,DT
  TIME
  ...
  ...
NEXT,time
....
...
END
DT,,0.1
DT,,0.2
DT,,0.4
```

could be given to indicate three time steps with dt = 0.1, 0.2, and 0.4 respectively.

```
deac,,k1,k2,k3
deac,all
```

The first form of this command deactivates regions `k1` through `k2` in increments of `k3`; `k2` $\geq$ `k1`. With the key word `all`, this deactivates all regions. See Mesh manual for ways to define mesh `REGI`ons. See also Command Language Manual for `ACTI`vate.

```
debug,,ndebug
debug,on,ndebug
debug,off
```

Use of the `DEBUg,ON,ndebug` or `DEBU,,ndebug` command enables internal prints controlled by the `DEBU`g parameter in `common /debugs/ ndebug,debug`. The `ndebug` parameter is provided to allow setting of different levels for displaying prints. The debug print option is disabled using the `DEBUg,OFF` command.

```
dire
dire,bloc,v1
dire,spar
```

The DIREct command sets the mode of solution to direct for the linear algebraic equations generated by a TANGent or a UTANgent command. The direct solution is performed using a variant of Gauss elimination. The direct command without options requires the tangent matrix to fit within the computer memory available. In the interactive mode of solution a warning will be issued and control returned to the user to permit a selection of an alternate method of solution.

One option to reduce memory demands, is to solve the equations by a blocked direct procedure in which disk storage is used to store the tangent array. Memory then is only required to store two *blocks* of the tangent array. This option is selected using the DIREct,BLOCk,v1 command. If the parameter v1 is not input or is zero, a default value is set to the size necessary to assemble the tangent array as a sparse matrix. Normally this is quite small and it may be desirable to increase the size to reduce the I/O requirements of the blocks. Sufficient disk space is required to store the tangent array.

The *sparse matrix* solver exists for symmetric tangents only.

Another option is to solve the equations using an iterative method (see, the ITERative command language manual page).

DISPlacements FEAP SOLUTION COMMAND MANUAL

```
disp,,<n1,n2,n3>
disp,all
disp,coor,dir,xi,tol
disp,list,n1
disp,node,x1,x2,x3
disp,cmpl,<n1,n2,n3>
disp,imag,<n1,n2,n3>
```

The command `DISPlacement` may be used to print the current values of the solution *generalized displacement* vector as follows:

1. Using the command:

   ```
   disp,,n1,n2,n3
   ```

   prints out the current solution vector for nodes `n1` to `n2` at increments of `n3` (default increment = 1). If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only the first nodal solution is reported.

2. If the command is specified as:

   ```
   disp,all
   ```

   all nodal solutions are output.

3. If the command is specified as:

   ```
   disp,coor,dir,xi,tol
   ```

   all nodal quantities for the coordinate direction `dir` with value equal to `xi` (within the tolerance `tol`) are output. The default for `tol` is 0.01 coordinate units.

   Example:

   ```
   disp,coor,1,3.5
   ```

   prints all the nodal solution vector which have $x_1 = 3.5 \pm 0.01$i units.

   This is useful to find the nodal values along a particular constant coordinate line.

4. If the command is specified as:

   ```
   disp,list,n1
   ```

   all nodal quantities contained in `list` number `n1` are output (see command `LIST` for specification of the list).

   Example:

   ```
   disp,list,3
   ```

   prints the nodal solutions contained in list number 3.

5. If the command is specified as:

   ```
   disp,node,x1,x2,x3
   ```

   the single value for the displacement *nearest* the coordinate with values `x1, x2, x3` is output. Only coordinates up to the dimension of the mesh need be specified.

6. If the command is specified as:

   ```
   disp,cmpl,n1,n2,n3
   ```

   the current *real and imaginary* part of a complex solution vector for nodes `n1` to `n2` at increments of `n3` (default increment = 1) is output. If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only node one (1) is reported.

7. If the command is specified as:

   ```
   disp,imag,n1,n2,n3
   ```

   only the current *imaginary* part of a complex solution vector for nodes `n1` to `n2` at increments of `n3` (default increment = 1) is output. If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only node one (1) is reported.

In order to output a solution vector it is first necessary to specify commands language instructions to compute the desired values, e.g., for displacements perform a static or transient analysis.

DSOLve FEAP SOLUTION COMMAND MANUAL

```
dsol
```

The command `DSOLve` is used to specify when the equations generated by a `FORM` are to be solved. In *FEAP*, a solution of the equations is performed using a diagonal (lumped) mass for the equations

$$\mathbf{M}\, d\mathbf{u} = \mathbf{R}$$

where $\mathbf{M}$ is the diagonal mass and $\mathbf{R}$ the residual.

The command must be proceded by a `MASS LUMP` and a `FORM` command.

DT FEAP SOLUTION COMMAND MANUAL

```
dt,,v1
```

The `DT` solution command specifies the value of the time step for time dependent problems (i.e., transient or quasi- static problems). The value of `v1` indicates the time step to be used and should be greater or equal to zero. Generally, it is necessary to use a `TIME` solution command, in conjunction with the `DT` command, to advance the time and compute proportional loading values if necessary.

`echo,<on,off>`

The `ECHO` solution command may be used in a batch solution to display in the text window the sequence of commands executed during solution The command is included in the batch commands as:

```
BATCh
  ECHO ON
    ... Commands to execute
END
```

The `ON` paramter is optional. It is also possible to discontinue the screen displays by inserting a command

```
ECHO OFF
```

in the batch sequence. The `ECHO` command may also be given in interactive mode, but has no effect until a `BATCh,END` pair is executed.

```
eige,<vect>,k1,k2
  eige,mass,k1,k2
  eige,norm,k1,k2
```

The use of the `EIGE`lement command permits the computation of the eigenvalues associated with the element `k1` tangent array. If `k1` is zero or negative the last element in the mesh is used. If `k2` is greater than zero the eigenvalues of the element mass are reported, otherwise the stiffness matrix is used. Optionally the command `EIGE MASS` may be used.

The normal modes for

$$\mathbf{K}\,\boldsymbol{\phi} + \mathbf{M}\,\boldsymbol{\phi}\,\Lambda$$

may be used to compute the eigenvalues and vectors for element `k1`.

It is assumed that the arrays are symmetric and have real eigenvalues. This option is useful during solution of problems by explicit methods to compute critial time size or in element development to study the spectral properties of the element, including number of zero eigenvalues or those associated with some parameter. Use of the `VECT` parameter reports both the eigenvalues and eigenvectors for the element.

```
eigv,nn,<n1,n2,n3>
eigv,coor,idir,xi,nn
eigv,list,n1,nn
eigv,all,n1,nn
eigv,dofs,<list>
```

The command `EIGVector` may be used to print the current values of eigenvector vector `nn` as follows:

1. Using the command:

   ```
   eigv,nn,n1,n2,n3
   ```

   prints out the eigenvector `nn` for nodes `n1` to `n2` at increments of `n3` (default increment = 1). If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only the first nodal solution is reported.

2. If the command is specified as:

   ```
   eigv,coor,idir,xi,nn
   ```

   all nodal quantities for the coordinate direction `idir` with value equal to `xi` are output.

   Example:

   ```
   eigv,coor,1,3.5,2
   ```

   prints all the nodes in eigenvector 2 which have $x_1 = 3.5$.

   This is useful to find the nodal values along a particular constant coordinate line.

3. If the command is specified as:

   ```
   eigv,list,n1,nn
   ```

   all nodal quantities contained in `list` number `n1` are output (see command `LIST` for specification of the list).

   Example:

```
eigv,list,3,4
```

prints eigenvector 4 nodes contained in list number 3.

4. If the command is specified as:

```
eigv,all,nn
```

all nodal solutions for eigenvector `nn` are output.

In order to output a solution vector it is first necessary to specify commands language instructions to compute the desired values, e.g., for displacements perform a static or transient analysis.

For problems which have different partitions. The degrees of freedom to include in the eigen-computation may be specified with the `EIGV,DOFS` command. The list following the command is given for all degrees of freedom as 1 for any degree of freedom to include and 0 for those to exclude. For example the command

```
eigv,dofs,1,1,0
```

for a problem with three degrees of freedom would include only the first two in the eigenproblem.

```
else expression
```

The `ELSE` command may be used with a matching pair of `IF-ENDIf` commands. The `expression` is optional and is used to control the actions taken during the solution. If the expression is absent the commands between the `ELSE` and `ENDIf` are executed. If the expression evaluates to be positive then the commands contained between the `IF` and the `ELSE` or `ENDI` are executed, otherwise solution continues with a check of the next `ELSE` For example, the sequence

```
ZEROA
 ...
IF  10-a
  tang,,1
  ZEROA
ELSE b
  pause
ELSE
  form
  solv
ENDIf
INCRA
 ...
```

would compute a tangent, residual, and solution increment if `10-a` is positive; otherwise the solution increment is computed using a previous tangent. The parameter a may be computed using a function command. For example,

```
FUNCtion ZEROA
  a = 0
END
```

would zero the counter `a`.

```
FUNCtion INCRA
  a = a + 1
END
```

would define a function which increments `a`.

```
end
```

The last batch command must be `END` or `QUIT`. This terminates the current execution sequence and returns the program the main driver, which may then perform additional solution tasks on the same data, modify the data, enter a new problem, or `STOP` execution. The use of `END` causes a restart file to be updated for subsequent resumptions of execution with the current status preserved.

Immediately following the end command any data required by statements in the *command language program* should appear when a batch execution is performed.

Additional solution steps may be performed by including additional `BATCh-END` or `INTEractive-END` pairs.

ENDIf FEAP SOLUTION COMMAND MANUAL

```
endi
```

The `ENDIf` command is used with a matching `IF` command to terminate the control construction. For example, the sequence

```
ZEROA
 ...
IF  10-a
  tang,,1
  ZEROA
ELSE b
  pause
ELSE
  form
  solv
ENDIf
INCRA
 ...
```

would compute a tangent, residual, and solution increment if `10-a` is positive; otherwise the solution increment is computed using a previous tangent. The parameter `a` may be computed using a function command. For example,

```
FUNCtion ZEROA
  a = 0
END
```

would zero the counter `a`.

```
FUNCtion INCRA
  a = a + 1
END
```

would define a function which increments `a`.

`ener` `tang mate ma`

The energy command is used to output current values for the momentum and energy values. Two forms for the command are available. Use of

```
BATCh
  ENERgy
END
```

outputs the values for all parts of the problem. Alternatively,

```
BATCh
  ENERgy MATErial ma
END
```

outputs values for material number `ma` only.

```
epri
```

The use of the `EPRI`nt command outputs the last element matrix (S) and vector (P). This may be used after `TANG`ent, `UTAN`gent, `MASS`, or `DAMP`ing commands.

```
erro,stre
erro,ener
```

The command `ERROr` is used to perform error assessment calculations of finite element solutions. The command requires that an element have computations for the `isw = 11` option. Prior to use nodal stresses must be computed. Errors may be projected on the basis of `stre`ss norms or `ener`gy norms.

Example usage:

```
TANG,,1
STRE,NODE
ERRO,ENER
STRE,ERRO
```

N.B. This options does not produce outputs for standard *FEAP* elements.

```
exit
```

The last interactive command must be `EXIT` or `QUIT` (they may also be abbreviated as `E` or `Q`. This terminates the command language execution and returns the program to perform additional tasks on the same data, change the data, enter a new problem, or `STOP` execution. The use of `EXIT` causes a restart file to be updated for subsequent resumptions of execution with the current status preserved.

For interactive execution, using `INTE`ractive, any additional data will be requested as needed.

```
expl
```

The use of the EXPLicit command permits the computation of solutions associated with an explicit Newmark integration scheme. It replaces the normal solve routines and is operational only for diagonal mass matrices. It is to indicate a TRANsient explicit solution, then solutions may be achieved using the sequence.

```
TRAN,EXPL
MASS,LUMP      ! Form lumped mass array
LOOP,,no_steps
  TIME
  FORM
  EXPL
NEXT
```

Note that no iterations are required for traditional explicit methods, however if *FEAP* computes a solution with a zero residual then convergence is assumed and an exit from the current loop occurs. In the above algorithm this would omit any time steps not yet performed. This may be corrected by using the modified algorithm

```
TRAN,EXPL
MASS,LUMP      ! Form lumped mass array
LOOP,,no_steps
  TIME
  LOOP,,1
    FORM
    EXPL
  NEXT
NEXT
```

Also the mass type *must* be lumped. It is then desireable (though not absolutely required) to have each material set request a lumped element mass.

`expo`

This command is used to export part of the tangent matrix and residual to another program or a file. It requires a user to write part of a routine. The results from the other program may be imported using the `IMPOrt` command.

```
forc,,<n1,n2,n3>
forc,all
forc,coor,dir,xi,tol
forc,node,x1,x2,x3
```

The command `FORCe` may be used to output the nodal forces.

The command may be given as follows:

1. Using the command:

   ```
   forc,,n1,n2,n3
   ```

   prints out the current force vector for nodes `n1` to `n2` at increments of `n3` (default increment = 1). If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only the first nodal force is reported.

2. If the command is specified as:

   ```
   forc,all
   ```

   all nodal forces are output.

3. If the command is specified as:

   ```
   forc,coor,dir,xi,tol
   ```

   all nodal quantities for the coordinate direction `dir` with value equal to `xi` (within the tolerance `tol`) are output. The default for `tol` is 0.01 coordinate units.

   Example:

   ```
   forc,coor,1,3.5
   ```

   prints all the nodal force vector that has $x_1 = 3.5 \pm 0.01$i units.

   This is useful to find the nodal values along a particular constant coordinate line.

4. If the command is specified as:

   ```
   forc,node,x1,x2,x3
   ```

   the single value for the force *nearest* the coordinate with values `x1, x2, x3` is output. Only coordinates up to the dimension of the mesh need be specified.

```
form
form,acce
form,expl
```

The `FORM` command computes the residual for the current time and iteration of a solution. *FEAP* is a general nonlinear program and computes a residual for each solution by subtracting from any applied loads: (1) The force computed for the stresses in each element, often called the *stress divergence* or *internal force* term; (2) If the problem is dynamic the inertia forces.

At the end of each computation *FEAP* reports the value of the current residual in terms of its Euclidean norm, which is the square root of the sum of squares of each component of force.

If the `ACCEleration` option is present an acceleration is computed by solving the equation:

$$\boldsymbol{M}\,\boldsymbol{a} \;=\; \boldsymbol{R} \tag{D.4}$$

where $\boldsymbol{M}$ is a consistent mass or a lumped mass computed by the `MASS` command and must be computed before the specification of the `FORM` command. This option may be used to compute consistent accelerations for starting a transient analysis using the Newmark type integration algorithms when initial forces or initial displacements are specified.

If the `EXPLicit` option is present *FEAP* computes a solution to the equations of motion (momentum equations) using an explicit solution option. Prior to using the `FORM,EXPLicit` command it is necessary to specify the explicit solution option using the `TRANsient,EXPLicit` command. Explicit solutions are conditionally stable, thus, a critical time step must be estimated before attempting a solution. An estimate to the critical time step may be obtained using the maximum wave speed in the material, c, and the closest spacing between nodes, h. The maximum time step used must be less or equal to h/c.

```
func,name
```

The use of the `FUNC`tion command is used to execute a pre-defined function. To use this option there must be a file `name.fcn` which contains a set of parameter expressions which are to be executed. This may be used to change the values of parameters occurring in subsequent commands.

```
geom
geom,on
geom,off
```

The command GEOMetric stiffness command is used in two ways. The first is to compute a *geometric* stiffness matrix for use in linear buckling analysis. This option is performed when no parameters are appended to the command. A parameter `imtyp` is set to 2 and each element then computes a contribution to the geometric stiffness in the array `S` when `isw = 5`.

A geometric stiffness matrix may be used for eigencomputations (see solution command SUBSpace). Reported eigenpairs correspond to linearized buckling for a loading multiplied by the eigenvalue. Not all elements have this feature.

The second use of the option is to enable and disable the geometric stiffness during `TANG` and `UTAN` computations. For many problems the inclusion of the geometric stiffness during early iterations of a Newton type solution can lead to divergent results. The geometric matrix may be disabled during early iterations using the `GEOM,OFF` command and then enabled for later iterations using the `GEOM,ON` command. A typical example is:

```
LOOP,time,nsteps
  TIME
  GEOM,OFF
  LOOP,newton,3
    TANG,,1
  NEXT
  GEOM,ON
  LOOP,newton,25
    TANG,,1
  NEXT
NEXT,time
```

where three iterations are performed with no geometric stiffness and, later, additional iterations with the geometric stiffness. At convergence each loop can terminate before the number of specified iterations. If this occurred in the first loop one additional iteration would be made in the second loop.

```
get param
```

The `GET` command is used to determine the current value for the parameter with the name `param`. The command is given as

```
GET param
```

For example, if the following parameters are set to compute beam properties during mesh input

```
PARAmeter
  b = 0.1
  h = 0.2
  a = b*h
  i = a*h*h/12
```

one may display the value for `i` during solution steps using the command

```
GET i
```

Another option is

```
PARA j = 0
LOOP,,20
  TIME
  PARA j = j+1
  TANG,,1
NEXT
GET j
```

would tell how many iterations were performed to get convergence.

```
help
help name
```

The use of the `HELP` command will produce a list of the currently implemented commands at the current manual level. The manual level is set by the command `MANU`al, n where n is an integer between 0 and 3. The help feature is useful only in an interactive mode of solution. If additional information is required for a specific command it is necessary for the user to consult the users manual.

The use of the help command as `HELP name`, where `name` is a solution command provides a brief description for alternative uses of the command, including required options. For example

```
HELP TANG
```

displays the options:

```
Help requested for Solution    : tang

tang: form symmetric tangent
tang,,1: form rhs and solve.
tang line 1 shift,value: line search on value
tang eigv 0 shift: with no mass/damping added
tang nume 0 shift: numerical tangent
```

This is available only in an interactive mode. The command may also be given in plot mode using plot command names.

```
hill                              hill,stress
   hill,tangent
   hill,file
```

The command `HILL` is used to perform computation of averaged properties on a representative volume element (RVE) using the Hill-Mandel hommogenization method. The approach may be used to compute the properties for either a thermal problem and a small or finite deformation solid problem. For this case a typical RVE may be defined and subjected to either displacement or periodic boundary conditions where the boundaries of the RVE are subjected to a homogeneous thermal gradient: a strain (for small deformations) or a deformation gradient (for finite deformations). The description for the gradients is given in Sect. 5.5.7.

The computation of the averaged properties is determined using a solution command

```
TANGent,,1
HILL <TANGent,     >
```

to obtain the homogenized stress (or thermal flux) and associated tangent array or

```
TANGent,,1
HILL STREss
```

to obtain stress or thermal flux averages only. The properties for non-linear problems may also be obtained, however, in this case it is necessary to define time variations to the deformation by a proportional loading.

If a time series of values is stored in a file and the RVE uses material models with history variables the command should be given as

```
LOOp,,no_times
  TIME
  HILL FILE
  LOOP,,no_its
    TANG,,1
  NEXT
```

```
  HILL <TANGent,STREss>
NEXT
```

`hist,<clab,n1,n2>`

The use of the `HISTory` command permits the user to keep a history of the previously executed commands and use this history to re-execute specific commands. The history command has several different modes of use which permit easy control of the execution of commands while in an interactive mode (use is not recommended in a batch execution). The following options are available:

clab	n1	n2	Description
read			Input the list of commands which were 'saved' in a previous execution. Warning, this command will destroy all items currently in the 'history' list, hence it should be the first command when used.
save			Save the previous 'history' of commands which have been 'added' to the 'history' list on the file named 'Feap.hist'.
add			Add all subsequent commands executed for the current analysis to the 'history' list (default).
noad			Do not add subsequent commands executed to the 'history' list.
list	x	x	List the current 'history' of statements. 'n1' to 'n2', (default is all in list).
edit	x	x	Delete items 'n1' to 'n2' from current 'history' list.
xxxx	x	x	Re-execute commands 'n1' to 'n2' in the current 'history' list. (Note: 'xxxx' may be anything not defined above for 'clab' including a blank field).

Use of the history command can greatly reduce the effort in interactive executions of command language programs. Since it is not possible to name the file which stores the

history commands, it is necessary for the user to move any files needed at a later date to a file other than `Feap.his` before starting another analysis for which a history will be retained. Prior to execution it is necessary to restore the list to file `Feap.his` before a `HIST,READ` command may be issued.

Note that the history of commands will not be saved in `Feap.his` unless a command `HIST,SAVE` is used. It is, however, possible to use the history option without any read or save commands.

