FEAP - - A Finite Element Analysis Program

Version 7.5 User Manual

Robert L. Taylor
Department of Civil and Environmental Engineering
University of California at Berkeley
Berkeley, California 94720-1710
E-Mail: rlt@ce.berkeley.edu

February 2005

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 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]

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*, 5th edition, by O.C. Zienkiewicz and R.L. Taylor [1, 2, 3] or the 4th edition [13, 14]) 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:

1. Use in course instruction to illustrate performance of different types of elements and modeling methods;
2. 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 or a Windows environment and includes an integrated set of modules to perform:

1. Input of data describing a finite element model;
2. Construction of solution algorithms to address a wide range of applications; and
3. Graphical and numerical output of solution results.

The problem solution step 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 specific needs. There are sufficient commands included in the system for 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 general element library. Elements are available to model one, two or three dimensional problems in linear and/or non-linear structural and solid mechanics and for linear heat conduction problems. Each solid element accesses a material model library. Material models are provided for elasticity, viscoelasticity, plasticity, and heat transfer constitutive equations. Elements also provide capability to generate mass and 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: www.ce.berkeley.edu/~rlt/feap).

This manual describes how to use 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 the input file is given as:

```
FEAP * * Start record and title
  ...
Control and mesh description data
  ...
END
  ...

Solution and graphics commands
  ...
STOP
```

The *FEAP Example Manual* may be consulted for examples on use of some input and solution options described in this report (see: www.ce.berkeley.edu/~rlt/feap).

1.1 Example: A simple truss

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 shall assume that all members are elastic with the same elastic modulus and cross-sectional area.

A complete input file to solve this problem is shown in Table 1.1. The first two lines of the file are called the *control information* and describe the start record followed by

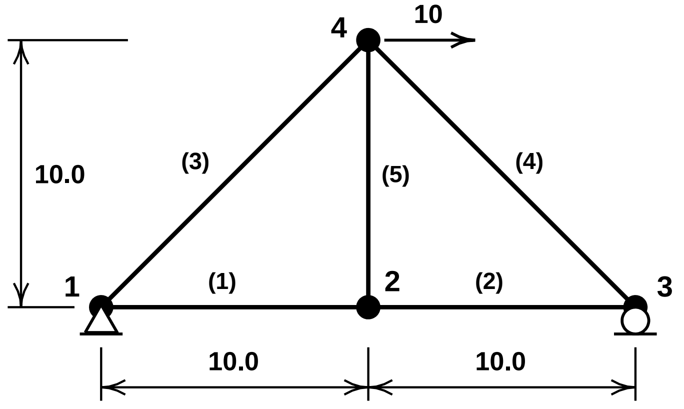


Figure 1.1: King-post truss example. ● = Nodes; (n) = Element n

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

MATErial 1
  TRUSS
    ELAStic isotropic 10000.0
    CROSs   section   0.25

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

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

BOUNdary restraints
    1    0     1    1
    3    0     0    1

FORCe
    4    0    10.   0.

END

BATCh
  TANGent
  FORM
  SOLV
  DISPlacement all
  STREss all
END
STOP
```

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

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. These records are followed by data sets which describe material and geometric properties; the coordinates for each node; nodal connections and material set identifier for each element; and the boundary restraint and load descriptions. 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 a solution algorithm, and here only information needed to perform a steady state (static) analysis are used. The second `END` statement informs the program that this set of `BATCh` commands is complete (multiple sets of `BATCh` or `INTEractive` solution sets are permitted. The final `STOP` record informs *FEAP* that all data has been processed and execution ceases.

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.

1.2 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 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 basic solution algorithms to solve problems]; and

11. Plot features contained within the program (Chapter 15).

The various options and parameters for each command to describe mesh input, problem solution, and plotting are included in the appendices to this manual. As noted previously, 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 available at the web site `www.ce.berkeley.edu/~rlt/feap`.

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: (a) the type of elements to use, (b) where to place nodes, (c) how to apply the loading and boundary restraints, (d) the appropriate material model and parameters values for each element, and (e) 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. The steps to define a mesh for *FEAP* are contained in Chapters 5 to 11. Each command available to define the 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 `END` mesh input command and 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 also appearing after the mesh `END` command.

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 to a detailed transient non-linear 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. 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 report. Users may add their own solution commands as described in the *Programmers Manual*.

2.1 Execution of FEAP and Input/Output Files.

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

1. A command line input from a 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 a 'pop-up' window will appear which needs to be traversed to the location of the directory containing the desired 'input file' to be executed. The file may be sellected using the mouse in a standard Windows manner.

   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 an X11-window.

   In a first execution of the program in each directory it is necessary to provide a name for the file containing the 'input data'. Default names will be provided for the other files but may be chanced by the user if desired.

2. 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. 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 names for all other files is selected by the program and may not be changed.

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

Upon a successful first execution of the program a file `feapname` will be written to disk in the solution directory to preserve the name for each of the input and output file names. If it is desired to reinstall the program the `feapname` file may be deleted and the `FEAP` command then reissued.

For each subsequent execution of the program using a `FEAP` command line input, the user receives 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). An indicated default filename may be accepted by pressing the return (enter) key without specifying any new data. 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 the steps needed to create 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 read file: Rfile is filename
s = restart save file: Sfile is filename
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`.

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

[3]This form of the solution command **must** be given from a ‘Command Prompt’ window