```
iden,,<n1,n2>
```

The `IDEN`tity command is used to specify an identity matrix. In general it may be used in conjunction with an eigen computation to compute the eigenpairs of a stiffness matrix. When `n1` and `n2` are specified they indicate the node range (i.e., `n1` to `n2`) for which the identity matrix is to be specified. When used in this mode all boundary restraints must be omitted and a shift used to compute any zero eigenvalues.

IF FEAP SOLUTION COMMAND MANUAL

```
if expression
```

The `IF` command must be used with a matching `ENDIf` command. Optionally, one or more `ELSE` commands may be included between the `IF-ENDIf` pair. The `expression` is used to control the actions taken during the solution. If the expression evaluates to be positive then the commands contained between the `IF` and the `ELSE` or `ENDI` are executed, otherwise solution continues with a check of the next `ELSE` For example, the sequence

```
ZEROA
 ...
IF  10-a
  tang,,1
  ZEROA
ELSE
  form
  solv
ENDIf
INCRA
 ...
```

would compute a tangent, residual, and solution increment if `10-a` is positive; otherwise the solution increment is computed using a previous tangent. The parameter `a` may be computed using a function command. For example,

```
FUNCtion ZEROA
  a = 0
END
```

would zero the counter `a`.

```
FUNCtion INCRA
  a = a + 1
END
```

would define a function which increments `a`.

`imag`

The `IMAGinary` solution command is used to set output of solution displacement, velocity, acceleration or stress, etc. quantities to their imaginary parts. To use this option it is necessary for a problem to be solved in `*COMplex` mode (See Sect. 1).

For example

```
BATCh
  IMAGinary
  DISP,,1,10
END
```

outputs the imaginary part of the displacement solution for nodes 1 to 10. The real part is set using the `REAL` command which is the default mode.

```
impo
```

This command is used to import results from another program. Results may be exported to the other program using the `EXPOrt` command. The export module requires a user to write part of a routine.

```
init,disp
init,rate
init,spin,w1,w2,w3
```

Non-zero initial displacements or rates (e.g., velocities) for a dynamic solution may be specified using the `INITial` command. The values for any non-zero vector are specified after the `END` command for batch executions and may be generated in a manner similar to nodal generations in the mesh input. For interactive execution prompts are given for the corresponding data. Accordingly, the vectors are input as:

```
n1,ng1,v1-1, . . . ,v1-ndf
n2,ng2,v2-1, . . . ,v2-ndf
etc.
```

where, `n1` and `n2` define two nodes; `ng1` defines an increment to node `n1` to be used in generation; `v1-1`, `v2-1` define values for the first degree of freedom at nodes `n1`, `n2`, respectively; etc. for the remaining degree of freedoms. Generated values are linearly interpolated using the `v1` and `v2` values; etc. for the remaining degree of freedoms. Note that `ng2` is used for the next pair of generation records. If a value of `ng1` or `ng2` is zero or blank, no generation is performed between `n1` and `n2`.

When using the `SPIN` option, `w1,w2,w3` are the angular velocities of the body rotating about the origin. This initializes all active nodes.

```
iter,,,icgit
iter,bpcg,v1,icgit
iter,ppcg,v1,icgit
iter,tol,v1,v2
```

The `ITERative` command sets the mode of solution to iterative for the linear algebraic equations generated by a `TANGent`. Currently, iterative options exist only for symmetric, positive definite tangent arrays, consequently the use of the `UTANgent` command should be avoided. An iterative solution requires the sparse matrix form of the tangent matrix to fit within the available memory of the computer.

The solution of the equations is governed by the relative residual for the problem (i.e., the ratio of the current residual to the first iteration in the current time step). The tolerance for convergence may be set using the `ITER,TOL,v1,v2` option. The parameter `v1` controls the relative residual error given by

$$(\mathbf{R}^T\mathbf{R})_i^{1/2} \leq v1\,(\mathbf{R}^T\mathbf{R})_0^{1/2}$$

and, for implementations using `PETSc` the parameter `v2` controls the absolute residual error given by

$$(\mathbf{R}^T\mathbf{R})_i^{1/2} \leq v2$$

The default for `v1` is `1.0d-08` and for `v2` is `1.0d-16`. By default the maximum number of iterations allowed is equal to the number of equations to be solved, however, this may be reduced or increased by specifying a positive value of the paramter `icgit`.

The symmetric equations are solved by a preconditioned conjugate gradient method. Without options, the preconditioner is taken as the diagonal of the tangent matrix. Options exist to use the diagonal nodal blocks (i.e., the $ndf \times ndf$ nodal blocks, or reduced size blocks if displacement boundary conditions are imposed) as the preconditioner. This option is used if the command is given as `ITERative,BPCG`. Another option is to use a banded preconditioner where the non-zero profile inside a specified half band is used. This option is used if the command is given as `ITERative,PPCG,v1`, where `v1` is the size of the half band to use for the preconditioner.

The iterative solution options currently available are not very effective for poorly conditioned problems. Poor conditioning occurs when the material model is highly nonlinear (e.g., plasticity); the model has a long thin structure (like a beam); or when structural elements such as frame, plate, or shell elements are employed. For compact

three dimensional bodies with linear elastic material behavior the iterative solution is often very effective.

Another option is to solve the equations using a direct method (see, the DIREct command language manual page).

```
jump,label
```

The command `JUMP` is used to specify where a LOOP,NEXT set is to exit.

Example:

If a set of multiple loop commands is used, it is sometimes necessary to *jump* beyond the inner loop at convergence. The set of commands

```
LOOP level_1 3
   ...
   LOOP newton 3
     TANG,,1
   NEXT j_label
   ....
NEXT level_1
....
JUMP j_label
....
```

would be executed until convergence of the *Newton* loop satisfied the convergence taolerance when it would transfer to the `JUMP` statement. The label *j_label* serves to match the jump location. Note, normally a label on a `NEXT` command is not used, such as the *level_1* on the outer loop.

LIST FEAP SOLUTION COMMAND MANUAL

```
list,,n1
   <values>                          list,mate,n1
   <values>
```

The command `LIST` is used to specify lists of nodes for outputs or material sets for plots. It is possible to specify up to three different lists of each type where the list number corresponds to `n1` (default = 1). The list of nodes to be output or material sets for plots follows with 8 values per record. The input terminates when less than 8 values are specified.

List of node outputs are obtained by specifying the command:

```
name,list,n1
```

where `name` may be `DISP`lacement,`VELO`city,`ACCE`leration, or `STRE`ss and `n1` is the desired list number.

Example:

```
BATCh
  LIST,,1
END
1,5,8,20

BATCh
  DISP,LIST,1  !Outputs nodes 1,5,8,20
   ...
END
```

The material set option is given with the `mate` argument and plots are activated using the plot command

```
PLOT LIST n
```

where `n` is the list number to be plotted. Individual sets `ma` in the list may be displayed using the `PLOT MATE ma` command followed by the display command.

Example:

```
BATCh
  LIST MATE 1
END
1 3

BATCh
  PLOT LIST 1
  PLOT CONT 1  !Outputs materials 1 and 3 only
  PLOT MATE 3
  PLOT CONT 1  !Outputs material 3 only
   ...
END
```

```
loop,<xxxx>,ni
loop,infinite>
```

The LOOP command must be used in conjunction with a matching NEXT command.

A LOOP-NEXT pair is used to repeat the execution of a set of commands ni times. The LOOP appears first, followed by one or more commands then a NEXT command. The loop-next commands may be nested to a depth of 8. That is,

```
LOOP,level_1,n1
  LOOP,level_2,n2
    LOOP,level_3,n3
       etc. to 8-levels
    NEXT
  NEXT
NEXT
```

is permitted. If desired, the xxxx may be used (as above) to describe the type of next which is being closed, i.e., NEXT,time would indicate the end of a time loop.

In some solution programs the actual number of times the LOOP is executed cannot be determined in advance. In these cases the form

```
LOOP INFInite
   .....
NEXT
```

Used alone this form is dangerous as there is no way for the loop to terminate. One option is to specify a *time* when solution should cease. This may be given as

```
LOOP INFInite
  TIME,,time_stop
    ....
NEXT
```

The TIME command causes time to be advanced by the current time increment set in a DT command. When the value of the time reaches the value specified by time_stop the loop will be terminated. Another option to terminate is using the JUMP command

```
LOOP INFInite
    ....
NEXT j_label
JUMP j_label
  ....
```

However, this form must have a statement in the loop that sets a convergence limit.

During interactive executions, `LOOP-NEXT` commands are not executed until the matching `NEXT` command is input. In this way a set of statements may be grouped and executed together.

MANUal FEAP SOLUTION COMMAND MANUAL

```
manu,level
```

The `MANU`al command will set the `level` of help commands shown when the command `HELP` is given in any solution mode. The levels are: 0 = basic; 1 = intermediate; 2 = advanced; 3 = expert. The default level is 0.

```
mass
mass lump
mass unsy
```

The command `MASS` is used to compute a consistent or a diagonal *mass* matrix. Each library element computes a contribution to both the consistent mass diagonal mass as part of its standard operations. The type of global mass assembled is controlled by the parameter on the `MASS` command, with `LUMP` producing a diagonal mass, `UNSYmetric` storing an unsymmetric consistent mass; and any other option the symmetric consistent mass.

A symmetric consistent mass or a lumped (diagonal) mass may be used for eigencomputations (see command `SUBS`pace). Both may also be used for transient solutions computed using the explicit method (see command `TRAN`sient,`EXPL`icit). They are not needed for other time integration methods.

The both the symmetric and unsymmetric consistent forms are stored in sparse form. At present the unsymmetric form is provided to assist in user developed modules.

```
mate,,n1
```

The `MATE`rial command is used to indicate which material number is to be active during subsequent contour or fill plots. The `n1` value is the material number, and a value of zero indicates all materials are to be displayed. (Default: n1 = 0).

MEMOry FEAP SOLUTION COMMAND MANUAL

```
memo
```

The use of the `MEMO`ry command will display the amount of memory currently used from the blank common, together with the total size available in the version of *FEAP* loaded.

MESH FEAP SOLUTION COMMAND MANUAL

`mesh`	`mesh filename`

The use of the `MESH` command permits the redefining of the mesh data. Nodal forces may be redefined during solution to consider additional loading distributions. In addition, nodal coordinates, values of temperatures, angles of sloping boundaries, constants, material set numbers for elements, and material properties ties may be redefined. It is also permitted to change the boundary restraint codes or the element connection data.

For convenience the actual mesh data can be defined in a separate file and input using the command

```
MESH filename
```

where `filename` is the data file. This file must contain an `END` statement to exit back to the next solution command. Using this option one can have multiple sets of changes in data in a convenient way.

MODAl solution FEAP SOLUTION COMMAND MANUAL

```
moda
```

The solution of transient linear problems may be performed using either the time stepping algorithms defined by the TRANsient command language statement or using mode superposition using the MODAl statement.

The mode superposition routine in *FEAP* solves only the second order transient problem

$$\boldsymbol{M}\,\ddot{\boldsymbol{u}} \;+\; \boldsymbol{K}\,\boldsymbol{u} \;=\; \boldsymbol{f}\;. \tag{D.5}$$

To use the modal command it is necessary to first solve the eigenproblem to the above problem. The command language statements to solve the eigenproblem are:

```
MASS
TANGent
SUBSpace,,nfreq
```

where nfreq is the number of modes to be included in the solution. Non-zero initial conditions for the modal solution are obtained from specified nodal initial conditions which are input using the INITial command as:

```
INITial,DISPlacements
 and/or
INITial,RATEs
```

The initial conditions need not be specified if they are zero. Once the above steps are provided, the transient solution is accomplished using the commands:

```
LOOP,,ntime
  TIME
  MODAL
NEXT
```

For each time step the modal solutions are reprojected to the nodes so that all graphics and output commands may be used. For example, time history plots may be output for a set of nodes (e.g, see the page for the plot command TPLOt) by inserting the command:

```
TPLOt
```

before the first time loop.

MONOlithic solution FEAP SOLUTION COMMAND MANUAL

```
mono
```

The solution of problems performed with all degrees of freedom active is termed a *monolithic* solution. This is the default when no partitions are used (see mesh manipulation and solution commands `PARTition`). When a solution is performed using partitions with the degrees of freedom assigned to different parts it is possible to return to a monolithic solution by using the command:

```
MONOlithic
```

and continuing with normal solution steps using `TANG` or `UTAN`.

```
newf
newf,forc
newf,disp
newf,both
newf,zero
```

The use of the NEWForce command will set a fixed pattern of nodal forces and displacements to the values of the current pattern in boundary force and displacements plus the previous "fixed" pattern. That is:

1. For degree-of-freedoms where forces (loads) are specified:

$$\texttt{f0(i,1)} = \texttt{f(i,1)} * \texttt{prop(t)} + \texttt{f0(i,1)}$$

2. For degree-of-freedoms where displacements are specified:

$$\texttt{f0(i,2)} = \texttt{u(i)}$$

   where `f0(i,n)` is the *fixed* pattern forces and displacements, `f(i,1)` is the pattern specified in force boundary loads, `prop(t)` is the current value of the proportional loading at the current time t, and `u(i)` is the current displacement value.

Use of the command `NEWF FORCe` only sets the array ${}_{\texttt{f}}\texttt{0}(*,\texttt{1})$; whereas use of the command `NEWF DISPl` only sets the array ${}_{\texttt{f}}\texttt{0}(*,\texttt{2})$; use of the command `NEWF BOTH` sets both forces and displacements.

When execution is initiated the values in `f0(i,n)` are all zero. NOTE at restart they again will become all zero so that caution must be exercised at any restart where NEWForce had been used in generating the results.

The `f0(i,n)` may be reset to zero using the `NEWForce ZERO` command (values are not updated). N.B. This only affects the current partition degree of freedoms.

`next,<xxxx>`

The `NEXT` command `must` be used in conjunction with a `LOOP` command.

A `LOOP-NEXT` pair is used to repeat the execution of a set of commands. The `LOOP` appears first, followed by one or more commands then a `NEXT` command. The loop-next commands may be nested to a depth of 8. That is,

```
LOOP,level-1,n1
  LOOP,level-2,n1
    LOOP,level-3,n1
       etc. to 8-levels
    NEXT
  NEXT
NEXT
```

is permitted. If desired, the `xxxx` may be used (as above) to describe the type of next which is being closed, i.e., `NEXT,time` would indicate the end of a time loop.

During interactive executions, `LOOP-NEXT` commands are not executed until the `NEXT` command is input. In this way a set of statements may be grouped and executed together.

NOPRint FEAP SOLUTION COMMAND MANUAL

```
nopr
```

The use of the `NOPRint` command will discontinue most output of commands. Plot results and element outputs will normally still be reported. The use of `PRINt` will cause the output of execution descriptions to again be reported. The default value is `PRINt` at start of command language program execution.

```
ntan,mate,<n1>
ntan,elem,<n1>
ntan,off
```

Numerically compute the tangent matrix using residuals. Use of the `MATErial` option computes the tangent for all elements belonging to the specified material number `n1`. Individual element tangent `n1` may be computed using the `ELEMent` option. This option is intended for help in computing correct tangent matrices for elements. It is not recommended for general use. In particular, if discontinuous load paths exist (e.g., plasticity loading-unloading) incorrect answers may result from the perturbation technique used on the residuals.

For transient problems the numerical tangent for an element relies on correct use of the dynamic factors through the use of `ctan(1:3)` parameters [Consult programmer manual[12] for details]. The solution commands are given as:

```
DT,,dt                  ! set a non-zero time increment
TRANS <NEWMark, BACK>   ! for order 2 or 1 problems
TIME                    ! set values of ctan(1:3)
NTAN ELEM n
```

where `dt` is a numerical value of the time increment and `n` the number of the element to evaluate.

The `OFF` option is used to discontinue use of numerical computations of tangents.

`omeg` `omeg,hz`

The `OMEGa` solution command is used to set the frequncy of periodic solutions. The value may be set in radians per second using the form

```
OMEGa,,value
```

or in Hertz using the form

```
OMEGa HZ value
```

The `value` parameter may be an expression to permit solutions over a frequency range. For example

```
PARAmeter om = 100
LOOP,,11
  OMEGA HZ om
  tang,,1
     ... ! Outputs or other commands
  PARAmeter om=om+100
NEXT
```

would produce 11 solutions for $\omega = 100$ to $\omega = 1000$ Hz.

```
opti,cont
opti,off
opti
```

This option performs optimization of the ordering of unknowns for the direct profile equation solver. For optimization of the current system, the command `OPTImize` is given alone. To return to the default ordering obtained from the mesh input order the command is given as `OPTImize,OFF`. Dynamic optimization can be done during a contact solution by issuing the command as `OPTImize,CONTact`. For each geometric computation a profile is checked and if possible optimized (this has not worked reliably on all problems)

```
outm,<bina>
```

The use of the OUTMesh command writes an output file which contains some of the mesh data. Two modes of output are possible. Using the OUTMesh command without any parameters outputs the data in text mode in a file which has the same name as the input file with an added extender *opt*. Filenames (with the extender) are limited to 18 characters. This format is useful if the mesh has been constructed using TIE, LINK and/or profile optimizations using the OPTImize command. The output file contains: Coordinates (coor), element connections (elem), boundary codes (boun), and the forced values (forc). In addition the file is set for an interactive mode of execution.

The second mode of output is a binary file which has the same name as the input file with an added extender *bin* (18 character limit). This mode is produced using the **OUTM**esh,BINAry command. The file contains: Coordinates (coor), element connections (elem), boundary codes (boun), forced values (forc), temperatures (temp), angles (angl), and material data (mate). The binary form of data is used in FEAP by preparing an input file which has the form:

```
BINAry,filename.bin
(optional mesh data)
 ...
END
 ...
INTEractive or BATCh
STOP
```

This form is useful when TIE, LINK, and/or OPTImize have been used. It also may be used on very large models which are time consuming to generate the input data.

```
outp,array
```

The use of the `OUTPut` command permits current values in some arrays to be output in a format that can be processed by MATLAB. Ouputs are made in a sparse matrix form with each record given as:

```
i  j  value(i,j)
```

Vectors have `j = 1` for all components. Diagonal arrays are given as

```
i  i  value(i,i)
```

The options for the `array` are:

Name	Description
`TANG`	Symmetric tangent matrix
`UTAN`	Unsymmetric tangent matrix
`MASS`	Symmetric consistent mass matrix
`CMAS`	Symmetric consistent mass matrix
`UMAS`	Unymmetric consistent mass matrix
`LMAS`	Lumped (diagonal) mass matrix
`DAMP`	Consistent symmetric damping matrix
`CDAM`	Consistent symmetric damping matrix
`UDAM`	Consistent unsymmetric damping matrix
`DR`	Residual vector
`FORM`	Residual vector

For arrays associated with complex form both the real and imaginary parts may be output. By default the *real* part is output, however if the solution command

```
IMAGinary
```

precedes the output of the array the imaginary part is output and this will occur until the real form is issued by the command

```
REAL
```

Output files have the form

```
REAL_XXXX_nnn
IMAG_XXXX_nnn
```

where `XXXX` is the name of the array and `nnn` is a sequence counter (maxium value is 999).

Output of the `TANGent` or `UTANgent` matrix can be done for either the unfactored or the factored form. To output an unfactored tangent the form

```
TANGent,,-1
OUTPut TANGent
```

should be used. If it is desired to have a factored tangent then the command form is

```
TANGent
OUTPut TANGent
```

All arrays are output to a file with the same name as the array. For example:

```
OUTPut TANGent
```

by default will create a file with the name `REAL_TANG_001`.

```
para letter
para
< After end command >
  letter = expression
  list
```

The use of the PARAmeter command permits the input of data parameters during execution. These are normally used during the data input phase to vary the input values. For example, parameters may be set and used during proportional loading table inputs. Use of LIST will display the parameters and values for all letters set previously to non-zero values. Only 1 or 2 character parameters are permitted and should be lower case letters and numerals (first character must be a letter) only.

Two forms of the command are possible:

1. Use of

```
BATCh
  ....
  PARA
  ....
END
  a = 12
  b = a/9
  list
```

   inputs a parameter set from data after the END command.

2. Use of

```
BATCh
  ....
  LOOP,,25
    ....
    PARA a = a+1
    ....
  NEXT
END
```

resets the command repeatedly during the loop.

```
part,,n1
part
```

The command `PART`ition is used to set the active partition to `n1`. The default at initiation of execution is n1 = 1.

Partitions are used to perform operator split or staggered solutions on the global finite element problem. Each degree of freedom may be assigned to a partition after input of the mesh data (e.g., following the `END` command for the mesh input) using a command:

```
part
 <list of dofs in partition 1>
 <list of dofs in partition 2>
    ...
 <list of dofs in partition n>
     ! Blank record
```

where the number of partitions is between 1 and 4 and each list uses a non-zero number for an active dof and a 0 or blank for an inactive one. For example, the solution of a two dimensional thermo-mechanical problem in which the first 2 dof are for the displacements and the 3rd dof is the temperature, is given as

```
part
  1,1,0
  0,0,1
```

and, thus, assigns the displacement degrees of freedom to partition 1 and the temperature to partition 2. During solution, a mechanical step is specified by

```
part,,1
```

and a thermal solution by

```
part,,2
```

Any solution commands given apply to the active partition.

In interactive mode use of the `PART` command without a number displays the current active partition number.

```
paus
```

The PAUSe command is used in the inner loop of a Newton solution strategy to permit interactive control in situations where divergence may occur. The command is used in the sequence

```
LOOP,,<Newton number of iterations>
  TANG,,1
  PAUS
NEXT
```

The solution will pause whenever the energy of the computed solution is 100 times or more of the initial energy in the step. The user may then indicate whether or not to continue with the solution. If the step is terminated transfer is made to the statement following the next statement of the Newton loop (may be a prompt).

```
plot,quantity,[n1,n2,n3]
plot
```

In *FEAP*, screen and hard copy PostScript plots may be made for several quantities of interest.

A `PLOT` may be specified to initiate interactive graphics outputs. After entering graphics mode a prompt will be displayed. At this time, `quantity` and the `n1`, `n2`, and `n3` values may be specified. Alternatively, a `PLOT,quantity,n1,n2,n3` command also may be issued while in interactive execution mode (this is the only option for batch executions).

See the PLOT Manual for admissible values of `quantity` and parameters.

```
prin
prin,on
prin,off
prin,comm
prin,data
prin,less
prin,<xxxx>
```

The use of the PRINt restores printing turned off by the NOPRint command or resets the level of printing to the screen. In interactive mode the use of PRINt,OFF eliminates all printing to the screen and the output file. PRINt,ON restores all printing.

Use of PRINt,LESS reduces the amount of command information displayed. Use of PRINt,COMMand restores command prints if they have been disabled by a PRINt,OFF or NOPRint,COMM.

The PRINt,DATA option restores printing of mesh data to the output file.

The default value is PRINt,ON.

The specification of:

```
xxxx = TANGent
xxxx = UTANgent
xxxx = CMASs
xxxx = LMASs
xxxx = RESIdual
```

will output the diagonal entries for the specified array. This may be useful in debugging elements, etc. The DEBUg option is also available.

```
proj <cont>
```

The PROJect plot command controls the continuity of element projections between different material sets. By default *FEAP* computes the projection to element nodes from the computation points (e.g., quadrature points) to be continuous only within each individual material set. The projected values output for *paraview* display are averaged between material sets. Screen displays in *FEAP* may also be averaged by using the command

```
PROJect CONTinuous
```

before giving commands

```
STREss NODE ....
```

or

```
PLOT <STREss,STRAin,FLUX,HISTory, etc.>
```

Giving the command

```
PROJect
```

restores the discontinuous form.

PROPortional load FEAP SOLUTION COMMAND MANUAL

```
prop,,<n1>
prop,,<n1,n2>
prop,off,<time,dt>                    prop,user,<n1,n2>
```

In the solution of transient or quasi-static problems in which the `TIME` command is used to describe each new time state the loading may be varied proportionally. At each time the applied loading will be computed from:

```
F(i,t) = f0(i) + f(i)*prop(t)
```

where `f0(i)` is a fixed pattern which is initially zero but may be reset using `NEWF`orce; `f(i)` are the *force* and *displacement* nodal conditions defined during mesh input or revised during a `MESH` command; and `prop(t)` is the value of the proportional loading at time `t`. Up to ten different proportional loading factors may be set. Individual proportional factors may be assigned to degree of freedoms using the mesh command `FPRO`portional. If the assigned proportional loading number defined by `FPRO` is zero, the sum of all active sets is taken as the proportional factor. If the proportional loading number defined by 'fpro' is 'n1' then the value defined by set 'n1' only is used. This permits individual nodal loads to be controlled by particular loading factors.

For the form `PROP,,N1`, the specific proportional loading is defined by specifying one set of records for each of the 'n1' values up to a maximum of 10 (default for `N1` is 1, that is, `PROP` alone inputs one set). For the form `PROP,,N1,N2`, the specific data for proportional loading `N1` to `N2` are input. Thus, `PROP,,2,2` will assign the input data set to proportional loading number 2.

Each set contains the following data:

```
type, k, t-min, t-max, a(i),i=1,4
```

The proportional loading may be specified as:

1. Type 1 is defined by:

$$Prop(t) = a_1 + a_2\,(t - t_{min}) + a_3\,\left(\sin(a_4\,(t - t_{min})\right)^k \tag{D.6}$$

   for all time values between `t_min` and `t_max`. The value of `k` must be a positive integer all other parameters are real with the argument of the *sine* function given in *radians*.

If a blank record is input the value of `t_min` is set to zero; `t_max` to 10^8; a(1), a(3), and a(4) are zero; and a(2) is 1.0 - this defines a ramp loading with unit slope.

Example: The following defines a linearly increasing load to a maximum of 1.0 at time 10 and then a linearly decreasing load to time 20, after which the loading is zero:

```
prop,,1,2
1  0  0.0  20.  0.  0.1  0.0  0.0  ! Set 1
1  0 10.0  20.  1. -0.2  0.0  0.0  ! Set 2
```

Note that the negative slope is twice that of the increasing ramp.

Also, if individual nodal forced conditions (e.g., displacements or loads) have been assigned to proportional load number 1 (using the mesh 'fpro' command), the first input record result is used, whereas if assigned to number 2 the second input record is used. When no assignment is made or a zero is specified for the dof using the `FPRO`, `EPRO`, and/or `CPRO` mesh commands the sum of the records is used.

2. Type 2 is a table input. The input is as follows:

```
prop,,3    ! Input proportional loading 3 only
2,nn  (default nn is 1)
t_1   ,p_1,   t_2   ,p_2   , ... ,t_nn  ,p_nn
t_nn+1,p_nn+1,t_nn+2,p_nn+2, ... ,t_2*nn,p_2*nn
           ! etc., terminate with blank record
```

The time points must be in an increasing order. After the input of t_1, a zero time value terminates the input. Linear interpolation is used between each pair of times, t_i and t_{i+1}, for the two values, p_i and p_{i+1}. This option is particularly useful for specifying cyclic loading.

Example:

```
BATCH
  PROP,,3
END
  2,4
  0.,0.  1.,1.  3.,-1.  5.,1.
  7.,-1.  8.,0.  0.,0.
         ! blank record
```

gives a cyclic loading with linear behavior between the times 0. and 8. and is zero thereafter.

3. Type 3 is a roll and slide option that allows for translation and rotation of objects. The input is given as:

```
3 tmin tmax v_1 v_2 omega x_0 y_0
```

where v_1 and v_2 are translational velocity in the 1 and 2 directions, ω is angular velocity in degrees/time, and x_0 and y_0 are the origin about wihch rotation occurs. For each node '(a)' to which the roll and slide option is applied the nodal displacement is determined from

$$
\begin{aligned}
r_a &= \sqrt{(x_a - x_0)^2 + (y_a - y_0)^2} \\
\theta_a &= \tan^{-1}(\frac{y_a - y_0}{x_a - x_0}) \\
u_a &= v_1\, t + r_a[\cos(\theta_a + \pi\omega t/180) - \cos(\theta_a)] \\
v_a &= v_2\, t + r_a[\sin(\theta_a + \pi\omega t/180) - \sin(\theta_a)]
\end{aligned}
$$

with the arguments of the *sine* and *cosine* functions given in *radians*.

4. Type 4 is a sawtooth cyclic option to apply large numbers of repeated cyles of loading. The input is given as:

   ```
   4 l tmin tmax p_0 dt
   ```

5. Type 5 is a polynomial loading with input given by

   ```
   5 tmin tmax a_1 a_2 a_3 a_4 a_5
   ```

   The proportional load is computed as

   $$Prop(t) = a_1 + a_2 \cdot (t - t_{min}) + a_3 \cdot (t - t_{min})^2 + a_4 \cdot (t - t_{min})^3 + a_5 \cdot (t - t_{min})^4$$

6. Type 6 is an exponential type of loading with input given by

   ```
   6 l tmin tmax a_1 a_2 a_3 a_4 a_5
   ```

   The proporional load is computed as

   $$Prop(t) = a_1 + a_2 \exp(a_3(t - t_{min})) + a_4(\sin a_5(t - t_{min}))^l$$

   with the argument of the *sine* function given in *radians*.

7. Type 7 is a piecewise velocity variation between discrete time values. The input is identical to the Type 2 method. Each piecewise velocity is integrated to yield piecewise quadratic variation of the proportional factor $Prop(t)$ between the specified time points.

During solution it is possible also to delete all defined proportional loads using the solution command:

Parameter	Type	Description
J	Integer	Loading type
IE	Integer	User integer
TMIN	Real*8	Minimum time
TMAX	Real*8	Maximum time
A(5)	Real*8	User real array
T	Real*8	Current time
UPRLD	Real*8	User proportional load
ISW	Integer	Switch: 1 = input; 2 = compute

Table D.1: PROP: Parameters for user proportional load

```
PROP OFF time dt
```

The additional parameters time and dt set the value of the time and time increment to new values, respectively. Following an OFF specification, new proportional load functions may again be defined using any of the methods defined above.

User Function

Additional types of proportional loading may be added to the program by writing and compiling the module

```
SUBROUTINE UPROP(J,IE,TMIN,TMAX,A, T, UPRLD, ISW)
```

to the main program Where the parameters are defined according to Table D.1. The parameter J permits users to define as many types of proportional loads as desired. A user proportional loading is accessed using the command

```
BATCh
  PROPld USER nprop
END
j ie tmin tmax a(1:5)
```

where the values of j, ie, etc. correspond to the arguments of the module.

For computation of non-zero values ofttUPRLD, users are responsible for ensuring that T is between TMIN and TMAX. In addition retention of data needed to describe a load is the responsibility of the user (except for the A(5) array).

Example:

Definition of an exponential given by:

$$uprld(t) = C_0 \exp[-C_1 (t - t_{min})]$$

```
      subroutine uprop(j,ie,tmin,tmax,a, t, uprld, isw)

!     User proportional load function

      implicit   none

      integer       :: j,ie, isw
      real (kind=8) :: tmin,tmax,a(5), t, uprld

!     Function 1
      if(j.eq.1) then

!       Output parameters
        if(isw.eq.1) then
          write(iow,2000) tmin,tmax,a(1),a(2)

!       Compute load
        else
          if(t.ge.tmin .and. t.le.tmax) then
            uprld = a(1)*exp(-a(2)*(t-tmin))
          else
            uprld = 0.0d0
          endif
        endif
      endif

!     Format
2000  format(/10x,'T_min =',1p,1e12.5/' T_max =',1p,1e12.5
     &       /10x,'C_0   =',1p,1e12.5/' C_1   =',1p,1e12.5/)
       end
```

Table D.2: PROP: User proportional load

Let `A(1)` $= C_0$ and `A(2)` $= C_1$. Assign the function to `J` $= 1$. A simple subprogram is given in Table D.2

PVIEw FEAP SOLUTION COMMAND MANUAL

`pvie`	`pvie,<time,file>`

The `PVIEw` solution command is used to output data files for processing by *Paraview*.

For example

```
BATCh
  PVIEw
END
```

produces a file named `feap_paraview.vtu` containing data which may be displacyed using the Paraview program. The use of the form

```
BATCh
  PVIEw file_name
END
```

permits specifying a specific file name for the output. Finally the form

```
BATCh
  PVIEw TIME
END
```

may be used to output files in time dependent solutions. The file names are set from the *plot* file name specified at start of a *FEAP* solution (`fplt`) and appended by a numerical sequence ranging from `00000` to `99999`. Thus the first file would be designated as `fplt00001.vtu` the second as `fplt00002.vtu` and so forth for subsequtne outputs.

```
quit
```

The last solution command may be `QUIT`, or just `Q`. This terminates the command language solution and returns the program to perform additional tasks on the same data, modify data, enter a new problem, or `STOP` execution. The `QUIT` command causes termination of execution without writing the restart files (they remain the same as at the beginning of execution if they existed).

```
rayl,freq,zeta,w1,w2
rayl,,a0,a1
```

This command is used to set the Rayleigh damping values for a modal solution. Use of the option

```
rayl,freq,zeta,w1,w2
```

assigns the damping values in the damping matrix

$$\mathbf{C} = a_0 \, \mathbf{M} + a_1 \, \mathbf{K} \tag{D.7}$$

so that the damping ratio is `zeta` at frequencies `w1` and `w2`. The other option sets the parameters directly.

```
reac,,<n1,n2,n3>
reac,coor,idir,xi
reac,all
reac,list,n1
reac,file
```

Nodal reactions may be computed for all nodes in the problem and reported for nodes `n1` to `n2` at increments of `n3` (default increment = 1). If `n2` is not specified then only the values for node `n1` are output. When both `n1` and `n2` are not specified only total sum information is reported.

If the command is specified as:

```
reac,coor,idir,xi
```

prints all nodal reactions for the coordinate direction `idir` with value equal to `xi`. This option is useful in finding the nodal values along a particular constant coordinate line.

Example:

```
reac,coor,1,3.5
```

will print all the nodal reactions which have $x_1 = 3.5$.

All reactions may be output using the `REAC,ALL` command as:

```
reac,all
```

In addition to computing the reaction at each degree of freedom an equilibrium check is performed by summing the values for each degree of freedom over all nodes in the analysis. The sum of the absolute value of the reaction at each degree of freedom is also reported to indicate the accuracy to which equilibrium is attained. It should be noted that problems with rotational degrees of freedom or in curvilinear coordinates may not satisfy an equilibrium check of this type. For example, the sum for the radial direction in an axisymmetric analysis will not be zero due to the influence of the *hoop stresses*.

In addition to sums over all the nodes a sum is computed for only the nodes output. This permits the check of equilibrium on specified series of nodes, or the computation of the applied load on a set of nodes in which motions or restraints are specified.

If the command is specified as:

```
reac,list,n1
```

all nodal reactions contained in `list` number `n1` are output (see command `LIST` for specification of the list).

Example:

```
reac,list,3
```

will print all the nodal reactions which are in list number 3.

The `FILE` option outputs reactions to the restart save file with the extender `.reacxxxx` (starting from `reac0000`). These may be used as input in Mesh (see Mesh `REAC`tion command).

```
read,xxxx
```

The `READ` command may be used to input the values of displacements and nodal stresses previously computed and saved using the `WRITE` command - it is primarily used for plots related to deformations or nodal stresses. It is not intended for a restart option (see `RESTart`) but may be used to restore displacement states of linear and non-linear elastic elements (or other elements with no data base requirements) for which reactions, stresses, etc. may then be computed.

The values of `xxxx` are used to specify the file name (4-characters only), manipulate the file, and read displacements and nodal stresses. The values permitted are:

```
xxxx = wind: Rewind current file.
xxxx = back: Backspace current file.
xxxx = clos: Close current file.
xxxx = disp: Read displacement state from current file.
xxxx = stre: Read nodal stress state from current file.
xxxx = Anything else will set current filename.
```

Only four characters are permitted and only one file may be opened at any time. Files may be opened and closed several times during any run to permit the use of more than one file name.

A `READ` input is created using the `WRITE` command which has identical options for `xxxx` except for the backspace option.

```
real
```

The `REAL` solution command is used to set output of solution displacement, velocity, acceleration or stress, etc. quantities to their real parts. This is the default mode of outputs and is only required for solution of problems in complex (`*COMplex`) mode.

For example

```
BATCh
  REAL
  DISP,,1,10
END
```

outputs the real part of the displacement solution for nodes 1 to 10. The imaginary part is set using the `IMAGinary` command.

```
rect
```

The solution command `RECTangular` is used to set the mode of output for the variables (`CONT, VELOcity, ACCEleration, STREss` and `ESTRess` to a rectangular cartesian form. Other options for outputs are are `CYLindrical` and `SPHErical`.

The `RECTangular` form is the default.

```
renu
```

The use of the `RENU`mber command writes to the output the renumbering map from `OPTI`mize together with the nodal coordinates.

```
resi
```

The `RESIdual` command computes the residual for the current time and iteration of a solution. *FEAP* is a general nonlinear program and computes a residual for each solution by subtracting from any applied loads: (1) The force computed for the stresses in each element, often called the *stress divergence* or *internal force* term; (2) If the problem is dynamic the inertia forces.

At the end of each computation *FEAP* reports the value of the current residual in terms of its Euclidean norm, which is the square root of the sum of squares of each component of force.

The residual is not used in any solution step, however, the values may be output using SHOW DR. For solution steps the command FORM should be used.

```
rest,,k
```

A restart may be made using the results from previous analyses (which are retained in the restart read file specified at the start of each analysis). After entering the command language program the restart may be specified. If the previously computed problem was "dynamic", it is necessary to specify the `TRAN`sient command prior to issuing a `REST`art command in order to restore the velocity and acceleration states. If the previous problem was static and the new analysis is to be continued as a dynamic calculation, the `REST`art is issued before the `TRAN`sient command (since the previous analysis did not write a velocity or acceleration state to the restart file.

The `k` parameter is used to restart with files generated using the `SAVE` command with the same specified `k` parameter.

The use of the restart option requires considerable care to ensure that the previous results used are proper. At the termination of any analysis which computes a solution state a new file is saved on the restart write file specified at the start of the analysis. If the last analysis performed is for a different problem than the current one an error will result.

If no new solution state is computed during command language execution (e.g., only plotting is performed) no restart file is written to the specified file set - the previous restart file is retained on the original file set.

SAVE FEAP SOLUTION COMMAND MANUAL

```
save,,k
```

The `SAVE` command may be used to save the current solution state and history data for use as a restart file. In the solution of complicated nonlinear problems where difficulties are expected in achieving convergence (e.g., a solution step may produce an overflow which terminates execution) a restart state may be saved on the disk for each converged state. The problem may then be initiated from any of the saved states and continued.

The `k` parameter is optional. If omitted, a counter is used to increment from the last value. However, it may be specified as a 3 digit integer appended to the name of the current save file named when the problem was started. In interactive mode if the specified file exists a user is allowed to specify an alternate name or to replace the old file with current data.

```
scre,on
scre,off
```

The `SCREen` command permits graphics to be disabled (`OFF` option) or enabled (`ON` option) during interactive mode solutions. The `OFF` option permits the solution of a problem in which PostScript graphics outputs are created but the graphics is not displayed on the screen. The default mode is `ON`.

```
set comp dof
set disp node dof
set temp dof
```

The `SET` command may be used to:

1. Set the `component` number for subsequent output of displacements.

   For example, using

   ```
   SET COMP 2
   COMP,,1,10
   ```

   would output only the second component of the displacements for nodes 1 to 10.

2. Set the value of the *specified displacement* for degree of freedom *dof* at `node` to the current solution value for this node and degree of freedom. A specified displacement is a quantity specified during mesh input by a `DISP`, `EDIS`, or `CDIS` command.

   The command is given as:

   ```
   SET DISP node dof
   ```

   This is useful for assigning a specified nodal value when a boundary condition is changed from *free* to *fixed* (i.e., the boundary code from a `BOUN` is changed from zero to non-zero).

   When using subsequent solution steps it is generally necessary for the proportional load value associated with this displacement component to be set to unity.

3. Move solution values for degree of freedom `dof` into the TEMP array. The TEMP array is the array specified during mesh input using the mesh command `TEMP`. Many of the exisiting elements in *FEAP* use the `TEMP` values for temperatures and pass them to each element in a local array `TL(nen)`. Thus, if at some stage a temperature has been compupted in `dof` it may be used for thermal stress analysis by placing it in TEMP without recoding any other steps.