```
     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'!*

2.2 Modification of Default Options

At the time that the executable version of *FEAP* is created default values for several parameters may be set in the main program file `feap75.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
	—		Makes Grayscale PostScript files with normal order.
helplev	basic		Default level for commands Same as: MANU,0
	interm		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!
increment	value		Set increment value change to force reduction in array size.

Table 2.1: Options for Changing Default Parameters

Chapter 3

ELEMENTS TYPES

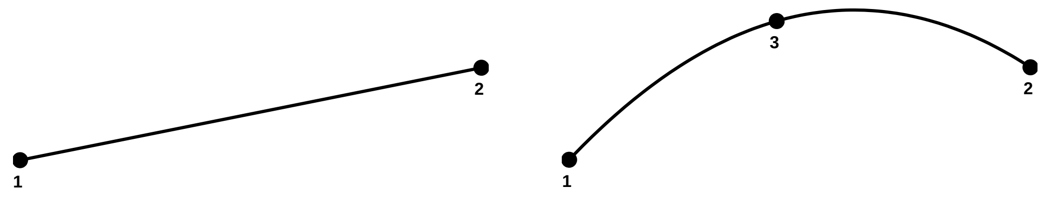
The description of *elements* in *FEAP* is expressed as a set of *node* numbers which describe the connectivity. Elements may have a topology of a line, a surface or a solid. In *FEAP* the nodes for each element are generally associated with the unknown parameters of the problem. To describe a problem 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.

3.1 Line Elements

Line elements are defined by 2 or more nodes and the types included in the *FEAP* library are shown in Fig. 3.1. Numbers shown with the elements describe the ordering that connectivity is to be specified on each data record for an element. While the element library included with the *FEAP* system has only 2 or 3 node elements those with more may be added by a user.

Two node elements may be used to describe *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* or *thermal convection surface conditions* for heat conduction problems.

Elements with the minimum number of nodes are called a *simplex* and those with more nodes are of *higher order*. An advantage of higher order elements 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.



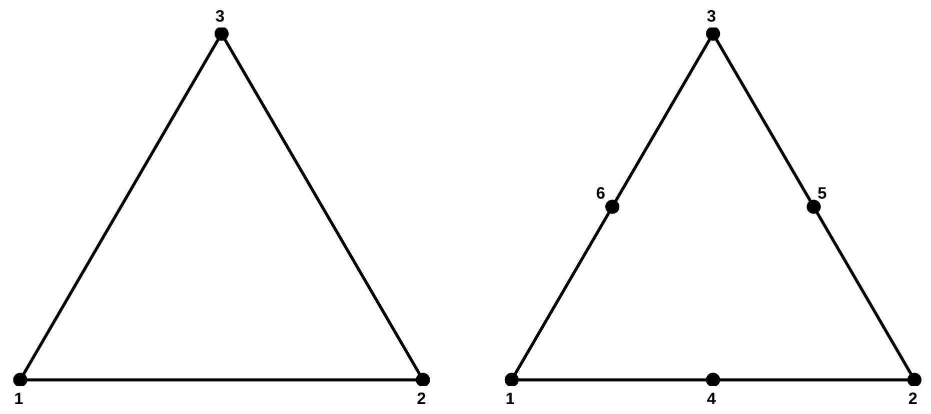
2-Node Element 3-Node Element

Figure 3.1: Line type elements in *FEAP* library

3.2 Surface Elements

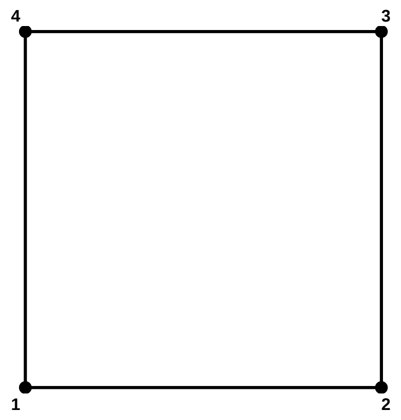
Surface finite elements are generally described by triangular or quadrilateral shapes. Triangular elements included with the system may be described by the 3-node simplex or by the 6-node element shown in Fig. 3.2. A 6-node element may have curved sides, as shown for the 3-node line element in Fig. 3.1. For most element formulations this is accomplished using an *isoparametric mapping* as described in standard reference books on finite elements [1].