```
show
show,dict
show,elem
show,name,n1,n2
```

The use of the `SHOW` command will display the current solution status for the problem. Values include the `time`, `dt`, `tol`, `prop`, Maximum energy in step, current energy in step, augmented factor, and command print status (T=on; F=off).

The `SHOW,DICT` command will produce a table of the current array name and number together with their length and precision. All main solution arrays are dynamically allocated out of available memory in the computer.

The `SHOW,name,n1,n2` option (where *name* is the array name displayed using the `SHOW,DICT` command) outputs the current values of the `name` array entries between `n1` and `n2`. If the range entries are both zero (omitted), the entire array is output.

The `SHOW,ELEM`ent provides a one-line description of the currently loaded *user* elements.

```
smooth,,nsmth
```

The command `SMOOth` is used to improve the quality of meshes with poor initial layouts. A simple node averaging scheme is used to reposition nodes in the mesh and is expressed as

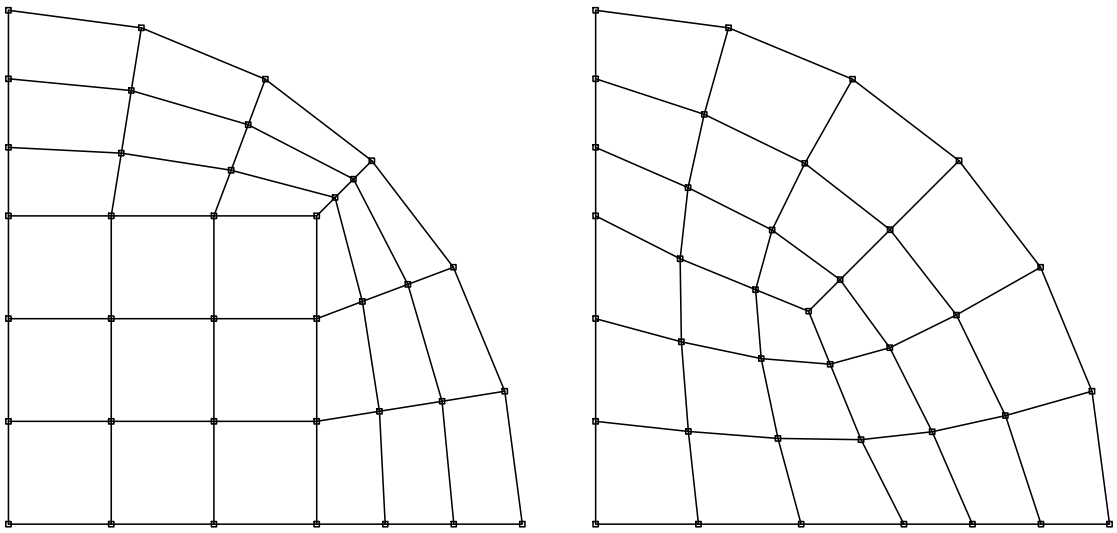
$$\mathbf{x}_n = \frac{1}{N_e} \sum_{i=1}^{N_e} \mathbf{x}_i$$

where the list N_e includes all nodes associated with the elements connected to node n (except the node itself). The algorithm is iterative and `nsmth` defines the number of iterations performed over the mesh.

For example, the quadrant of a circle has an initial mesh as shown in Fig. D.1(a). Using the solution command

```
SMOOth,,25
```

yields the mesh shown in Fig. D.1(b).



(a) Initial mesh (b) Smoothed mesh

Figure D.1: Mesh smoothing using SMOOth command

```
solv,<line,v1>
```

The command `SOLVe` is used to specify when the equations generated by a `FORM` are to be solved. In *FEAP*, a direct solution of the equations is performed using a profile storage with a variable band (active column) method of solution or by an iterative method.

In the solution of some nonlinear problems it is possible to obtain convergence for a wider range of loading and time step size using a "line search". The line search may be requested by placing `LINE` in the second field of the solve command. The parameter `v1` is the required energy reduction to preclude a line search being performed (if the current energy is larger than `v1` times the minimum energy in the step so far, a line search is performed). If not specified `v1` defaults to 0.8 (recommended values are between 0.6 and 0.9). Line search should never be used in a linear problem since extra evaluations of the residual are required during the line search.

SPHErical FEAP SOLUTION COMMAND MANUAL

```
sphe,x1,x2,x3
```

The `SPHErical` solution command sets the form of the output variables (`CONT, VELOcity, ACCEleration, STREss` and `ESTRess` to a cylindrical coordinate mode. The form of plots for these variables is also in a cylindrical coordinate frame. In cylindrical mode subsequent outputs are interpreted as:

- 1 = radial (r) value
- 2 = tangential (θ) angle
- 3 = meridian (ϕ) angle.

Use the command `RECTangular` to return to a rectangular cartesian coordinate mode.

The values `x_1, x_2, x_3` are cartesian values for the origin of the cylindrical coordinate system.

Another option for a output mode is `CYLIndrical`.

```
stre,,<n1,n2,n3>
stre,all
stre,aver
stre,coor,idir,xi
stre,<node,n1,n2,n3>
stre,<on,off>
stre,erro
```

The `STREss` command is used to output stress results in elements `n1` to `n2` at increments of `n2` (default = 1), or at nodes using *projected* values. Thus, two options exist for reporting stress values. These are:

1. Stresses may be reported at selected points within each element. The specific values reported are described in each element type. In general elements report values at gauss points. The values at all points are reported when the command `STREss,ALL` is used.

2. For solid elements results may be reported at nodes using the `STREss,NODE` option. A projection method using stresses at points in each element is used to compute nodal values. In general, nodal values are not always as accurate as stresses within elements. This is especially true for reported *yield* stresses where values in excess of the limit value result in the projection method employed. For a mesh producing accurate results inside elements this degradation should not be significant.

3. The command specified as:

   ```
   stre coor idir xi
   ```

   prints all nodal stresses for the coordinate direction `idir` with value equal to `xi`. Example:

   ```
   stre coor 1 3.5
   ```

   will print all the nodal stresses which have $x_1 = 3.5$. This is useful in finding the nodal values along a particular constant coordinate line.

4. The element stress (and in some cases strains) may be projected to the nodes using the command:

```
stre node n1 n2 n3
```

   where `n1` to `n2` defines a nodal range to be output at increments of `n3`. Table D.3 summarizes the meaning of the values for the different element types.

5. The average of stress and strain over each material may be computed using the command

```
stre aver
```

6. The command `STRE,ON` activates output of stresses at quadrature points for some solid elements. The command `STREss,OFF` results in the output of *average* stresses over each individual element.

With the `ERROr` option `STREss` computes element sizes for adaptive mesh refinement. N.B. The error option does not function with all elements.

Element	Values	Description
Solid	1 to 6	Cartesian Stresses
	7 to 12	Cartesian Strains
	13 to 14	Fiber stress (3d only)
Truss	1	Axial force
Frame(2d)	1 to 3	Axial, Shear, Moment resultant
Frame(3d)	1 to 6	Axial, 2 Shears, Torque, 2 Moments
Plate	1 to 6	1=N_x, 2=N_y, 4=N_{xy}, 5=S_{xz}, 6=S_{yz}
	7 to 10	7=κ_x, 8=κ_y, 10=κ_{xy},11=γ_{xz}, 12=γ_{yz}
Membrane	1 to 6	1=N_{11}, 2=N_{22}, 4=N_{12}, 5=N_1, 6=N_2
	7 to 12	7=ϵ_{11}, 8=ϵ_{22}, 10=ϵ_{12}, 11=ϵ_1, 12=ϵ_2
Shell (2d)	1 to 5	1=N_{11}, 2=N_{22}, 3=S_{13}, 4=M_{11}, 5=M_{22}
	7 to 11	1=ϵ_{11}, 1=ϵ_{22}, 9=γ, 10=κ_{11}, 11=κ_{22}
Shell (3d)	1 to 6	1=N_{11}, 2=N_{22}, 4=N_{12}, 5=Q_{13}, 6=Q_{23}
	7 to 10	7=M_{11}, 8=M_{22}, 10=M_{12}
Thermal	13 to 15	13=q_x, 14=q_y, 15=q_z
Torsion	1 to 5	1=τ_x, 2=τ_y, 3=$\lvert\tau\rvert$, 4=Warp fn., 5=Stress fn.
	6 & 11	6=Yield fn., 11=Pl. strain

Table D.3: Stress projection values

```
subs,<prin,n1,n2,stol>
```

The SUBSpace command requests the solution for n1 eigenpairs of a problem about the current state. An additional n2 vectors are used to expand the subspace and improve convergence (by default, n2 is set to the minimum of n1 plus 8 or 2 times n1 or the maximum number of eigenvalues in the problem). The SUBSpace command must be preceded by the specification of the tangent stiffness array using a TANGent command, and a mass array (either a lumped mass by MASS,LUMP or a consistent mass by MASS). Note that the smallest n1 eigenvalues and eigenvectors are computed with reference to the current shift specified on the TANGent command. If n2 is larger than the number of non-zero mass diagonals it is truncated to the actual number that exist. Whenever n1 is close to the number of non-zero mass diagonals one should compute the entire set since convergence will be attained in one iteration (this applies primarily to small problems).

Use of the PRINt option produces an output of all subspace matrices in addition to the estimates on the reciprocals of the *shifted* eigenvalues. For large problems considerable output results from a use of this option, and thus it is recommended for small problems only.

All eigenvalues are computed until two subsequent iterations produce values which are accurate to stol, (default stol = max(tol, 1.d-12)).

```
tang,,<n1,v2>
tang,line,<n1,v2,v3>
tang,eigv,,n1
```

The TANGent command computes a symmetric tangent stiffness matrix about the current value of the solution state vector. For linear applications the current stiffness matrix is just the normal *stiffness* matrix.

If the value of n1 is non-zero, a force vector for the current residual is also computed (this is identical to the FORM command computation) - thus leading to greater efficiency when both the tangent stiffness and a residual force vector are needed. The resulting equations are also solved for the solution increment. Thus,

```
TANGent,,1
```

is equivalent to the set of commands

```
TANGent
FORM
SOLVe
```

If the value of v2 is non-zero a *shift* is applied to the stiffness matrix in which the element mass matrix is multiplied by v2 and subtracted from the stiffness matrix. This option may be used with the SUBSpace command to compute the closest eigenvalues to the shift, v2. Alternatively, the shift may be used to represent a forced vibration solution in which all loads are assumed to be harmonic at a value of the square-root of v2 (rad/time-unit).

After the tangent matrix is computed, a triangular decomposition is available for subsequent solutions using FORM, SOLVe, BFGS, etc.

In the solution of non-linear problems, using a full or modified Newton method, convergence from any starting point is not guaranteed. Two options exist within available commands to improve chances for convergence. One is to use a line search to prevent solutions from diverging rapidly. Specification of the command TANGent,LINE plus options invokes the line search (it may also be used in conjunction with SOLVe,LINE in modified Newton schemes). The parameter v3 is typically chosen between 0.5 and 0.8 (default is 0.8).

The second option to improve convergence of non-linear problems is to reduce the size of the load step increments. The command `BACK` may be used to *back-up* to the beginning of the last time step (all data in the solution vectors is reset and the history data base for inelastic elements is restored to the initial state when the current time is started). Repeated use of the back command may be used. However, it applies only to the current time interval. The loads may then be adjusted and a new solution with smaller step sizes started.

The `EIGV`alue option is used in transient algorithms to compute eigenvalues of the (static) stiffness matrix. If `IDEN`tity has been issued, then the shift given by non-zero `n1` is with respect to the identity otherwise the element mass matrix is used. Note, `SUBS`pace is used to compute the actual eigen-pairs.

The `TANG`ent operation is normally the most time consuming step in problem solutions - for large problems several seconds are required - be patient!

```
tie off
```

Meshes which have been merged using a `TIE` command may be disconnected during solution using the `TIE OFF` solution command. For example

```
BATCh
  TIE OFF
END
```

would disconnect all parts. Subsequently, if desired, parts may be reconnected using different tie options. For example

```
BATCh
  TIE OFF
END
TIE REGIon 1 2
```

would disconnect all parts and then reconnect `region 1` to `region 2`.

```
time,,<t_max>
time,<set,t>
```

The use of the `TIME` command will increment the current time by `DT`, the current time increment. In addition, a new value of the proportional loading will be computed, if necessary. The value of the current time and proportional loading are reported in the output (or to the screen). The time command also will perform the first update for an active time integration algorithm of the equations of motion (e.g., the Newmark-beta method) , as well as, update the history data base for any elements with non-linear constitutive equations (e.g., those which require variables other than the displacement state to compute a solution). Accordingly, it is imperative to include a time command for this class of problems. Example: Time dependent solution with loop control

```
DT,,1.
LOOP,,10
  TIME
   ..
  etc.
   ..
NEXT
```

Performs 10 time steps of a solution.

As an option, it is possible to specify the maximum time that integration is to be performed. Accordingly, when a variable time step is employed the `TMAX` parameter value may be used as a convenient stop marker. This also is essential if an automatic time stepping algorithm is implemented. Example: Time dependent solution with loop control, terminate at specified time.

```
DT,,1.
LOOP,,10
  TIME,,5.0
   ..
  etc.
   ..
NEXT
```

Performs 10 time steps of a solution; however, if the time reaches the value of 5.0 before the 10 steps terminate the execution. This may happen if the `DT` value is automatically adjusted by another step in the solution process.

The current time may be set to a specified value, `T`, using the command `TIME,SET,T` (where `T` is the value desired). No other action is taken. This may be helpful in certain steady state problems where solutions are desired for certain specified times.

```
tol,,v1 v2
tol,ener,v1
tol,emax,v1
tol,iter,v1,v2
```

The `TOL` command is used to specify the solution tolerance values to be used at various stages in the analysis. Uses include:

1. Convergence of nonlinear problems in terms of the norm of energy in the current iterate (the inner, dot, product of the displacement increment and the solution residual vectors).
2. Convergence of iterative solution of linear equations.
3. Convergence of the subspace eigenpair solution which is measured in terms of the change in subsequent eigenvalues computed.

The basic command, `TOL,,tole tolr`, without any arguments sets the parameter *tol* used in the solution of non-linear problems where the command sequence

```
LOOP,,30
  TANG,,1
NEXT
```

is given. In this case, the loop is terminated either when the number of iterations reaches 30 (or whatever number is given in this position) or when the *energy error* is less than *tole*. The energy error is given by

$$E_i = (d\mathbf{u}^T\mathbf{R})_i \leq tol\,(d\mathbf{u}^T\mathbf{R})_0 = E_0$$

in which $\mathbf{R}$ is the residual of the equatons and $d\mathbf{u}$ is the solution increment. The default value of *tol* for the solution of nonlinear problems is 1.0d-16. A secondary check is made on the residual norm using `tolr` which by default is set at the value

$$tolr = 100\,(tole)^{1/2}$$

The `TOL` command also permits setting a value for the energy below which convergence is assumed to occur. The command is issued as `TOL,ENERgy,v1` where `v1` is the value of

the converged energy (i.e., it is equivalent to the tolerance times the maximum energy value). Normally, *FEAP* performs nonlinear iterations until the value of the energy is less than the TOLerance value times the value of the energy from the first iteration as shown above. However, for some transient problems the value of the initial energy is approaching zero (e.g., for highly damped solutions which are converging to some steady state limit). In this case, it is useful to specify the energy for convergence relative to early time steps in the solution. Convergence will be assumed if either the normal convergence criteria or the one relative to the specified maximum energy is satisfied.

The TOL command also permits setting the maximum energy value used for convergence. The command is issued as

```
TOL,EMAX}imum,v1
```

where v1 is the value of the maximum energy quantity. Note that the TIME command sets the maximum energy to zero, thus, the value of EMAXimum must be reset after each time step using, for example, a set of commands:

```
LOOP,time,n
  TIME
  TOL,EMAX,5.e+3
  LOOP,newton,m
    TANG,,1
  NEXT
  etc.
NEXT
```

to force convergence check against a specified maximum energy. The above two forms for setting the convergence are nearly equivalent; however, the ENERgy tolerance form can be set once whereas the EMAXimum form must be reset after each time command.

The command

```
TOL,ITERation,rtol,atol
```

is used to control the solution accuracy when an *iterative* solution process is used to solve the equations

$$\mathbf{K}\, d\mathbf{u} = \mathbf{R}$$

In this case the parameter *rtol* sets the relative error for the solution accuracy, i.e., when

$$(\mathbf{R}^T\mathbf{R})_i^{1/2} \leq rtol\, (\mathbf{R}^T\mathbf{R})_0^{1/2}$$

The parameter *atol* is only used when solutions are performed using the `KSP` schemes in a `PETSc` implementation to control the absolute residual error

$$(\mathbf{R}^T\mathbf{R})_i^{1/2} \leq atol$$

The default for `rtol` is `1.0d-08` and that for `atol` is `1.0d-16`.

```
tplo,,inc
   < After end record give the data >
 disp,node,dof,x,y,z
 velo,node,dof,x,y,z
 acce,node,dof,x,y,z
 reac,node,dof,x,y,z
 cont,node,dof,x,y,z
 arcl,node,dof
 stre,elmt,comp,x,y,z
 stra,elmt,comp,x,y,z
 flux,elmt,comp,x,y,z
 hist,elmt,comp,x,y,z
 rsum,comp,node1,node2
 sums,dof,dir,x,xtol
 ener
 show
```

The `TPLOt` command can be used to specify components of displacement, velocity, acceleration, reaction, contact node, arclength parameter, stress, and energy which are to be saved to construct time history plots as a post processing operation. The command may be issued several times; however, the total number of components to be saved for each type of plot (time vs. displacement or time vs. reaction, etc.) is limited to 200. The `inc` option is used to specify the number of time steps between output of information. Additional items may be added to the `TPLOt` list by inserting the command more than one time. For example:

```
BATCh
  TPLOt
END
DISP,5,1
  ... additional components

BATCh or INTEractive
  ... solve steps
END  ! for BATCh only
```

```
BATCh
  TPLOt
END
DISP,5,2
  ... additional components

BATCh or INTEractive
  ... solve steps
END  ! for BATCh only
```

In the above, the displacement component 1 for node 5 would be included in the `tplot` file during the next solution steps (a new line is added for each `TIME` command processed during solution). After these steps a new entry for displacement component 2 for node 5 is added and will appear in the output file (in addition to the other components).

The option `SHOW` is used to echo the current list to the screen during interactive executions.

Options which include both `node` or `x,y,z` may be used. Giving the command as:

```
xxxx,node,dof
```

requires specific numbers to be provided for the `node` and `dof` parameters. The value of `node` must be an *active* global node number of the mesh (i.e., one which has not been deleted by at `TIE` command). Alternatively, the command may be given as:

```
xxxx,,dof,x,y,z
```

where `x,y,z` are values for the necessary number of coordinates (ndm). A search will be made to locate the node which is *closest* to the coordinates given.

The `DISP`lacement option will save the node and degree of freedom value, together with the time in a file `Pxxxy.dis`, where `xxx` is the name assigned for the input data file (with the I stripped) and `y` ranges between `a` and `j`. The components are on one record in the order given during the tplot inputs. Similarly for other node based quantities.

The `ENER`gy option maybe used to accumulate total linear/angular momentum and kinetic/potential energy.

The `ARCL`ength option output the arc-length load level versus the selected nodal displacement dof.

The `STRE`ss option will save the element and component value, together with the time in a file `Pxxxy.str`, where `xxx` is the name assigned for the input data file (with the I stripped) and `y` ranges between `a` and `j`. The components are on one record in the

order given during the tplot inputs. The meaning of components is element dependent, however, for solid elements the stress values are repeated at a stride of 6 for the different quadrature points of the the element. Thus, component 1 would be σ_{11} for the first Gauss point, compontent 7 would be the σ_{11} for the second Gauss point, etc. for the other points.

THe `STRAin` and `FLUX` options are identical to the behavior of stresses. For thermal problems the first 3 components of `FLUX` are the thermal flux and the next 3 the thermal gradient.

An example for the use of tplot is:

```
BATCh
  TPLOt
END
stre,3,24
stre,25,24
stre,25,26
disp,11,2
disp,,2,5.2,4.3,-1.2
show
           ! blank termination record
```

requests stress output for component 24 in element 3 and components 24 and 26 from element 25. The program will also output nodal displacement as requested by `disp` for dof 2 at node 11 and at the node located at the coordinates closest to (5.2, 4.3, -1.2). Finally, the list will be echoed by the `show` command.

The `HISTory` option will save the element and component value, together with the time in a file `Pxxxy.str`, where `xxx` is the name assigned for the input data file (with the I stripped) and `y` ranges between `a` and `j`. The components are on one record in the order given during the tplot inputs. The meaning of components is as defined on the material record.

The `SUMS` option accumulates the total reaction force for degree of freedom `dof` in the mesh coordinate direction `dir` for all nodes with coordinate value `x` within a tolerance `xtol`. For example the command set

```
BATCh
  TPLOt
END
sums,2,1,5,0.001
show
           ! blank termination record
```

would accumulate the total reaction in direction 2 for the coordinate x_1 (1-direction) with value 5 ± 0.001 units.