Surface elements may also be of quadrilateral shape as shown in Fig. 3.3 The basic element has 4-nodes and can be mapped using isoparametric concepts into a general shape quadrilateral 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]. These elements are of higher order and may have curved sides when mapped using the isoparametric concept. In general, it is preferable to use the 9 and 16

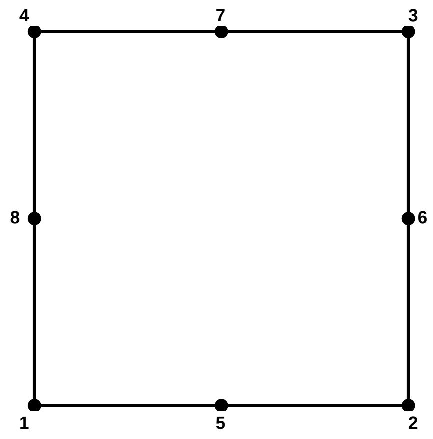


3-Node Simplex 6-Node Element

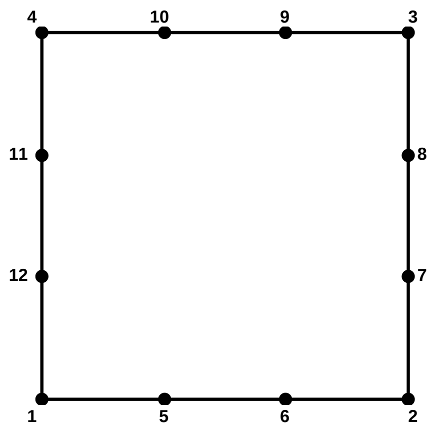
Figure 3.2: Triangular surface type elements in *FEAP* library



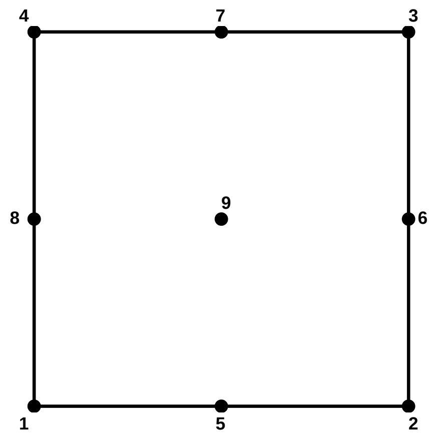
4-Node Element



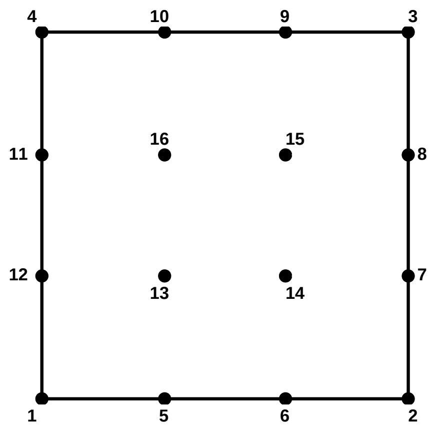
8-Node Element



12-Node Element



9-Node Element



16-Node Element

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

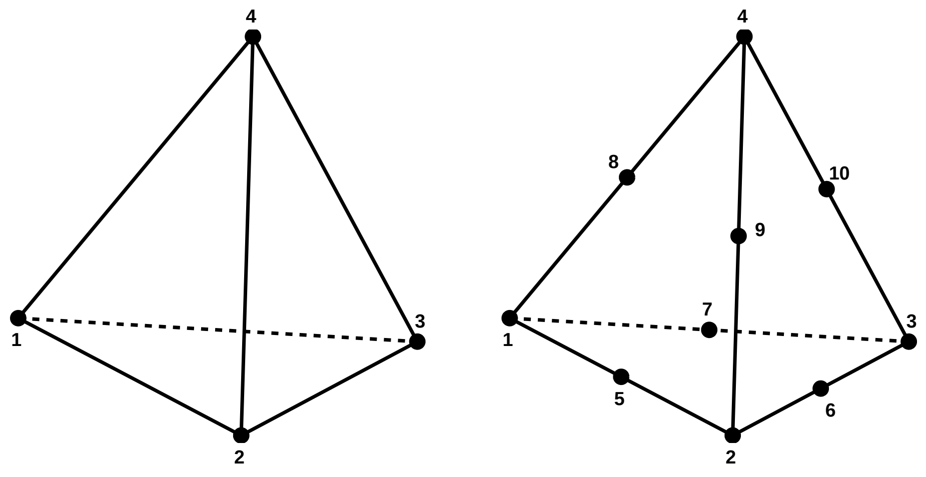
node elements rather than those with 8 or 12 nodes. This is especially true if problems in solid or fluid mechanics are solved in which *near incompressibility* conditions can exist. Also, when *lumped* or *diagonal* mass matrices are used in transient situations the properties of those for Lagrangian 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]

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, currently the element shape is restricted to a 3-node triangle or a 4-node quadrilateral.