```
tran,name,<v1,v2,v3>
tran,off
```

The use of the command `TRAN`sient indicates that a transient solution is to be computed. Several options are implemented:

1. A generalized mid-point method for static problems.
2. An Euler-backward difference implicit method for first order ordinary differential equations such as heat transfer, etc.
3. An Euler-forward difference explicit method for first order ordinary differential equations such as heat transfer, etc.
4. A backward difference second order formula (BDF2) for for first order systems.
5. A generalized mid-point method for first order systems.
6. The Newmark-beta step-by-step integration of the equations of motion.
7. Hilber-Hughes-Taylor alpha method for second order systems.
8. An Euler-backward difference method for second order equations.
9. An energy conserving generalized mid-point method for second order systems.
10. An explicit implementation of Newmark.
11. A central difference explicit method for second order systems.

The `OFF` option for `name` turns off any active time integration algorithm returning *FEAP* to its default quasi-static solution mode.

The method used depends on the specified `NAME` in the command.

1. Static Alpha Method (`name` is `stat`) has the input form

   ```
   TRANsient STATic alpha
   ```

   The method requires the specification of the alpha parameter for the momentum equation

   $$\boldsymbol{P}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha})$$

   where $\boldsymbol{P}$ is the nonlinear internal force term.

v1 = alpha (default = 0.5)
Alpha should be between 0.5 and 1.

2. Backward Euler (`name` is `back`) with input record given as

```
TRANsient BACKard
```

The backward Euler method requires no parameters for `v1`, etc., and may be used to solve any first order ode set. In this method only one rate vector exists, namely the rate of the solution vector.

3. Forward Euler (`name` is `forw`) with input record given as

```
TRANsient FORWard
```

The forward Euler method requires no parameters for `v1`, etc., and may be used to solve any first order ode set. The method is conditionally stable and a critical time step size should be set. In this method only one rate vector exists, namely the rate of the solution vector.

4. Alpha Method for First Order Systems(`name` is `gen1`) has the input form

```
TRANsient GEN1 alpha
```

The method requires the specification of the alpha parameter for the equation,

$$\boldsymbol{M}\boldsymbol{v}(t_{n+\alpha}) + \boldsymbol{P}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha})$$

where $\boldsymbol{P}$ is the nonlinear internal force term.

v1 = alpha (default = 0.5)
Alpha should be between 0.5 and 1.

5. The bacward difference formula BDF2 is used for first order systems and has the input form

```
TRANsient BDF2 alpha
```

The method requires the specification of the alpha parameter for the equation,

$$\boldsymbol{M}\boldsymbol{v}(t_{n+\alpha}) + \boldsymbol{P}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha})$$

where $\boldsymbol{P}$ is the nonlinear internal force term.

v1 = alpha (default = 0.5)
Alpha should be between 0.5 and 1.

6. Newmark Method (`name` is `newm` or blank) with input record given as

```
TRANsient NEWMark beta gamma
```

The values of the Newmark parameters are specified as follows:

v1	= beta -	the Newmark parameter which primarily controls stability (default is 0.25).
v2	= gamma -	the Newmark parameter which primarily controls numerical damping (default is 0.50) Note: gamma must be greater than or equal to 0.50.

This option does not permit an *explicit* solution using beta = 0.0, only implicit solutions are considered. Accordingly, it is recommended that values of beta be set to 0.25 (the default value) unless there is a compelling reason not to use this value. With gamma set to 0.50 and beta set to 0.25 the method becomes the "average" acceleration or trapezoidal method.

7. HHT Alpha Method (`name` is `alph` or `hht`) With input record given as

```
TRANsient ALPHa beta gamma alpha
```

the `alph` form of the HHT method requires the specification of three parameters. The first two are identical to the Newmark beta and gamma parameters, the third is the HHT alpha parameter (definition is different than original paper where $\alpha = 1 + \alpha_H$ with $1/3 < \alpha_H < 0$) in momentum equation:

$$\boldsymbol{M}\,\boldsymbol{a}(t_{n+1}) + \boldsymbol{P}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha})$$

where $\boldsymbol{P}$ is the nonlinear internal force term.

v1 =	beta (default = 0.25)
v2 =	gamma (default = 0.5)
v3 =	alpha (default = 0.5) Alpha should be between 0.5 and 1. N.B. 1 = Newmark.

For the HHT ALpha Method with `name` given as `hht` the input form is

```
TRANSient HHT alpha
```

and the other parameters are computed from

$$\beta = \tfrac{1}{4}\,(2-\alpha)^2 \quad \text{and} \quad \gamma = \tfrac{3}{2} - \alpha$$

8. The Euler-backward difference formula for second order systems has the input form

   ```
   TRANsient EULEr
   ```

   The method has no input parameters

   $$\boldsymbol{M}\boldsymbol{a}(t_{n+1}) + \boldsymbol{P}(\boldsymbol{x}(t_{n+1})) = \boldsymbol{F}(t_{n+1})$$

   where $\boldsymbol{P}$ is the nonlinear internal force term.

9. Conserving Alpha Method (`name` is `cons`) has the input form

   ```
   TRANsient CONServing beta gamma alpha
   ```

   The conserving method requires the specification of three parameters. The first two parameters are associated with the update formulas, the third with the momentum equation,

   $$\frac{1}{\Delta t}\,\boldsymbol{M}(\boldsymbol{v}(t_{n+1}) - \boldsymbol{v}(t_n)) + \boldsymbol{P}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha})$$

   where $\boldsymbol{P}$ is the nonlinear internal force term.

   v1 = beta (default = 0.5)
   v2 = gamma (default = 1.0)
   v3 = alpha (default = 0.5)
   Alpha should be between 0.5 and 1.

10. Explicit Newmark Method (`name` is `expl`) has the input form

    ```
    TRANSient EXPLicit gamma
    ```

    This option permits the explicit form of the Newmark method to be implemented. The input parameter is only

    v1 or v2 = gamma (default = 0.5)

11. The central difference explicit method has the input form

    ```
    TRANsient CENTral
    ```

and no parameters are possible. The algorithm is computed from the following:

$$\begin{aligned} \boldsymbol{v}_{n+1/2} &= \frac{1}{\Delta t}\left(\boldsymbol{u}_{n+1} - \boldsymbol{u}_n\right) \\ \boldsymbol{a}_n &= \frac{1}{\bar{\Delta} t}\left(\boldsymbol{v}_{n+1/2} - \boldsymbol{v}_{n-1/2}\right) \\ \boldsymbol{M}\,\boldsymbol{a}_{n+1} &= \boldsymbol{F}_{n+1} - \boldsymbol{P}(\boldsymbol{u}_{n+1}, \boldsymbol{v}_{n+1/2}) \end{aligned}$$

where $\boldsymbol{P}$ is the nonlinear internal force, $\bar{\Delta} t = (\Delta t_{n-1} + \Delta t)/2$ and $\Delta t = t_{n+1} - t_n$.

It is possible to specify non-zero values for the initial velocity in second order system integrators using the command `INITial` (for initial values). If the initial state is not in equilibrium an initial acceleration may be obtained by using a `FORM,ACCE` command before initiating any transient state. It is necessary for the parameters to first be set using a `TRAN`sient command. It is also possible to compute self equilibrating static states with non-zero displacements and then switch to a dynamic solution. Alternatively, a restart mode (`REST`art) may be used to start from a previously computed non-zero state.

```
utan,,<n1,v2>
utan,line,<n1,v2,v3>
```

The UTANgent command computes an unsymmetric tangent stiffness matrix about the current value of the solution state vector. For linear applications the current stiffness matrix is just the normal stiffness matrix.

If the value of n1 is non-zero, a force vector for the current residual is also computed (this is identical to the FORM command computation) - thus leading to greater efficiency when both the tangent stiffness and a residual force vector are needed.

If the value of v2 is non-zero a *shift* is applied to the stiffness matrix in which the element mass matrix is multiplied by v2 and subtracted from the stiffness matrix. This option may not be used with the SUBSpace algorithm, which is restricted to symmetric tangents only (see TANGent command). The shift may be used to represent a forced vibration solution in which all loads are assumed to be harmonic at a value of the square-root of v2 (rad/time-unit).

After the tangent matrix is computed, a triangular decomposition is available for subsequent solutions using FORM and SOLVe, etc.

In the solution of non-linear problems, using a full or modified Newton method, convergence from any starting point is not guaranteed. Two options exist within available commands to improve chances for convergence. One is to use a line search to prevent solutions from diverging rapidly. Specification of the command UTAN,LINE plus options invokes the line search option (it may also be used in conjunction with SOLVe,LINE in modified Newton schemes). The parameter v3 is typically chosen between 0.5 and 0.8 (default is 0.8).

The second option to improve convergence of non-linear problems is to reduce the size of the load step increments. The command BACK may be used to *back-up* to the beginning of the last time step (all data in the solution vectors is reset and the history data base for inelastic elements is restored to the initial state when the current time is started). Repeated use of the BACK command may be used. However, it applies only to the current time interval. The loads may then be adjusted and a new solution with smaller step sizes started.

The UTANgent operation is normally the most time consuming step in problem solutions - for large problems several seconds are required - be patient!

```
velo,,<n1,n2,n3>
velo,all
velo,coor,dir,xi,tol
velo,list,n1
velo,node,x1,x2,x3
velo,cmpl,<n1,n2,n3>
velo,imag,<n1,n2,n3>
```

The command `VELO`city may be used to print the current values of the velocity vector as follows:

1. Using the command:

   ```
   velo,,n1,n2,n3
   ```

   prints out the current velocity vector for nodes `n1` to `n2` at increments of `n3` (default increment = 1). If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only the first nodal velocity is reported.

2. If the command is specified as:

   ```
   velo,all
   ```

   all nodal velocities are output.

3. If the command is specified as:

   ```
   velo,coor,dir,xi,tol
   ```

   all nodal quantities for the coordinate direction `dir` with value equal to `xi` (within the tolerance `tol`) are output. The default for `tol` is 0.01 coordinate units.

   Example:

   ```
   velo,coor,1,3.5
   ```

   prints all the nodal velocity vector components which have $x_1 = 3.5 \pm 0.01$i units.

   This is useful to find the nodal values along a particular constant coordinate line.

4. If the command is specified as:

   ```
   velo,list,n1
   ```

   all nodal quantities contained in `list` number `n1` are output (see command `LIST` for specification of the list).

   Example:

   ```
   velo,list,3
   ```

   prints the nodal velocities contained in list number 3.

5. If the command is specified as:

   ```
   velo,node,x1,x2,x3
   ```

   the single value for the velocity *nearest* the coordinate with values `x1, x2, x3` is output. Only coordinates up to the dimension of the mesh need be specified.

6. If the command is specified as:

   ```
   velo,cmpl,n1,n2,n3
   ```

   the current *real and imaginary* part of a complex velocity vector for nodes `n1` to `n2` at increments of `n3` (default increment = 1) is output. If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only node one (1) is reported.

7. If the command is specified as:

   ```
   velo,imag,n1,n2,n3
   ```

   only the current *imaginary* part of a complex velocity vector for nodes `n1` to `n2` at increments of `n3` (default increment = 1) is output. If `n2` is not specified only the value of node `n1` is output. If both `n1` and `n2` are not specified only node one (1) is reported.

In order to output a velocity vector it is first necessary to specify commands language instructions to compute the desired values for a transient analysis.

WRITe FEAP SOLUTION COMMAND MANUAL

```
writ,xxxx
```

The WRITe command may be used to save the current values of displacements and nodal stresses for subsequent use. This option is particularly useful for saving states which are to be plotted later. It is not intended as a restart option (see RESTart for restarting a previously saved problem state).

The values of xxxx are used to specify the file name (4-characters only), manipulate the file, and write out states. The values permitted are:

xxxx =	wind	Rewind current output file.
xxxx =	clos	Close current output file.
xxxx =	disp	Write current displacement state onto the current file.
xxxx =	stre	Write current nodal stress state onto the current file.
xxxx =	????	Anything else is used to set current filename. Only four characters are permitted and only one file may be opened at any time. Files may be opened and closed several times during any run to permit use of more than one file name.

A WRITe output is reinput using the READ command which has nearly identical options for xxxx.

```
zero
zero,regi,k1
zero,node
zero,hist
```

This command zeros the nodal and history variables when used without any options. With the `node` option only the nodal quantities are zeroed and with the `hist`ory option only the history variables are zeroed. With the `regi`on option it zeros the displacements associated with region `k1` if the nodes are not part of another region; the history variables are not affected.

```
zzhu,,ma
zzhu,off
```

The use of the `ZZHU` command specifies the Zienkiewicz-Zhu algorithm is to be used to construct the projections of element quantities. Use of `ZZHU,OFF` disables this projection and *FEAP* then uses a *lumped* projection scheme. Caution should be exercised in using the `ZZHU` form as all elements are not coded yet.

If a non-zero value is specified for the `MA` parameter the projection is performed for material set `MA` only. Use of a zero value projects all material sets.

Appendix E

Plot manual pages

FEAP has several options which may be used to display results on a graphics screen or to prepare PostScript files for later printing in documents. The following pages summarize the commands which are available to plot specific results. Commands exist to plot results for one to three dimensional problems. Three dimensional results are best displayed using a perspective view and hidden surface removal methods. Very simple schemes are used and anomalies can exist due to the order in which surface facets are sorted. Results can also be saved and displayed using other display tools.

ACCEleration FEAP PLOT COMMAND MANUAL

```
acce,n1,n2,n3
```

Plot contours for acceleration degree of freedom `n1` (default is 1). Two options are available to construct contours:

1. If `n2` is zero or negative, areas between contours will be shaded in colors or gray scale. For this case, only the minimum and maximum contour values are specified - by default (enter return) the program constructs 12 evenly spaced intervals for the shading.

   The default is to set the range based on the maximum and minimum values for the contours plotted. The range displayed may be reset using the plot `RANGe` command.

   If `n3` is positive, plotting of the mesh is suppressed. If `n3` is negative, plotting of the mesh is suppressed and the previously range of contour values are used. Note: The contours must have been already set by a previous call to `ACCE`leration for this option to function properly.

2. If `n2` is a positive number specific contour lines may be designated and plotted as lines. It is necessary to define the value for each of the `n2` contour lines. If `n3` is non-zero a numerical label will be added near each contour indicating the relationship to a value table given on the screen.

If the default settings are turned off using the plot `DEFAult OFF` command then in interactive mode, after an `ACCE,n1,n2,n3` command is given, prompts for additional data will appear. For each contour line the values to be plotted should be entered (maximum of 8 items per record). Maximum and minimum existing values are indicated on the screen. For shaded plots only a lower and an upper value separating the smallest and largest shading from their adjacent ones are input.

Note that default settings may be restored using the `DEFAult ON` command.

`aplo,<n1,n2>`

For transient solutions of second order equations, the `APLOt` command may be used (in interactive mode only) to specify any line for a plot of an `n1` acceleration component. After entering the command the LEFT mouse button is used to select the ends of a line through or in the mesh which defines the location for the acceleration plot.

After entering the two ends for the line (labeled A and B), an X-Y plot for the `n1` acceleration component is superposed on the screen. The X-axis of the plot corresponds to the selected A-B mesh line. The Y-axis of the plot displays the magnitude of projected acceleration component along the line. The magnitude of acceleration plotted is proportional to the largest and smallest values which occur anywhere in the mesh. A contour plot of the acceleration component may be used to identify locations for the maximum and minimum (use the `CONTour` command). If `n2` is non-zero it is used as the plot number (up to 12 plots may be placed on the same figure). If `n2` is zero, the previous plot number is incremented and assigned as the current plot number.

Currently, this command works only for 2-D problems displayed in a `CARTesian` mode.

AWIRe FEAP PLOT COMMAND MANUAL

```
awir,n1,n2,n3
```

Plot contours for acceleration degree of freedom `n1` in a wire-frame mode (default is `n1 = 1`). The mesh is displayed as in a wire-frame mode with contour values of acceleration added to each edge. Two options are available to construct contours:

1. If `n2` is zero or negative, areas between contours will be shaded in colors or gray scale. For this case, only the minimum and maximum contour values are specified - by default (enter return) the program constructs 12 evenly spaced intervals for the shading.

   The default is to set the range based on the maximum and minimum values for the contours plotted. The range displayed may be reset using the plot `RANGe` command.

   If `n3` is positive, plotting of the mesh is suppressed. If `n3` is negative, plotting of the mesh is suppressed and the previously range of contour values are used. Note: The contours must have been already set by a previous call to `AWIRe` for this option to function properly.

2. If `n2` is a positive number specific contour lines may be designated and plotted as lines. It is necessary to define the value for each of the `n2` contour lines. If `n3` is non-zero a numerical label will be added near each contour indicating the relationship to a value table given on the screen.

If the default settings are turned off using the plot `DEFAult OFF` command then in interactive mode, after an `AWIR,n1,n2,n3` command is given, prompts for additional data will appear. For each contour line the values to be plotted should be entered (maximum of 8 items per record). Maximum and minimum existing values are indicated on the screen. For shaded plots only a lower and an upper value separating the smallest and largest shading from their adjacent ones are input.

Note that default settings may be restored using the `DEFAult ON` command.

```
axis,v1,v2
```

A set of axes defining the coordinate directions will be plotted with the origin of the axes placed at coordinates x= `v1`, y= `v2`. The x, y coordinates are specified relative to the origin of the problem dimensions.

BACKground FEAP PLOT COMMAND MANUAL

```
back,v1
```

If `v1` is zero (or unspecified), change plot background color for PostScript files to black. If `v1` is non-zero change plot background to white. Used to make color slides with either black or white backgrounds.

```
bord,n1 number
bord,on number
bord,off number
```

The `BORD`er command permits the plot border to be displayed in color `n1` or to be turned `off` or `on` while in interactive mode.

BOUNdary conditions FEAP PLOT COMMAND MANUAL

```
boun,n1
```

The `BOUN`dary condition command may be used to display all active restraints, or those in a particular directions (only first three are displayed). If `n1` is zero all restraints are shown, otherwise only those for the `n1` degree of freedom are shown.

```
bplot
```

The `BPLOt` command may be used with beams whose cross sections are defined using the `SECTion` options in `MATErial` data inputs. The cross section is projected normal to the beam axis and surface plots for the axial stress is superposed on the surface. This command works with three-dimensional beam elements only.

CAPTion FEAP PLOT COMMAND MANUAL

```
capt,text
```

This command specifies the label to be assigned to the next contour plot. The string `label` replaces the default parameter (e.g., `DISPLACEMENT 1` from `CONT,1`, etc.). Only one plot will use the caption, with the default being restored for any subsequent plots.

CARTesian FEAP PLOT COMMAND MANUAL

```
cart
```

All plots are to be drawn in a `CART`esian frame. This is the default view for plots. A plot may also be in a perspective view (see `PERS`pective plot manual page).

```
cent,x,y
```

The `CENTer` command is used to place the center at a specific location on the screen. The input values of `x` and `y` locate the center of the plot in terms of normalized screen coordinates. The plot region covers approximately the area bounded by $0 < x < 1.4$ and $0 < y < 1.0$.

```
clea
```

Wipe the center of the plot area leaving the border, logo, and legend area untouched. Some graphics terminals do not support the feature of erasing only part of the screen; in these cases the entire screen may be erased instead of only the part specified.

```
clip,n1,v1,v2
```

The clip feature permits the user to plot part of the mesh and/or results. The `n1` specifies the coordinate direction ($1 = x_1$, etc.) for the clipping, `v1` and `v2` are the values of the coordinate which define the range of the plot coordinate to display. Clipping is performed by requiring the entire element to be within the clip boundaries. The command may be given more than once to clip in different coordinate directions.

If the command is given without parameters the entire mesh region is selected for the plot. Thus, issuing `CLIP` alone cancels any previous clip definitions.

This command is primarily intended for three dimensional objects which contain internal voids which are not visible in perspective views. By clipping it is possible to remove elements which block the internal void.

```
colo,n1,n2
```

Sets PostScript outputs to color or gray scale. If `n1` < 0, gray scale plots are produced (by default PostScript plots are in gray scale). If `n1` $>$ or $= 0$ color PostScript plots are enabled. The `n2` parameter permits reversing the color order: `n2` $= 0$ is called normal order, `n2` non-zero is reversed order.

```
cont,n1,n2,n3
```

Plot contours for solution degree of freedom `n1` (default is 1). Two options are available to construct contours:

1. If `n2` is zero or negative, areas between contours will be shaded in colors or gray scale. For this case, only the minimum and maximum contour values are specified - by default (enter return) the program constructs 12 evenly spaced intervals for the shading.

   The default is to set the range based on the maximum and minimum values for the contours plotted. The range displayed may be reset using the plot `RANGe` command.

   If `n3` is positive, plotting of the mesh is suppressed. If `n3` is negative, plotting of the mesh is suppressed and the previously existing contour values are used. Note the that the contours must have been already set by a previous call to `CONTour` for this option to function properly.

2. If `n2` is a positive number specific contour lines may be designated and plotted as lines. It is necessary to define the value for each of the `n2` contour lines. If `n3` is non-zero a numerical label will be added near each contour indicating the relationship to a value table given on the screen.

If the default settings are turned off using the plot `DEFAult OFF` command then in interactive mode, after the `CONT,n1,n2,n3` command is given prompts for additional data will appear. For each contour line the values to be plotted should be entered (maximum of 8 items per record). Maximum and minimum existing values are indicated on the screen. For shaded plots only a lower and an upper value separating the smallest and largest shading from their adjacent ones are input.

Note that default settings may be restored using the `DEFAult ON` command.

```
cwir,n1,n2,n3
```

Plot contours for solution degree of freedom `n1` in a wire-frame mode (default is `n1 = 1`). The mesh is displayed as in a wire-frame mode with contour values added to each edge. Two options are available to construct contours:

1. If `n2` is zero or negative, areas between contours will be shaded in colors or gray scale. For this case, only the minimum and maximum contour values are specified - by default (enter return) the program constructs 12 evenly spaced intervals for the shading.

   The default is to set the range based on the maximum and minimum values for the contours plotted. The range displayed may be reset using the plot `RANGe` command.

   If `n3` is positive, plotting of the mesh is suppressed. If `n3` is negative, plotting of the mesh is suppressed and the previously existing contour values are used. Note the that the contours must have been already set by a previous call to `CWIRe` for this option to function properly.

2. If `n2` is a positive number specific contour lines may be designated and plotted as lines. It is necessary to define the value for each of the `n2` contour lines. If `n3` is non-zero a numerical label will be added near each contour indicating the relationship to a value table given on the screen.

If the default settings are turned off using the plot `DEFAult OFF` command then in interactive mode, after the `CWIRe,n1,n2,n3` command is given prompts for additional data will appear. For each contour line the values to be plotted should be entered (maximum of 8 items per record). Maximum and minimum existing values are indicated on the screen. For shaded plots only a lower and an upper value separating the smallest and largest shading from their adjacent ones are input.

Note that default settings may be restored using the `DEFAult ON` command.

DEFAult FEAP PLOT COMMAND MANUAL

```
defa,on
defa,off
```

Normally, *FEAP* will issue prompts for parameters needed to construct plots. Usually, default values may be accepted by pressing the return (or enter) key. The `DEFA`ult command may be used to eliminate the need to press the key to accept the default values. The command has one parameter which is either `ON` or `OFF`. Omitting the parameter turns off the prompts.

DEFOrmed FEAP PLOT COMMAND MANUAL

```
defo,v1,n2,v2
```

This command sets the plot options to be associated with a deformed mesh with the displacements scaled by the `v1` value (default: v1 = 1). If any part of an element in the deformed mesh leaves the plot region, it will not appear in the plot.

Specification of a nonzero `n2` value retains the plot scaling at a previously set `v1` value. This permits superposition of undeformed or previous solutions on the current plot for comparison purposes.

The parameter `v2` is used to scale eigenvectors. By default all eigenvectors are unscaled. The program attempts to scale eigenvectors for plotting such that the maximum component is unity. Thus, the scaling should compare the mesh dimensions such that a reasonable shape is obtained.

An undeformed option is specified using the `UNDE`formed command.

DISPlacements FEAP PLOT COMMAND MANUAL

```
disp,<tip,length>
```

Plot nodal generalized displacements as vectors at each node. If `tip` is nonzero the vector tip will appear next to the node; whereas, if `tip` is zero the tail of the load vectors are on the nodes. Default: `tip = 0`. Vector lengths will be scaled in proportion to the maximum displacement, accordingly some vectors may be too small to be visible. If the parameter `length` is positive it will be used to scale the displacements and, thus, can assist in visualizing both large and small values.

DOFS FEAP PLOT COMMAND MANUAL

```
dofs,n1,n2,n3
```

This command allows one to reorder or turn off the degree of freedoms for plotting purposes. `n1` becomes the first degree of freedom, `n2` becomes the second degree of freedom, and `n3` becomes the third degree of freedom. Entering a zero for any degree of freedom turns off that degree of freedom when plotting deformed shapes. By default, plotting is done with all degrees of freedom turned on and in their logical order (dofs,1,2,3).

`dplo,<n1,n2>`

The `DPLOt` command may be used (in interactive mode only) to specify any line for a plot of an `n1` displacement component. After entering the command the LEFT mouse button is used to select the ends of a line through or in the mesh which defines the location for the displacement plot.

After entering the two ends for the line (labeled A and B), an X-Y plot for the `n1` displacement component is superposed on the screen. The X-axis of the plot corresponds to the selected A-B mesh line. The Y-axis of the plot displays the magnitude of projected displacement component along the line. The magnitude of displacement plotted is proportional to the largest and smallest values which occur anywhere in the mesh. A contour plot of the displacement component may be used to identify locations for the maximum and minimum (use the `CONTour` command). If `n2` is non-zero it is used as the plot number (up to 12 plots may be placed on the same figure). If `n2` is zero, the previous plot number is incremented and assigned as the current plot number.

The `DPLOt` command may be combined with `SPLOt` to show all quantities along selected lines. Currently, this command works for 2-D problems displayed in a `CARTesian` mode only.

```
eige,n1,v1
```

Plot eigenvectors for last element computed by a `TANG`ent or `UTAN`gent command (which must be performed before entering a plot mode). The parameter `n1` specifies the vector number (sorted by increasing eigenvalues) and `v1` may be zero, positive or negative. If `v1` is positive it specifies the plot color; if not set, the eigenvector number is used for the plot color. Before using this command, execute `DEFO`rm. Using `PICK` to zoom in on the element is also helpful. `UNDE,,1` may be used to show the undeformed element for comparison purposes.

```
eigv,n1,n2,n3
```

Plot information related to eigenvector `n1`. An eigensolution must be performed (see `SUBS`pace command in Appendix B) before attempting an eigenvector plot. The plot mode must also be set as `DEFO`rmed.

If `n3` is zero a deformed plot for the superposed eigenvector will be given.

If `n3` is nonzero contours for the `n3` degree of freedom for eigenvector `n1` will be constructed according to the value specified in `n2`.

For `n2` positive, `n2` contour values will be constructed. The values for each contour must be specified after the command language program for batch execution or at the prompt for interactive execution. Eight values per record are input. The number for the first contour is specified on the record (or prompt) immediately following the values.

For `n2` non-positive, a fill-type plot will be constructed. The maximum and minimum value of the quantity to be plotted must be given. The program will compute equally spaced intervals between these values for the plot. Alternatively a blank record may be input and the program will select values to be plotted based on maximum and minimum values of the component.

```
elem,n1
```

Plot numbers in or near the elements appearing in the visible plot region. If `n1` is non-zero plot number for specified element number only. After a `PICK`, `CLIP` or `ZOOM` some numbers may appear for elements surrounding the plot region even though no lines for element edges are shown.

```
estr,n1,n2,n3
```

This command functions exactly like `STRE`ss, except that the quantities plotted are done without inter-element smoothing.

The command plots contours of stresses, strains and for coupled thermo-mechanical problems heat flux (or other element variables), where `n1` is the component to be plotted and `n2` is the number of contours (same as for `CONT`our including shading options). The definitions of `n1` for 2 and 3 dimensional elasticity problems are:

n1	Component	n1	Component
1	11-stress	7	11-strain
2	22-stress	8	22-strain
3	33-stress	9	33-strain
4	12-stress	10	12-strain
5	23-stress	11	23-strain
6	31-stress	12	31-strain
		13	1-heat flux
		14	2-heat flux
		15	3-heat flux

The `n3` parameter is used for filled (solid color) stress plots as follows:

n3	Action
0	superpose mesh on plot
1	suppress showing mesh
-1	suppress showing mesh and uses previously set contour values

For contour line plots (`n2` > 0), a zero `n3` value will suppress numbers near each contour line (same as `CONT`our). Default: n3 = 0.

```
ewir,n1,n2,n3
```

This command functions exactly like `STRE`ss, except that the quantities plotted are done without inter-element smoothing. The mesh is displayed as in a wire-frame mode with contour values of element stresses are added to each edge.

The command plots contours of stresses, strains and for coupled thermo-mechanical problems heat flux (or other element variables), where `n1` is the component to be plotted and `n2` is the number of contours (same as for `CONT`our including shading options). The definitions of `n1` for 2 and 3 dimensional elasticity problems are:

n1	Component	n1	Component
1	11-stress	7	11-strain
2	22-stress	8	22-strain
3	33-stress	9	33-strain
4	12-stress	10	12-strain
5	23-stress	11	23-strain
6	31-stress	12	31-strain
		13	1-heat flux
		14	2-heat flux
		15	3-heat flux

The `n3` parameter is used for filled (solid color) stress plots as follows:

n3	Action
0	superpose mesh on plot
1	suppress showing mesh
-1	suppress showing mesh and uses previously set contour values

For contour line plots (`n2` > 0), a zero `n3` value will suppress numbers near each contour line (same as `CONT`our). Default: n3 = 0.

```
exno,n1
```

The `EXNOde` command displays the position of all exterior nodes on a mesh. If `n1` is negative, only the node position is shown, if `n1` is zero, numbers are placed near the position of all super nodes.

Individual nodes may be displayed using the `NODE` command.

```
eyes
```

This command allows one to pick the viewpoint in `PERS`pective plotting using the mouse. After issuing `EYES`, the users simply clicks the left mouse button at the desired viewpoint location in the 1-2 and 1-3 coordinate planes shown in the upper right hand corner of the plot window. After entering the desired coordinates, `HIDE` is automatically called by eyes to construct the new visible mode. The view point may be re-selected until the desired view is obtained. Use of the middle or right mouse buttons exits the `EYES` mode.

```
fact,v1
```

The entire plot is scaled by the value of `v1` (default = 1.). It is often better to scale the plot using `SCAL`e to permit the entire deformed region to appear in the screen area.

```
fill,n1,n2,n3
```

This command presents a plot of the mesh in which the faces of each element are filled in color. If `n1` is zero the color is set by the material number of the elements for the current material setting (see plot `MATErial` command). In three dimensional problems the plot must be in perspective view to display visible surfaces.

When `n1` is positive it denotes the color used for all elements.

If `n1` is negative the colors are selected based on the 'region' number (see command `REGIon` in Mesh Input Manual (Appendix A).

A non-zero value of the parameter `n2` may be used to suppress the display of the current value of problem 'time' on the display.

Normally, each element boundary is visible, however if `n3` is non-zero the boundary is drawn in the color of the filled face.

Example 1:

```
PLOT FILL
```

produces a display of the faces using the element material number to select colors.

Example 2:

```
PLOT FILL 2 0 1
```

displays all element faces in color 2 (red). Boundaries of each element are drawn in color 2 also. The effect is one where the entire region is shaded red. In three dimensional problems the edges of the mesh are not visible. It is better to then use the plot `OUTLine` command to see edges.

Example 3:

```
PLOT FILL -1
```

displays all element faces using their region number to select the color.

```
fram,n1
```

This command defines a region in the screen plot window according to the following options:

n1	Region used
0	Entire window used
1	Upper left quadrant
2	Upper right quadrant
3	Lower left quadrant
4	Lower right quadrant
	(Default: n1 = 0)

By using different frames, a large amount of information may be placed on a single screen. Each part of `FRAM`e may be cleared independently for some devices using a `WIPE,n1` command.

```
full
```

This command works only in the full screen Windows version.

Using the `PLOT FULL` command converts the main plot window to a full screen mode. The displayed text will be very limited. Use `PLOT NOFUll` to return to the default mode. It may be necessary to clear the screen again using `PLOT WIPE` to obtain a proper display of borders and the *FEAP* logo.

```
hide,n1,n2,n3
```

The `HIDE` command is used to compute the surface facets for a three dimensional solid region. Subsequent plots are then given on the surface facets only. A pseudo hidden surface routine is accomplished by sorting the facets and plotting from the one most distant from the viewer to the one closest.

If `n1` is `-1` the outline of facets is white; if `n1` is less than `-1` the outline is black (and invisible on the screen). If `n2` is non-zero then all boundary facets are plotted. If `n3` is positive the color is set to n3.

HISToryp FEAP PLOT COMMAND MANUAL

```
hist,n1 n2 n3
```

Plot contours of history variable `n1` is the component to be plotted and `n2` is the number of contours (same as for the `CONT`our command including shading options). For material sets that have different interpretation for the `n1` component a `PLOT MATE ma` command should be used before requesting the plot.

The `n3` parameter is used for filled (solid color) stress plots as follows:

n3	Action
0	superpose mesh on plot
1	suppress showing mesh
-1	suppress showing mesh and uses previously set values

For contour line plots (`n2` > 0), a zero `n3` value will suppress numbers near each contour line (same as `CONT`ours). Default: `n3 = 0`.

`imag`

This command sets plots to the imaginary part of complex contours of the dependant variable (using `CONT`ours). Default is `REAL`.

This command only works with solutions with complex (real and imaginary) values. That is, the initiation of a problem must start with

```
*complex
feap * * title information
  ....
```

to initate the storage of complex arrays.

```
jpeg
```

In a UNIX environment, the `JPEG` command may be used to produce a `jpeg` file of the entire current graphics screen. The files are named sequentially from `Feap00001.jpg`.

```
labe
```

The **LABE**l command will enable the display of contour plot scales on the right side of the plot window. The labels may be turned off using the **NOLA**bel command. The default mode is enabled. This command may be used to produce contour PostScript output files without the labels.

```
line,n1,v1
```

The `LINE` command may be used to set the line type. This command only affects PostScript outputs. The line type is assigned by the value of `n1` as:

Number	Line Type
0	solid
1	dotted
2	dash-dot
3	short dash
4	long dash
5	dot-dot-dash
6	short dash-long dash
7	wide dash

The width of the line is set using `v1` which may have values between 0.0 and 2.0 (normal is 1.0).

LIST FEAP PLOT COMMAND MANUAL

```
    list,n1
```

Output of specific material set numbers is activated using the plot command

```
       PLOT LIST n
\ene{verbatidm}
where \texttt{n} is the material list number to be plotted.
Individual sets \texttt{ma}
in the list may be displayed using the \texttt{PLOT MATE ma} command
followed by the display command.

The material set list is given with a solution command \texttt{list mate n}.

Example:
\begin{verbatim}
       BATCh
         LIST MATE 1
       END
       1 3

       BATCh
         PLOT LIST 1
         PLOT CONT 1  !Outputs materials 1 and 3 only
         PLOT MATE 3
         PLOT CONT 1  !Outputs material 3 only
          ...
       END
```

```
load,<tip,length>
```

The `LOAD` command may be used to display the applied forces acting on the nodes of the system. Prior to any solution steps all specified loads are displayed; however, after any solution step only the non-zero forces are displayed. these are obtained by multiplying the load intensity by the appropriate proportional load value.

Normally, the tail of the vector is placed on the node, however, if the parameter `tip` is positive the head of the vector is placed on the node. The scale factor for forces is automatically specified unless the parameter `length` is given a non-zero positive value which then serves as the scaling factor for forces.

LOGO FEAP PLOT COMMAND MANUAL

```
logo
```

The `LOGO` command places a new *FEAP* logo on the screen. May be used to add logo to postscript plots.

MANUal FEAP PLOT COMMAND MANUAL

```
manu,level
```

The `MANU`al command will set the `level` of help commands shown when the command `HELP` is given in any solution mode. The levels are: 0 = basic; 1 = intermediate; 2 = advanced; 3 = expert. The default level is 0.

MARK FEAP PLOT COMMAND MANUAL

```
mark,p1
```

When `p1` is `on` (or blank), the `MARK` command shows the location of maxima and minima on any contour plot. The default is `p1` = `off`.

```
mate,n1,n2
```

The `MATE`rial command is used to indicate which material number is to be active during contour or fill plots. The `n1` value is the material set identifier, and a value of zero indicates all materials are to be displayed. (Default: n1 = 0).

The parameter `n2` defines the material number and allows plot of reactions only for this number.

```
mesh
```

The `MESH` command causes a display of the current view of the mesh to be displayed in a line (wire-frame) mode. A surface mesh may also be displayed using the `FILL` command (N.B. For 3-d problems it is necessary to use a perspective view and the `HIDE` option for fill views to function correctly).

```
midn
```

The plot `MIDNode` command is used to fill missing values in meshes where different order approximations are used for the degree of freedoms. For example in Taylor-Hood type elements which use quadratic order interpolation for displacement (or in fluids velocity) values but linear for pressure. The values at mid-edge or mid-face nodes for the pressure will all be zero. If contour plots are then attempted erroneous looking results are obtained. The `MIDNode` command merely fills in the missing values by averaging vertex values. The solution command is given as

```
PLOT MIDNode
```

with no arguments. All degrees of freedom for every element are searched fo fill only the missing middle values. The command may also be given in solution mode as

```
MIDNode
```

N.B. Element types must be specified during mesh creation. That is, using element generations the `TYPE` of element must be given. For example if the elements are quadrilaterals the command is given as

```
ELEMent  TYPE=QUADrilateral ....
```

similarly for `BLOCk` or `BLENd` inputs.

```
moff
```

The display of mesh borders on filled plots may be turned off using the plot command

```
PLOT MOFF
```

Thus for subsequent filled plots it is only necessary to use

```
PLOT CONT n
```

to display filled contours for displacement component `n`. The mesh outline may be restored using the command

```
PLOT MON
```

```
mon
```

The display of mesh borders on filled plots may be turned on using the plot command

```
PLOT MON
```

For subsequent filled plots it is necessary to use

```
PLOT CONT n,,1
```

to display filled contours for displacement component `n`. The mesh outline may be turned off using the command

```
PLOT MOFF
```

which permits filled plots without mesh display given as

```
PLOT CONT n
```

NODE FEAP PLOT COMMAND MANUAL

```
node,n1,n2
```

The `NODE` command displays the position of all nodes. If `n1` is negative, only the node position is shown, if `n1` is positive the node numbers with the values between `n1` and `n2` are placed near the node position; if `n1` is zero numbers are placed near the position of all nodes.

`nofu`

This command works only in the full screen Windows version.

Using the `PLOT NOFUll` command returns the screen to permit display of three different plot windows. Control of the window to receive plot information is given using the `PLOT WINDow` command. Use `PLOT FUll` to create a full screen display for the main plot window. It may be necessary to clear the screen again using `PLOT WIPE` to obtain a proper display of borders and the *FEAP* logo.

NOLAbel FEAP PLOT COMMAND MANUAL

```
nola
```

The NOLAbel command will disable the display of contour plot scales on the right side of the plot window. The labels may be turned on using the LABEl command. The default mode is enabled. This command may be used to produce contour PostScript output files without the labels.

```
nopr
```

The `NOPRint` command will suppress the print mode for interactive plot prompts. A `PRINt` command will enable prints. The default is `PRINt`.