3.3 Solid Elements

Solid elements included in the *FEAP* library may be of tetrahedral or brick shape. The simplex element is a 4-node tetrahedron and the first higher order element a 10-node tetrahedron as shown in Fig. 3.4 and may have curved edges and faces when mapped using an isoparametric concept.

Solid elements may also have a *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 or 27-nodes. The 20-node element is a member of the Serendipity family with the 27-node element belonging to the Lagrangian family. Figure 3.5 shows these two types



4-Node Simplex 10-Node Element

Figure 3.4: Tetrahedron solid type elements in *FEAP* library

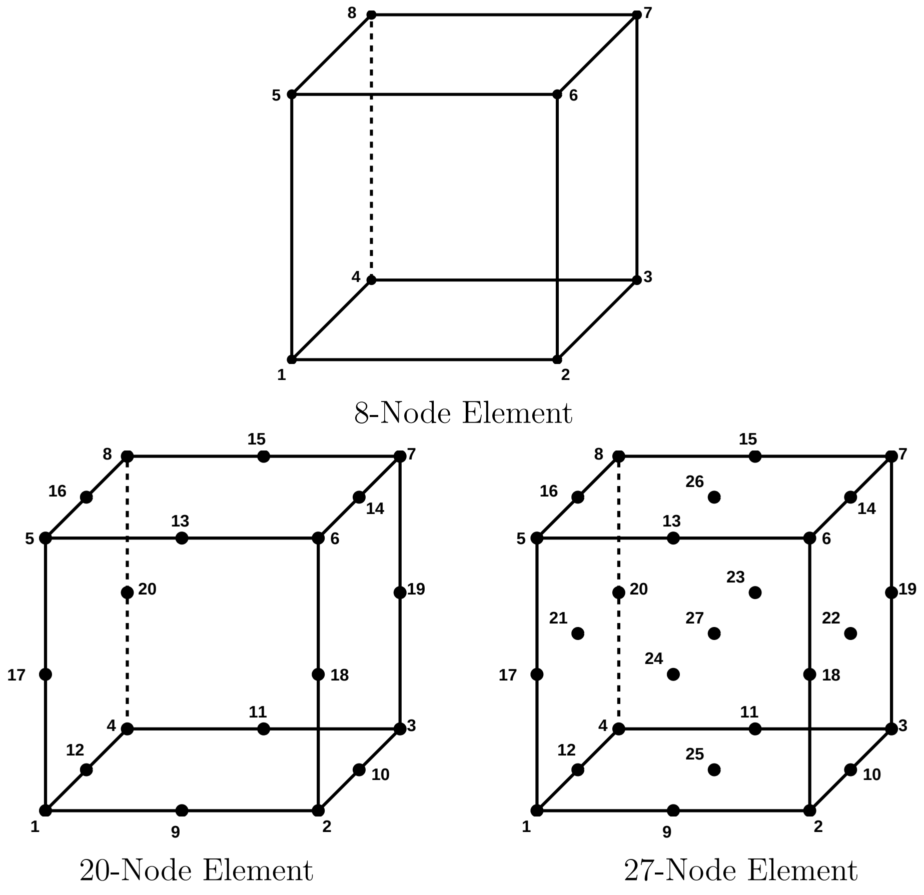


Figure 3.5: Brick solid type elements in *FEAP* library

of elements.

Solid elements are used in *FEAP* to model general three-dimensional problems in solid mechanics and heat conduction. Solid mechanics elements are available based on displacement, mixed, and enhanced strain formulations.[1] All element types permit use of 8-node brick elements. Displacement and mixed formulations permit use of 27-node and 64-node brick elements of Lagrangian type. Displacement formulation also permits use of 4-node or 10-node tetrahedra.

Except for the 20-node brick element, shape functions for Serendipity type elements are not provided with the *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.

Chapter 4

INPUT RECORD SPECIFICATION

Data input specifications in *FEAP* consist of records which may contain from 1 to 255 characters of information in free format form. Each record can contain up to 16 alphanumeric data items. The maximum field width for any single data item is 15 characters (14 characters of data and 1 character for separating fields). 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, the first four letters of each text command are shown in upper case letters, while the remainder are in lower case. It is only necessary to give the first four characters for each text command; however, additional ones 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).

Each input record may be in the form of text and/or numerical constants, parameters, or expressions. Text fields all start with the letters `a` through `z` (either upper or lower case may be used, however, internally *FEAP* converts 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 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.