```
norange
```

This command turns off fixed plot ranges. To turn on a fixed range for a subsequent plot use the command `RANGe`. The default is range *off*.

```
outl
```

The `OUTL`ine command causes a plot of an outline for the current view of the mesh to be displayed. For a perspective view of three dimensional bodies displayed after a `HIDE`n surface construction an edge definition is displayed.

```
pair,n1,n2,n3
```

Plot contact surface `n1` to `n2`. The parameter `n3` may be used to plot only the slave or the master side: `n3 = 1` plots slave surface, whereas `n3 = 2` plots the master surface. When `n3 = 0` both sides of the requested surfaces are displayed.

Where plot surface facets are defined by a single node the surface will appear as very small 'dots' on the screen. For other facet types the surface is displayed as a line drawing which outlines each target facet. For a properly defined contact surface each set of facets should define a closed region on a body.

```
pbou,n1
```

This command may be used to interactively add or delete boundary conditions for the `n1` degree of freedom using a graphical plot and the mouse. After entering the command, the text window will display use options: the LEFT mouse button is used to add a restraint to the `n1` degree of freedom for the node closest to the mouse cursor; the RIGHT button is used to delete a restraint; and, the MIDDLE button is used to terminate the input. The command works only in interactive mode. As restraints are added a diagonal slash is added on the node selected. If this is not the node desired, the restraint may be removed and the slash should disappear. After the MIDDLE button is pressed, the mesh should be erased and the `BOUN`dary command should be used to display the active restraints. The command must be given separately for each set of degree of freedom components to be restrained.

Currently, this command works for one and two dimensional problems.

The restraints active at the time the MIDDLE button is pressed will be saved in a file which is named `Ixxxx.bou`, where Ixxxx is the problem name. The data may be merged with the input file by using an include option (e.g., place a command "incl,Ixxxx.bou" in the input file).

PDISplacement FEAP PLOT COMMAND MANUAL

```
pdis,n1,value
```

This command may be used to interactively add or delete displacement boundary values for the `n1` degree of freedom using a graphical plot and the mouse. After entering the command the text window will display use options: the LEFT mouse button is used to add a displacement "value" for the `n1` dof of the node closest to the mouse cursor; the RIGHT button is used to delete a displacement value; and, the MIDDLE button is used to terminate the input. The command works only in interactive mode. As displacement values are added a diagonal slash is added on the node selected. If this is not the node desired, the displacement value may be removed and the slash should disappear. The command must be given separately for each set of degree of freedom components.

Currently, this command works for one and two dimensional problems.

The displacement values active at the time the MIDDLE button is pressed will be saved in a file which is named `Ixxxx.dis`, where Ixxxx is the problem name. The data may be merged with the input file by using an include option (e.g., place a command "incl,Ixxxx.dis" in the input file).

```
pele
```

The PELEment command may be used to plot features in user developed elements. Plots are constructed for switch value 20 (isw = 20) in the user element. See programmer manual for development of user elements for *FEAP*.

```
pers,n1
   Requires:
      inew:
      vx,vy,vz
     ex,ey,ez
```

All subsequent plots are to be drawn in a three dimensional perspective view. A plot may also be in a cartesian two dimensional view (see `CART`esian plot manual). The default plot mode is cartesian.

If the parameter `n1` is non-zero additional data is input (otherwise program computed values are used for the perspective view). In interactive mode prompts will be given for the additional data. In batch mode the additional data is given after the batch `END` command and ordered so that reads are performed at the execution of the correct instruction.

The data to be input is:

```
inew
vx vy vz
ex ey ez
```

where

inew	0 for input of new parameters
inew	1 for use of old parameters

If new parameters are to be specified input:

vx	x-coordinate of view point
vy	y-coordinate of view point
vz	z-coordinate of view point

and

ex	x-component of vertical vector
ey	y-component of vertical vector
ez	z-component of vertical vector

The view point and a target point computed at the center of the body establish the view direction; the vertical vector for the screen establishes the orientation of the body with respect to the view direction.

```
pfor,n1,value
```

This command may be used to interactively add or delete force boundary values for the `n1` degree of freedom using a graphical plot and the mouse. After entering the command the text window will display use options: the LEFT mouse button is used to add a force "value" for the `n1` dof of the node closest to the mouse cursor; the RIGHT button is used to delete a force value; and, the MIDDLE button is used to terminate the input. The command works only in interactive mode. As force values are added a diagonal slash is added on the node selected. If this is not the node desired, the force value may be removed and the slash should disappear. After the MIDDLE button is pressed, the mesh should be erased and the `LOAD` command should be used to display the active loads. The command must be given separately for each set of degree of freedom components.

Currently, this command works for one and two dimensional problems.

The force values active at the time the MIDDLE button is pressed will be saved in a file which is named `Ixxxx.frc`, where Ixxxx is the problem name. The data may be merged with the input file by using an include option (e.g., place a command "incl,Ixxxx.frc" in the input file).

```
pick
```

The `PICK` command may be used to select a portion of the plot region as a new plot region. The command uses the mouse for selection. Prompts are given to select two points from the screen with the left mouse button; these two points are used to center a new square plot region. The command may be used repeatedly to identify successively smaller parts of the mesh as the plot region. The command `ZOOM` may be used to restore the full mesh to the plotting region. The command works in any plot mode (e.g., Cartesian, perspective) for 2 or 3 dimensional problems.

`pnod`

This command may be used to interactively identify the numbers of nodes. After entering the command the text window will display use options: the LEFT mouse button is used to select nodes; the number for the node closest to the mouse cursor will be printed; the MIDDLE button is used to terminate the selection.

Currently, this command works only for one and two dimensional problems.

POSTscript FEAP PLOT COMMAND MANUAL

```
post,n1,n2
```

The `POST`Script command will enable the output of a PostScript file for later use in producing hard copy plots. The sequence is initiated by the first `POST`Script command (a non-zero `n1` is in landscape mode, a zero value is in portrait mode). The name of the file containing the output is `Feapxxxx.eps` (where xxxx is between `aaaa` and `zzzz`) and appears on the text screen. Subsequent commands will produce plots which appear on the screen and will also send information to the output file. A second `POST`Script command closes the output file and subsequent commands will give plots only on the screen.

Up to 456,976 PostScript output files may be produced during a work session. The program checks for existence of a file before the open operation. Files must be purged by the user outside a *FEAP* execution session. Note that PostScript files can be quite large and disk quotas can easily be exceeded.

If `n2` is non-zero the *FEAP* logo is printed in the PostScript file by any plot command that cause it to be printed on the screen (e.g., `WIPE`). Default: n2 = 0.

```
prax,n1,n2,n3
```

This command plots principal stress axes for 2 and 3 dimensional problems. The parameters are interpreted as follows:

n1	Description
0	all principal directions shown (or blank)
1	maximum principal direction only (2 and 3-D)
2	middle principal directions only (3-D) minimum principal direction only (2-D)
3	minimum principal direction only (3-D only)

n2	Description
0	principal directions associated with both negative and positive principal values shown
< 0	only principal directions associated with negative principal values shown
> 0	only principal directions associated with positive principal values shown

n3 corresponds to different plotting colors (range 1-7).

PRINt FEAP PLOT COMMAND MANUAL

```
prin
```

The PRINt command will enable the print mode for interactive plot prompts. The use of the command NOPRint will disable prints. The default is PRINt.

PROFile FEAP PLOT COMMAND MANUAL

```
profile,v1
```

This command displays a view of the profile for the tangent matrix. If the parameter `v1` is zero only the upper half is shown; whereas for non-zero values both sides are displayed. Using before and after a profile optimization command (see solution command `OPTImize`) provides a view of the effectiveness of each solution mode. Should be used only in a cartesian view (See command `CARTesian`).

PROJection FEAP PLOT COMMAND MANUAL

```
proj
```

The PROJection command will force a new computation of nodal projections for element results (e.g., nodal stresses and principal stresses). When used in conjunction with the MATErial command a correct projection of stresses at material interfaces may be obtained. In default mode, all materials are projected thus leading to incorrect representations at material interfaces. The use of the command sequence:

```
MATE,1
PROJ
STRE,1
MATE,2
PROJ
STRE,1
```

for a two material model, any discontinuities in the 1-stress at the interface between material sets 1 and 2 will be preserved.

```
prom,on
prom,off
```

Normally, *FEAP* will issue prompts for parameters needed to construct plots. Usually, default values may be accepted by pressing the return (or enter) key. The `PROMpt` command may be used to eliminate the need to press the key to accept the default values. The command has one parameter which is either `ON` or `OFF`. Omitting the parameter turns off the prompts.

```
psno
```

This command may be used to interactively identify the numbers for supernodes nodes. After entering the command the text window will display use options: the LEFT mouse button is used to select nodes; the number for the node closest to the mouse cursor will be printed; the MIDDLE button is used to terminate the selection.

Once a node is identified options to reposition the node are given. May be used to regenerate mesh for desired spacings.

Currently, this command works only for one and two dimensional problems.

PSTRess FEAP PLOT COMMAND MANUAL

```
pstr,n1,n2,n3
```

Plot contours of principal stresses, where `n1` is the component to be plotted and `n2` is the number of contours (same as for the `CONT`our command including shading options). The `n1` for 2 and 3 dimensional elasticity problems are:

n1	Component
1	1-principal stress
2	2-principal stress
3	3-principal stress (3-d) or angle (2-d)
4	Maximum shear (2-d)
5	I_1 Stress invariant
6	J_2 Stress invariant
7	J_3 Stress invariant

The `n3` parameter is used for filled (solid color) stress plots as follows:

n3	Action
0	superpose mesh on plot
1	suppress showing mesh
-1	suppress showing mesh and uses previously set values

For contour line plots (`n2` > 0), a zero `n3` value will suppress numbers near each contour line (same as `CONT`ours). Default: n3 = 0.

```
pvie
```

The plot commane `PVIEw` is used to output files for processing by the Paraview program. The command is given as

```
BATCh
  PLOT PVIEw
END
```

is mainly used to output files in time dependent solutions. The file names are set from the *plot* file name specified at start of a *FEAP* solution (`fplt`) and appended by a numerical sequence ranging from `00000` to `99999`. Thus the first file would be designated as `fplt00001.vtu` the second as `fplt00002.vtu` and so forth for subsequtne outputs.

```
pwir,n1,n2,n3
```

Plot contours of principal stresses, where `n1` is the component to be plotted and `n2` is the number of contours (same as for the `CONT`our command including shading options). The mesh is displayed as in a wire-frame mode with contour values of principal stresses are added to each edge. The `n1` for 2 and 3 dimensional elasticity problems are:

n1	Component
1	1-principal stress
2	2-principal stress
3	3-principal stress (3-d) or angle (2-d)
4	Maximum shear (2-d)
5	I_1 Stress invariant
6	J_2 Stress invariant
7	J_3 Stress invariant

The `n3` parameter is used for filled (solid color) stress plots as follows:

n3	Action
0	superpose mesh on plot
1	suppress showing mesh
-1	suppress showing mesh and uses previously set values

For contour line plots (`n2` > 0), a zero `n3` value will suppress numbers near each contour line (same as `CONT`ours). Default: n3 = 0.

`quad`

The `QUAD`rant command may be used in combination with the `SYMM`etry command to select which quadrant(s) will be output by subsequent plot commands. After issuing the command the user selects one or more quadrants for the active symmetries. A +1 is used for a positive quadrant and a -1 for a negative quadrant. If no values are entered, all quadrants for the current symmetry set become active. After selecting the quadrants to view a return is entered (blank record).

RANGe FEAP PLOT COMMAND MANUAL

```
rang,v1,v2
```

The values of `v1` and `v2` are used to set the plot range for the next plot. The range of the plot will be set so that `rangemin = min(v1,v2)` and `rangemax = max(v1,v2)`. The values will be used until the range is reset or turned of. In interactive mode the range may be turned off using the command `rang,off`. In batch mode the command `NORAnge` must be used. The default is range `off`.

REACtions FEAP PLOT COMMAND MANUAL

```
reac,<tip,length>
```

Plot nodal reactions for current solution state. If `tip` is non-zero the vector tip will appear next to the node; whereas, if `tip` is zero the tail of the load vectors are on the nodes. The maximum length will be automatically scaled. All other reactions will be scaled in proportion to the maximum, accordingly some very small values may scale to be too small to be visible. If `length` is positive it is used as the scaling factor for reaction components and can assist in making small values visible.

```
real
```

This command sets plots to the real part of real or complex contours of the dependant variable (using `CONT`ours). Default is real.

REFResh FEAP PLOT COMMAND MANUAL

```
refr
```

The REFResh command will redisplay all plot segments which are in the current X-window buffer. No action is taken for other display types.

```
regi,n1
```

The `REGIon` command is used to indicate which region number is to be active during contour or fill plots. The `n1` value is the region number, and a value of zero indicates all regions are to be displayed. (Default: n1 = 0).

```
rota,i_axis,a_degree <mesh>
rota
```

The `ROTAte` command is used to rotate an object displayed in a perspective view about the `i_axis` (1,2 or 3) of the object by an `a_degree` angle. For example, in an interactive `PLOT` state with an object in a perspective view, the command

```
ROTA 3 30
```

would rotate the object about its x_3 axis by 30-degrees. Anys subsequent plot of the object would give the new rotated view. If the command is given in the form

```
ROTA 3 30 1
```

a plot of the rotated mesh would be superposed on the current view. Giving a subsequent command

```
ROTA 2 -60 1
```

whould apply a second rotation to the current object about its x_2 axis by a -60-degree angle. Angles are described using a standard right hand rule. To reset the rotation state to the default view may be given by the command

```
ROTA
```

with no parameters.

SCALe FEAP PLOT COMMAND MANUAL

```
scal,v1,v2
```

Displacements are scaled by value `v1` (Default v1 = 1). If `v2` is zero, the plot region is resized to permit both the undeformed and the deformed plot to appear on the screen.

SCREen FEAP PLOT COMMAND MANUAL

```
scre,<on,off>
```

The `SCRE`en command will turn `on` or `off` the display of plot information to the screen. This command is intended for use in constructing PostScript outputs in which it is desired to not change the plot image on the screen. Issuing of plot commands while screen is off will send information only to the PostScript file (if it is active). The default is screen on.

SHOW FEAP PLOT COMMAND MANUAL

```
show
```

The `SHOW` command will output the state of for several plot parameters to the text window.

SIZE FEAP PLOT COMMAND MANUAL

```
size,n1
```

Specify the size of text to be plotted.

n1	Text size
1	small
2	normal (default)
3	large

This command is not active for all devices.

```
snod,n1,n2
```

The `SNODE` command displays the position of super nodes used for blending mesh constructions. If `n1` is negative, only the node position is shown, if `n1` is positive the node numbers with the values between `n1` and `n2` are placed near the node position; if `n1` is zero, numbers are placed near the position of all super nodes.

SPLOts FEAP PLOT COMMAND MANUAL

```
splo,<n1,n2>
```

The SPLOt command may be used (in interactive mode only) to specify any line for a plot of an n1 stress component. After entering the command the LEFT mouse button is used to select the ends of a line through or in the mesh which defines the location for the stress plot.

After entering the two ends for the line (labeled A and B), an X-Y plot for the n1 stress component is superposed on the screen. The X-axis of the plot corresponds to the selected A-B mesh line. The Y-axis of the plot displays the magnitude of projected stress component along the line. The magnitude of stress plotted is proportional to the largest and smallest values which occur anywhere in the mesh. A contour plot of the stress component may be used to identify locations for the maximum and minimum (use the STREss command). If n2 is non-zero it is used as the plot number (up to 12 plots may be placed on the same figure). If n2 is zero, the previous plot number is incremented and assigned as the current plot number.

The SPLOt command may be combined with DPLOt to show all quantities along selected lines. Currently, this command works for 2-D problems displayed in a CARTesian mode only.

STRAin FEAP PLOT COMMAND MANUAL

```
stra,n1,n2,n3
```

Plot contours of strains, where `n1` is the component to be plotted and `n2` is the number of contours (same as for the `CONT`our command including shading options). The `n1` for 2 and 3 dimensional elasticity problems are:

n1	Component
1	11-strain
2	22-strain
3	33-strain
4	12-strain
5	23-strain
6	31-strain

The `n3` parameter is used for filled (solid color) stress plots as follows:

n3	Action
0	superpose mesh on plot
1	suppress showing mesh
-1	suppress showing mesh and uses previously set values

For contour line plots (`n2` > 0), a zero `n3` value will suppress numbers near each contour line (same as `CONT`ours). Default: n3 = 0.

```
stre,n1,n2,n3
```

Plot contours of stresses, strains and for coupled thermo-mechanical problems heat flux, where `n1` is the component to be plotted and `n2` is the number of contours (same as for the `CONT`our command including shading options). The `n1` for 2 and 3 dimensional elasticity problems are:

n1	Component	n1	Component
1	11-stress	7	11-strain
2	22-stress	8	22-strain
3	33-stress	9	33-strain
4	12-stress	10	12-strain
5	23-stress	11	23-strain
6	31-stress	12	31-strain
		13	1-heat flux
		14	2-heat flux
		15	3-heat flux

The `n3` parameter is used for filled (solid color) stress plots as follows:

n3	Action
0	superpose mesh on plot
1	suppress showing mesh
-1	suppress showing mesh and uses previously set values

For contour line plots (`n2` > 0), a zero `n3` value will suppress numbers near each contour line (same as `CONT`ours). Default: n3 = 0.

```
symm,n1,n2,n3
```

The `SYMM`etry command permits reflection of the mesh about each coordinate direction. If `n1` is nonzero the reflection is with respect to the x-1 coordinates. Thus,

```
SYMM,1,0   or SYMM,1
```

produces a plot which includes elements defined using positive x-1 coordinate values and also using negative x-1 coordinate values. Symmetric reflections for 1, 2, and 3 dimensional problems is possible.

A prompt for the value of the coordinate to use in constructing the reflection is also given. Thus, using:

```
2.5,1.5
```

will reflect coordinates about x1 = 2.5 and x2 = 1.5 (for a 3d problem, x3 = 0.0).

```
swir,n1,n2,n3
```

Plot contours of stresses, strains and for coupled thermo-mechanical problems heat flux, where `n1` is the component to be plotted and `n2` is the number of contours (same as for the `CONT`our command including shading options). The mesh is displayed as in a wire-frame mode with contour values of stress added to each edge.

The `n1` for 2 and 3 dimensional elasticity problems are:

n1	Component	n1	Component
1	11-stress	7	11-strain
2	22-stress	8	22-strain
3	33-stress	9	33-strain
4	12-stress	10	12-strain
5	23-stress	11	23-strain
6	31-stress	12	31-strain
		13	1-heat flux
		14	2-heat flux
		15	3-heat flux

The `n3` parameter is used for filled (solid color) stress plots as follows:

n3	Action
0	superpose mesh on plot
1	suppress showing mesh
-1	suppress showing mesh and uses previously set values

For contour line plots (`n2` > 0), a zero `n3` value will suppress numbers near each contour line (same as `CONT`ours). Default: n3 = 0.

```
text,n1,x,y
```

Enter text to be placed in plot region. Prompt are given for the text to be entered. The *backspace* key may be used to delete text before it is placed on the screen. A null string will not appear This command uses graphical input (GIN) with a mouse and cross-hairs to position the lower left corner of the text in the view window. After the cross-hairs have been positioned to the desired location, the text is placed on the screen by pressing the left mouse button.

Use the `n1` parameter to specify the color of each text (default is last color plotted - initially white).

If the parameters `x` and `y` are input, the text will automatically be placed at the specified (x,y) location. The location is input in normalized screen coordinates; thus, $0 < \mathrm{x} < 1.4$ and $0 < \mathrm{y} < 1$.

```
time,<on,off>
```

The `TIME` command may be used to remove the time label from the right of the graphics window (use `OFF` - in interactive mode only). May be restored using the `ON` option. Primarily for use with postscript output where it is desired to have a borderless plot with a small bounding box.

TITLe FEAP PLOT COMMAND MANUAL

```
titl,n1
```

The `TITLe` command may be used to add the current problem title to the graphics in the color `n1`. The title appears at the bottom of the plot window. It may also be added to PostScript outputs using this command. Other text may be placed in the graphics region using the `TEXT` command.

```
tria,,,v1
```

The orientation of the nodal directors for shell and frame problems may be displayed using the

```
PLOT TRIAd,,,v1
```

plot command where `v1` is the color of the director.

```
unde,,n2
```

This command will set the plot options to be associated with a undeformed mesh.

Specification of a non-zero `n2` value retains the plot scaling to a previously set value. This permits superposition of deformed solutions on the current plot for comparison purposes.

A deformed option is specified using the `DEFO`rm command.

```
uplot,v1,v2,v3
```

The `UPLOt` command depends on options added by users. To make this command operational it is necessary to write a user subprogram

```
      SUBROUTINE UPLOT(CT)

      IMPLICIT   NONE

      REAL*8     CT(3)

C     Users to add plot commands here

      END
```

and compile with the main program and archives.

See *FEAP* Programmers Manual for more information on writing the subprogram.

```
velo,n1,n2,n3
```

Plot contours for velocity degree of freedom `n1` (default is 1). Two options are available to construct contours:

1. If `n2` is zero or negative, areas between contours will be shaded in colors or gray scale. For this case, only the minimum and maximum contour values are specified - by default (enter return) the program constructs 12 evenly spaced intervals for the shading.

   The default is to set the range based on the maximum and minimum values for the contours plotted. The range displayed may be reset using the plot `RANGe` command.

   If `n3` is positive, plotting of the mesh is suppressed. If `n3` is negative, plotting of the mesh is suppressed and the previously existing contour values are used. Note that the contours must have been already set by a previous call to `VELO`city for this option to function properly.

2. If `n2` is a positive number specific contour lines may be designated and plotted as lines. It is necessary to define the value for each of the `n2` contour lines. If `n3` is non-zero a numerical label will be added near each contour indicating the relationship to a value table given on the screen.

If the default settings are turned off using the plot `DEFAult OFF` command then in interactive mode, after a `VELO,n1,n2,n3` command is given prompts for additional data will appear. For each contour line the values to be plotted should be entered (maximum of 8 items per record). Maximum and minimum existing values are indicated on the screen. For shaded plots only a lower and an upper value separating the smallest and largest shading from their adjacent ones are input.

Note that default settings may be restored using the `DEFAult ON` command.

`vplo,<n1,n2>`

In transient problems, the `VPLOt` command may be used (in interactive mode only) to specify any line for a plot of a `n1` velocity component. After entering the command the LEFT mouse button is used to select the ends of a line through or in the mesh which defines the location for the velocity plot.

After entering the two ends for the line (labeled A and B), an X-Y plot for the `n1` velocity component is superposed on the screen. The X-axis of the plot corresponds to the selected A-B mesh line. The Y-axis of the plot displays the magnitude of projected velocity component along the line. The magnitude of velocity plotted is proportional to the largest and smallest values which occur anywhere in the mesh. A contour plot of the velocity component may be used to identify locations for the maximum and minimum (use the `CONTour` command). If `n2` is non-zero it is used as the plot number (up to 12 plots may be placed on the same figure). If `n2` is zero, the previous plot number is incremented and assigned as the current plot number.

Currently, this command works only for 2-D problems displayed in a `CARTesian` mode.

```
vwir,n1,n2,n3
```

Plot contours for velocity degree of freedom `n1` in a wire-frame mode (default is `n1 = 1`). The mesh is displayed as in a wire-frame mode with contour values of the velocity added to each edge. Two options are available to construct contours:

1. If `n2` is zero or negative, areas between contours will be shaded in colors or gray scale. For this case, only the minimum and maximum contour values are specified - by default (enter return) the program constructs 12 evenly spaced intervals for the shading.

   The default is to set the range based on the maximum and minimum values for the contours plotted. The range displayed may be reset using the plot `RANGe` command.

   If `n3` is positive, plotting of the mesh is suppressed. If `n3` is negative, plotting of the mesh is suppressed and the previously existing contour values are used. Note that the contours must have been already set by a previous call to `VWIRe` for this option to function properly.

2. If `n2` is a positive number specific contour lines may be designated and plotted as lines. It is necessary to define the value for each of the `n2` contour lines. If `n3` is non-zero a numerical label will be added near each contour indicating the relationship to a value table given on the screen.

If the default settings are turned off using the plot `DEFAult OFF` command then in interactive mode, after a `VWIR,n1,n2,n3` command is given prompts for additional data will appear. For each contour line the values to be plotted should be entered (maximum of 8 items per record). Maximum and minimum existing values are indicated on the screen. For shaded plots only a lower and an upper value separating the smallest and largest shading from their adjacent ones are input.

Note that default settings may be restored using the `DEFAult ON` command.

WINDow FEAP PLOT COMMAND MANUAL

```
wind,n1
```

The `WIND`ow command may be used to select a screen number for the active plot region. The value of `n1` denotes the window number. The PC version of *FEAP* has three windows. The X-Windows version has only one screen. The main screen is one (1).

```
wipe,n1
```

The frame is changed to `n1` and the region is cleared. The permissible values for `n1` are:

n1	Region used
0	Entire window used
1	Upper left quadrant
2	Upper right quadrant
3	Lower left quadrant
4	Lower right quadrant
	(Default: n1 = 0)

Some graphics terminals do not support the feature of erasing only part of the screen; in these cases the entire screen may be erased instead of only the part specified.

```
zoom
```

Restore current plot view to entire mesh. The limits for parts of the plot region to be displayed may be selected using the `PICK` view command.

Appendix F

Program changes

FEAP has undergone several changes since the release of Version 8.4. A summary for some of the changes is given below:

- A large number of diagnostics concerning wrong input or missing input has been added to assist users in getting a correct input of data.
- The input of nodal mesh data may be restricted to specific material groups or regions using the `FILTER` command. When a specific *material* or *region* is set then only the nodes belonging to it may will have loads, displacements, boundary conditions, etc. assigned by their mesh input commands.
- An option to extract the boundary mesh using the `EXTRact` solution command has been added. This allows for the *smooth* surface parts of a two or three dimensional mesh to be automatically detected and written to a file. These may be used subsequently to define behavior of a surface segment.
- A new command `*AUTo` has been added that automatically increments element numbers specified in different `ELEMent`, `BLOCk` or `BLENd` command sets. This is particularly useful when used in connection with the `EXTRaction` segment data.
- Both the real and imaginary parts of forces and displacements may be input for problems solved in a complex arithmetic mode.
- Structure vectors for fiber models may be defined for individual elements using the mesh `STRUcture` command.
- Printed value of energy and momenta may be written to the output file using the `ENERgy` solution command.
- The `ELEMent` command may have parameters defining the number of nodes in the group along with other options. This avoids need for multi-record input of

elements with less than 13 nodes. Additional parameters allow for a reset of the material set number for the entire group; setting a part number, and also setting an optional type value.

- Output of data sets for use with *Paraview* is now a part of the standard program.
- New options of splitting the graphics screen into two parts either horizontal or vertical using the plot `HFRAme` or `VFRAme` command are available.
- In multi-frame plots a caption of the plot type is placed under the graphics view.
- Projection of element variables may be either continuous or discontinuous between different material sets may be set using the plot `PROJection` command.
- Three dimensional objects in perspective view may be rotated about coordinate axes by a specified angle in degrees using the plot `ROTAte` command.
- A new set of elements for acoustic analysis (type `ACOUstic`) is added along with a Lysmer-Kuhlmeyer treatment for infinite regions.[59]
- An element to provide uniform elastic and/or viscous boundary support in all directions has been added. The element is a form of a *Robin* boundary condtion for surface traction on solid elements. A restricted form for elastic boundary support in the normal direction of the reference geometry is also added as a *Winkler* condition.
- The number of user macro modules has been increased to 50.

In addition many other small changes have been made to correct bugs or improve options available. Due to the many combinations of commands that can be used there may remain many options which do not work properly. As always, please inform us of these or desired new features by sending comments by e-mail to:

`feap@ce.berkeley.edu`

Bibliography

[1] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, Oxford, 6^{th} edition, 2005.

[2] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method for Solid and Structural Mechanics*. Elsevier, Oxford, 6^{th} edition, 2005.

[3] O.C. Zienkiewicz, R.L. Taylor, and P. Nithiarasu. *The Finite Element Method for Fluid Dynamics*. Elsevier, Oxford, 6^{th} edition, 2005.

[4] K.-J. Bathe. *Finite Element Procedures*. Prentice Hall, Englewood Cliffs, N.J., 1996.

[5] J. Bonet and R.D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, Cambridge, 1997. ISBN 0–521–57272–X.

[6] R.D. Cook. *Finite Element Modeling for Stress Analysis*. John Wiley & Sons, New York, 1994.

[7] R.D. Cook, D.S. Malkus, M.E. Plesha, and R.J. Witt. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, New York, 4^{th} edition, 2009.

[8] M.A. Crisfield. *Non-linear Finite Element Analysis of Solids and Structures*, volume 1. John Wiley & Sons, Chichester, 1991.

[9] M.A. Crisfield. *Non-linear Finite Element Analysis of Solids and Structures*, volume 2. John Wiley & Sons, Chichester, 1997.

[10] A. Curnier. *Computational Methods in Solid Mechanics*. Kluwer Academic Publishers, Dordrecht, 1993.

[11] T.J.R. Hughes. *The Finite Element Method: Linear Static and Dynamic Analysis*. Prentice-Hall, Englewood Cliffs, N.J., 1987.

[12] R.L. Taylor and S. Govindjee. *FEAP - A Finite Element Analysis Program, Programmer Manual*. University of California, Berkeley. http://projects.ce.berkeley.edu/feap.

[13] R.L. Taylor and S. Govindjee. *FEAP - A Finite Element Analysis Program, Installation Manual*. University of California, Berkeley. http://projects.ce.berkeley.edu/feap.

[14] R.L. Taylor and S. Govindjee. *FEAP - A Finite Element Analysis Program, Parallel User Manual*. University of California, Berkeley. http://projects.ce.berkeley.edu/feap.

[15] R.L. Taylor and S. Govindjee. *FEAP - A Finite Element Analysis Program, Multiscale User Manual*. University of California, Berkeley. http://projects.ce.berkeley.edu/feap.

[16] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, Oxford, 7^{th} edition, 2013.

[17] O.C. Zienkiewicz, R.L. Taylor, and D. Fox. *The Finite Element Method for Solid and Structural Mechanics*. Elsevier, Oxford, 7^{th} edition, 2013.

[18] O.C. Zienkiewicz, R.L. Taylor, and P. Nithiarasu. *The Finite Element Method for Fluid Dynamics*. Elsevier, Oxford, 7^{th} edition, 2014.

[19] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method: The Basis*, volume 1. Butterworth-Heinemann, Oxford, 5^{th} edition, 2000.

[20] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method*, volume 1. McGraw-Hill, London, 4^{th} edition, 1989.

[21] V. Kouznetsova, W.A.M. Brekelmans, and F.P.T. Baaijens. An approach to micro-macro modelling of heterogeneous materials. *Computational Mechanics*, 27:37–48, 2001.

[22] V.G. Kouznetsova. *Computational homogenization for the multi-scale analysis of multi-phase materials*. Phd. thesis, Eindhoven University of Technology, Eindhoven, Netherlands, 2002.

[23] J.H. Irving and J.G. Kirkwood. The statistical mechanical theory of transport processes. IV. *The Journal of Chemical Physics*, 18:817–829, 1950.

[24] B.S. Mercer, K.K. Mandadapu, and P. Papadopoulos. Novel formulations of microscopic boundary-value problems in continuous multiscale finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 286:268–292, 2015.

[25] R.L. Taylor. *FEAP - A Finite Element Analysis Program, Theory Manual*. University of California, Berkeley. http://projects.ce.berkeley.edu/feap.

[26] R.L. Taylor and R. Iding. Applications of extended variational principles to finite element analysis. In C.A. Brebbia and H. Tottenham, editors, *Proc. of the International Conference on Variational Methods in Engineering*, volume II, pages 2/54–2/67. Southampton University Press, 1973.

[27] J.C. Simo and L. Vu-Quoc. A three-dimensional finite strain rod model. Part II: Geometric and computational aspects. *Computer Methods in Applied Mechanics and Engineering*, 58:79–116, 1986.

[28] J.C. Simo and L. Vu-Quoc. On the dynamics in space of rods undergoing large motions - a geometrically exact approach. *Computer Methods in Applied Mechanics and Engineering*, 66:125–161, 1988.

[29] M.A. Crisfield and G. Jelenic. Objectivity of strain measures in geometrically exact 3d beam theory and its finite element implementation. *Proceedings of the Royal Society of London*, 455:1125–1147, 1998.

[30] G. Jelenic and M.A. Crisfield. Geometrically exact 3d beam theory: implementation of a strain-invariant finite element for statics and dynamics. *Computer Methods in Applied Mechanics and Engineering*, 171:141–171, 1999.

[31] A. Ibrahimbegovic and M. Al Mikdad. Finite rotations in dynamics of beams and implicit time-stepping schemes. *International Journal for Numerical Methods in Engineering*, 41:781–814, 1998.

[32] J.C. Simo, N. Tarnow, and M. Doblare. Non-linear dynamics of three-dimensional rods: Exact energy and momentum conserving algorithms. *International Journal for Numerical Methods in Engineering*, 38:1431–1473, 1995.

[33] J.C. Simo and N. Tarnow. On a stress resultant geometrically exact shell model. Part VI 5/6 dof treatments. *International Journal for Numerical Methods in Engineering*, 34:117–164, 1992.

[34] W. Voigt. *Lehrbuch der Kristallphysik*. Teubner, Leipzig-Berlin, 1910.

[35] L. Anand and S. Govindjee. *Continuum Mechanics of Solids*. Oxford University Press, Oxford, 2020.

[36] DS Simulia. *Abaqus Theory Manual*.

[37] R.M. Christensen. *Theory of Viscoelasticity: An Introduction*. Academic Press, New York, 1971 (Reprinted 1991).

[38] G.A. Holzapfel. *Nonlinear Solid Mechanics. A Continuum Approach for Engineering*. John Wiley & Sons, Chichester, 2000.

[39] J.C. Simo and N. Tarnow. The discrete energy-momentum method. conserving algorithm for nonlinear elastodynamics. *Zeitschrift für Mathematik und Physik*, 43:757–793, 1992.

[40] J.C. Simo and O. González. Recent results on the numerical integration of infinite dimensional hamiltonian systems. In T.R.J. Hughes, E. Oñate, and O.C.Zienkiewicz, editors, *Recent Developments in Finite Element Analysis*. CIMNE, Barcelona, Spain, 1994.

[41] J.C. Simo and T.J.R. Hughes. *Computational Inelasticity*, volume 7 of *Interdisciplinary Applied Mathematics*. Springer-Verlag, Berlin, 1998.

[42] J.C. Simo and R.L. Taylor. Quasi-incompressible finite elasticity in principal stretches: Continuum basis and numerical algorithms. *Computer Methods in Applied Mechanics and Engineering*, 85:273–310, 1991.

[43] R.W. Ogden. *Non-linear Elastic Deformations*. Ellis Horwood, Limited (reprinted by Dover, 1997), Chichester, England, 1984.

[44] K.C. Valanis and R.F. Landel. The strain-energy function of a hyperelastic material in terms of the extension ratios. *Journal of Applied Physics*, 38:2997–3002, 1967.

[45] E.M. Arruda and M.C. Boyce. A three-dimensional constitutive model for the large stretch behavior of rubber elastic materials. *Journal of the Mechanics and Physics of Solids*, 41:389–412, 1993.

[46] O.H. Yeoh. Characterization of elastic properties of carbon-black filled rubber vulcanizates. *Rubber Chemistry and Technology*, 63:792–805, 1990.

[47] R.L. Taylor, K.S. Pister, and G.L. Goudreau. Thermomechanical analysis of viscoelastic solids. *International Journal for Numerical Methods in Engineering*, 2:45–79, 1970.

[48] J.C. Simo. On a fully three-dimensional finite-strain viscoelastic damage model: Formulation and computational aspects. *Computer Methods in Applied Mechanics and Engineering*, 60:153–173, 1987.

[49] J.C. Simo and R.L. Taylor. Consistent tangent operators for rate-independent elastoplasticity. *Computer Methods in Applied Mechanics and Engineering*, 48:101–118, 1985.

[50] J.C. Simo and R.L. Taylor. A return mapping algorithm for plane stress elastoplasticity. *International Journal for Numerical Methods in Engineering*, 22:649–670, 1986.

[51] R. Hill. A theory of the yielding and flow of anistropic metals. *Proceedings of the Royal Society of London*, 193:281–297, 1948.

[52] P. Papadopoulos. On a class of higher order pseudo rigid bodies. *Mathematics and Mechanics of Solids*, 6:631–640, 2001.

[53] F. Auricchio and R.L. Taylor. Two material models for cyclic plasticity: Nonlinear kinematic hardening and generalized plasticity. *Int. J. Plasticity*, 11:65–98, 1995.

[54] F. De Angelis and R.L. Taylor. Numerical implementation of a generalized plasticity model at finite strains. In *Computational Plasticity XI: Fundamentals and applicaitons*, pages 1458–1465, Barcelona, Spain, 2011. CIMNE.

[55] J. Helfenstein, M. Jabareen, E. Maza, and S. Govindjee. On non-physical response in models for fiber-reinforced hyperelastic materials. *International Journal of Solids and Structures*, 47:2056–2061, 2010.

[56] G.A. Holzapfel, T.C. Gasser, and R.W. Ogden. A new constitutive framework for arterial wall mechanics and a comparative study of material models. *Journal of Elasticity*, 61:1–48, 2000.

[57] G.A. Holzapfel, T.C. Gasser, and R.W. Ogden. Comparison of a multi-layer structural model for arterial walls with a fung-type model, and issues of material stability. *Journal of Biomechanical Engineering*, 126:264–275, 2004.

[58] J.A. Weiss, B.N. Maker, and S. Govindjee. Finite element implementation fo incompressible, transversely isotropic hyperelasticity. *Computer Methods in Applied Mechanics and Engineering*, 135:107–128, 1996.

[59] J. Lysmer and R.L. Kuhlemeyer. Finite dynamic model for infinite media. *J. Eng. Mech, ASCE*, EM 4:859–877, 1969.

[60] E. Moireau, N.Xiao, M. Astorino, C.A. Figueroa, D. Chapelle, C.A. Taylor, and J.F. Gerbeau. External tissue support and fluid-structure simulation in blood flows. *Biomech Model Mechanobiol*, 11:1–18, 2012.

[61] MATLAB. www.mathworks.com, 2012.

[62] M.G.D. Geers, V. Kouznetsova, and W.A.M. Brekelmans. Gradient-enhanced computational homogenisation for the micro-macro scale transition. *Journal de Physique IV*, 11(5):5145–5152, 2001.

[63] H. Matthies and G. Strang. The solution of nonlinear finite element equations. *International Journal for Numerical Methods in Engineering*, 14:1613–1626, 1979.

[64] R. Lehoucq, K. Maschhoff, D. Sorensen, and C. Yang. ARPACK: Arnoldi/lanczos package for eigensolutions. http://www.caam.rice.edu/software/ARPACK/.

[65] N. Newmark. A method of computation for structural dynamics. *Journal Engineering Mechanics, ASCE*, 85:67–94, 1959.

[66] M. Géeradin and D.J. Rixen. *Mechanical Vibrations: Theory and Applications to Structural Dynamics*. John Wiley and Sons Ltd., Chichester, U.K., 3^{rd} edition, 2015.

[67] H. Hilber, T.J.R. Hughes, and R.L. Taylor. Improved numerical dissipation for the time integration algorithms in structural dynamics. *Earthquake Engineering and Structural Dynamics*, 5:283–292, 1977.

[68] J.C. Simo and N. Tarnow. Exact energy-momentum conserving algorithms and symplectic schemes for nonlinear dynamics. *Computer Methods in Applied Mechanics and Engineering*, 100:63–116, 1992.

[69] O. González. *Design and analysis of conserving integrators for nonlinear Hamiltonian systems with symmetry*. Ph.D thesis, Department of Mechanical Engineering, Stanford University, Stanford, California, 1996.

[70] A.K. Chopra. *Dynamics of Structures*. Prentice-Hall, Upper Saddle River, N.J., 1995.

[71] S. Govindjee, J. Strain, T.J. Mitchell, and R.L. Taylor. Convergence of an efficient local least-squares fitting method for bases with compact support. *Computer Methods in Applied Mechanics and Engineering*, 213–216:84–92, 2012. http://dx.doi.org/10.1016/j.cma.2011.11.017.

Index