4.1 Constants

Constants may be represented as integers or floating point numbers. Integers are specified without a decimal point as `1`, `-10`, 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). In particular, the above two examples will yield data input values of `4.0` and `-6.456`, respectively.

4.2 Parameters

The use of parameters will simplify the data input required to define steps in 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 definition of parameters
```

Blanks are permitted and are ignored in the processing of a record (except in expressions). Once a parameter is defined it may be used in place of any constant in the 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

```
1./4. + 4
```

is evaluated as `4.25`, whereas

```
1./(4. + 4)
```

is evaluated as `0.125`.

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. Expressions may appear in any location in place of a constant or a parameter. Accordingly, a force may be assigned as

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

Additionally, node and element numbers could also appear as expressions; however, the use of the `*NODe` 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,   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*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.

Chapter 5

MESH INPUT DATA SPECIFICATION

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

5.1 Start of Problem and Control Information

The first part of an input data file contains the *control data* which consists of two records:[1]

1. 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).
2. The second record contains problem size information with *required* data consisting of:[2]
   (a) `NUMNP` - Number of nodal points;
   (b) `NUMEL` - Number of elements;
   (c) `NUMMAT` - Number of material property sets;
   (d) `NDM` - Space dimension of mesh;
   (e) `NDF` - Maximum number of unknowns per node; and

[1]Parameter definitions may precede the control data and used in defining the size values.

[2]Do not place data beyond `NEN` as additional fields exist on the control record for advanced features. See Appendix A for details.

(f) `NEN` - Maximum number of nodes per element.

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 codes 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.

The use of the automatic determination of data requires the mesh data to be read twice: Once to do counts and once to perform actual inputs of 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.

We next consider commands used to describe the remainder of the finite element mesh. 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 before or after 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 `END` command. Thus, information will not necessarily appear in the output file in the same order that data is placed in the input file.

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 Nodal Coordinate and Element Connections

The basic mesh for *FEAP* consists of nodes and elements. For the general 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 provided in *FEAP* compute 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.

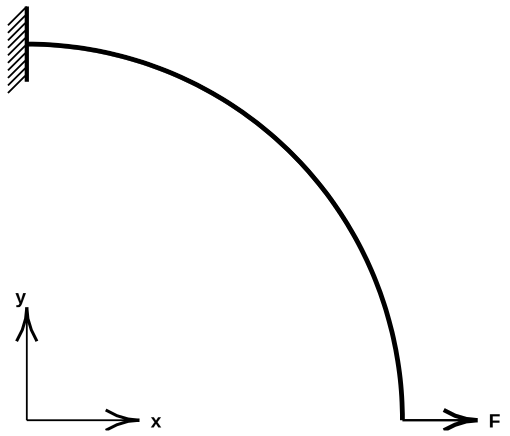


Figure 5.1: Curved Beam

5.2.1 The COORdinate Command

The coordinates for nodes may be specified using the `COOR`dinate command. The individual records defining each node and its coordinates are specified after the `COOR`dinate command 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) first their polar coordinate form given by:

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

and (2) their conversion from polar to Cartesian form using the `POLA`r 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`. Linear interpolation of coordinates is used to define the intermediate values for the generated nodes. If `MG` 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 5.2.4 below) or any combination of the commands.

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

```
COORdinate ALL
  1 0  X_1 Y_1 Z_1
      ..
```

should be used. 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 coordinate (and element) values are created by an external mesh generation program.

5.2.2 The ELEMent Command

The `ELEMent` command may be used to input the list of nodes connected to an individual element. For elements where the maximum number of nodes is less or equal to 13 (i.e., the `NEN` parameter on the control record), 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 each 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. Note that when some elements have fewer nodes needed to define the connection list it is still necessary that all element descriptions still have the same number of records.

The element numbers following each `ELEM`ent 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:

```
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.

Element data for the mesh 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.

Element data is preprocessed to determine the total number of elements `NUMEL` in a mesh. Element data may also be defined using the `BLOC`k and `BLEN`d commands.

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

```
ELEMent ALL
  1 0  M_1 N_1 N_2 ...
      ..
```

should be used. 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.

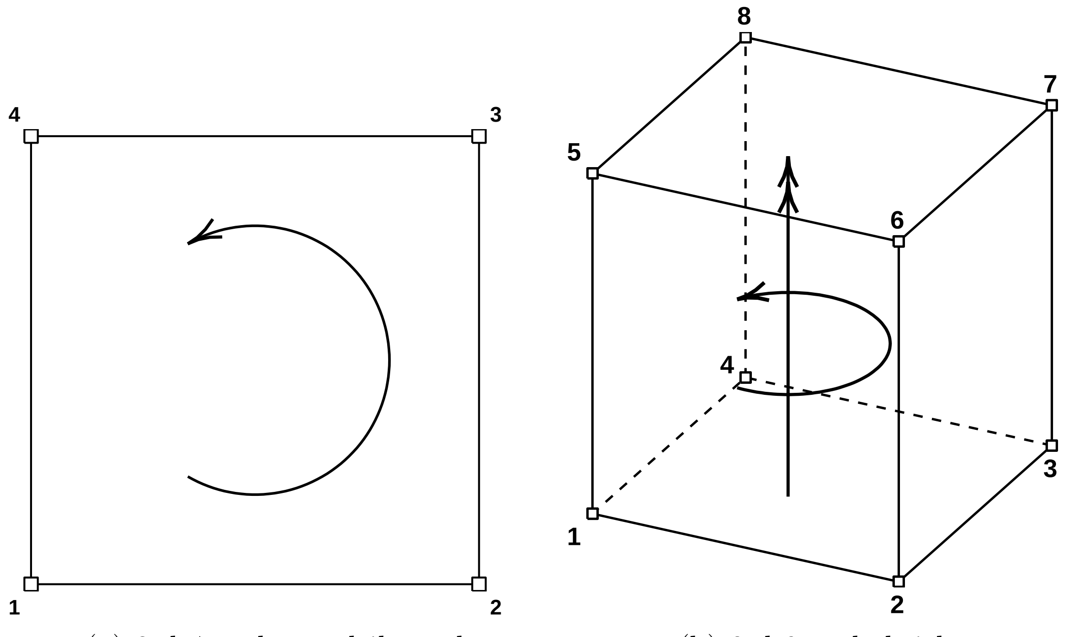
5.2.3 The BLOCk Command

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); triangles and quadrilaterals for surface elements and three dimensional hexahedra (brick) or tetrahedra for solid element types.

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

```
BLOCk
  ctype,r-inc,,node1,elmt1,mat,r-skip
  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



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

Figure 5.2: Right hand rule numbering of element nodes

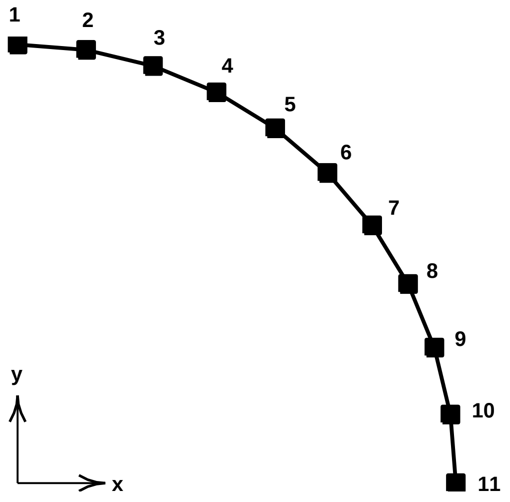


Figure 5.3: Mesh for Curved Beam. 10 Elements

`CARTesian` (default), `POLAr` 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 Chapter 3.1.

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
  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 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
  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.

The parameters of the `BLOCk` command are defined as:

Type	- Master node coordinate type (`CART, POLA,` 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).

		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)

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 1 0 0 1
  1   5.0   90.0
  2   5.0    0.0
           ! Termination record
```

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 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 the POLAr option 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 The BLENd Command

A block of nodes and elements also may be generated using a blending function approach (e.g., see [1], pp 226 or [13], 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:

1. `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.

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

   $$\begin{aligned} r(\xi) &= \sum_i L_i(\xi) r_{Vi} \\ \theta(\xi) &= \sum_i L_i(\xi) \theta_{Vi} \end{aligned}$$

   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.

3. `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).

Blending: BLENd Command

For two-dimensional blended meshes the `SURF`ace 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
    s1 s2 s3 s4
```

where the parameters are defined in Table 5.1.

Type	- Blend type (SURFace).
1-inc	- Number of nodal increments to be generated along 1-2 edge.
2-inc	- 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 5.1: Surface Blend Parameters

The two dimensional blended mesh shown in Figure 5.4 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 5.5.

For three-dimensional blended meshes either the `SURF`ace or the `SOLI`d option may be used to generate the mesh region. For the `SURF`ace 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 `SOLI`d option the vertex order is identical to that for the 8-node `BLOC`k 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
    s1 s2 s3 s4 s5 s6 s7 s8
```

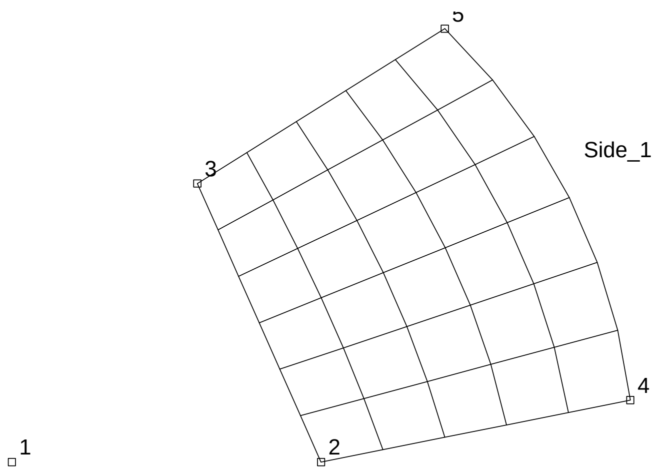


Figure 5.4: Two-dimensional Blended Mesh

```
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  0  0  1
   2  4  5  3
            ! Blank termination record
```

Figure 5.5: Two-dimensional blended mesh data

where the parameters are defined in Table 5.2.

A blended region for a three dimensional mesh is shown in Figure 5.6 and generated using the data shown in Figure 5.7.

Nodes and elements may be generated using a combination of the above schemes. Thus, it is possible to mix the `BLOC`k and `BLEN`d options with the `COOR`dinate and `ELEM`ent 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 `BLOC`k and `BLEN`d options simultaneously) converting the input and/or generated coordinates to Cartesian coordinate values using the `POLA`r or `SPHE`rical commands.

type	- Blend type `SOLId`.
1-inc	- Number of nodal increments to be generated along 1-2 edge.
2-inc	- Number of nodal increments to be generated along 2-3 edge.
3-inc	- 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 5.2: Three-dimensional Solid Blend Parameters

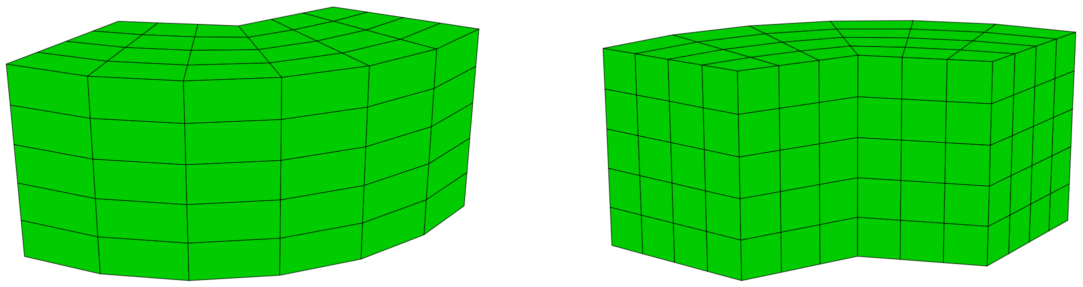


Figure 5.6: Three-dimensional Blended Mesh

5.3 Coordinate and Transformation Systems

The coordinates in *FEAP* must all be given in a Cartesian system. Input data, however, may be specified in *Cartesian*, *polar* (which in three dimensions is interpreted as cylindrical coordinates), or *spherical* coordinate systems.

5.3.1 POLAr, SPHErical and SHIFt Commands

When polar or spherical coordinates are used to define the nodal data using the `COOR`-dinate command, they may be transformed to the required Cartesian form using the

```
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
   2  3  6  4  8  9 12 10
            ! Blank termination record
```

Figure 5.7: Three-dimensional blended mesh data

`POLA`r or `SPHE`rical commands, respectively. Nodal coordinates generated with polar or spherical options in the `BLOC`k 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

$$\begin{aligned} x_1 &= x_0 + r \cos \theta \\ x_2 &= y_0 + r \sin \theta \end{aligned}$$

and

$$x_3 = z_0 + z$$

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 `SHIF`t command given as

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

By default x_0, y_0, z_0 are zero.

The `SPHE`rical command is similar to the `POLA`r command. The input records are specified as:

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

Transformations use the relations

$$\begin{aligned} x_1 &= x_0 + r \cos \theta \sin \phi \\ x_2 &= y_0 + r \sin \theta \sin \phi \end{aligned}$$

and

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

This is followed by use of the command set

```
SPHErical
  NODE n1 n2 inc
```

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

5.3.2 Coordinate Transformation

Cartesian systems may be translated, stretched, reflected and/or rotated using the `TRAN`sform 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 `TRAN`sform 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 resequences the node numbers on elements to maintain positive Jacobians (provided the original data is correct in its local Cartesian basis - $\hat{x}_i$).

5.4 Nodal Boundary Condition Inputs

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 both be input at each node. It is not necessary to input conditions for any node where all the components are zero. The actual condition to be imposed (i.e., force or displacement) is determined by the active values of the *boundary restraint conditions*. 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 the directions in the global Cartesian coordinates describing a mesh. It is possible, however, to specify components which are associated with directions different than the global coordinate ones. At present, the only option is a set of coordinates which are described by a rotation angle 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.

5.4.1 Basic input form.

The basic options to input the nodal quantities associated with boundary conditions are shown in Table 11.1. The use of a basic form (i.e., `BOUNdary, FORCe, DISPlacement, ANGLe`) implies a specification using a *node number*. The other options do not require node numbers and are preferred when possible.

Type	Boundary	Forces	Displacements	Angle
Nodal	`BOUNdary`	`FORCe`	`DISPlacement`	`ANGLe`
Edge	`EBOUndary`	`EFORce`	`EDISplacement`	`EANGle`
Coordinate	`CBOUndary`	`CFORce`	`CDISplacement`	`CANGle`

Table 5.3: 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. The input records for basic `FORC`e, `DISP`lacement, `BOUN`dary condition and `ANGL`e commands are similar to those for `COOR`inates 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 basic 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 a non-zero displacement at a node may be given using the command

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

This keeps the 1-component to zero and assigns the 2-component a valus of -0.1 units.

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 use different force/displacement combinations.

To use the basic input option it is a users responsibility to determine the correct number for each node - often 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.

5.4.2 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.

5.4.3 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 5.5).

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

5.4.4 Hierarchy of input forms.

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

Data input by the edge option is interpreted before any coordinate specified data. 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.

5.4.5 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 basic, 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 discrete mass term associated with an MPRO command will be computed as:

$$\boldsymbol{M}_{nn}\,(\ddot{\boldsymbol{x}}_n - \boldsymbol{g}(\boldsymbol{x}_n)) \tag{5.1}$$

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

$$g_i(\boldsymbol{x}_n) = f_i \, prop_k(t) \quad \text{where } k = np_i(n) \ . \tag{5.2}$$

The factors f_i are specified using the `GROUnd` global command.

5.5 Surface Loading

FEAP uses the `CSUR`face command to specify distributed tractions and displacements on portions of two or three dimensional surfaces defined by interpolation patches. For two dimensional problems the command has the structure

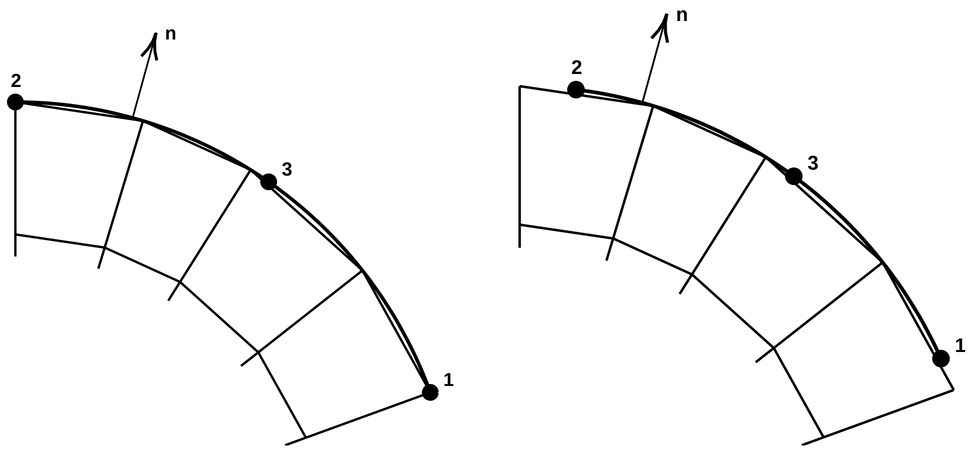
```
       CSURface
         type, data
         LINEar
         1,X_1,Y_1,P_1
         2,X_2,Y_2,P_2
                             ! blank termination record
```

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: `CART`esian; `POLA`r; `GAP`; `NORM`al traction; `TANG`ential traction; or `DISP`lacement pattern (default is normal traction). If the data type is `DISP`lacement the parameter `data` specifies the coordinate direction for the specified values. Multiple records of `type` may exist before input of interpolation patches and patterns.

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.



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

Figure 5.8: Two-Dimensional Surface Loading

FEAP will search for all nodes which are closer to the interpolation patch than `GAP` (default is 10^{-3}). 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 5.8 – 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 5.8(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.

For three dimensional problems the command has the structure

```
CSURface
```

```
    type, data
    SURFace
    1,X_1,Y_1,P_1
    2,X_2,Y_2,P_2
    3,X_3,Y_3,P_3
    4,X_4,Y_4,P_4
                          ! blank termination record
```

where `type` is the data type selected from: `GAP`; `NORM`al traction or `DISP`lacement pattern. No tangential option currently exists. Also, only those element surface facets which lie on or within the interpolation patch are selected. No partial facets are permitted.

The surface option may be used only for elements whose surface facets are either 3-node triangles or 4-node quadrilaterals.

5.6 Regions and Element Groups

The elements in *FEAP* may be assigned to different groups using the `REGI`on 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 whose nodes should be common but have different numbers (e.g., those defined using `BLOC`k 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 `ACTI`vate or `DEAC`tivate command language instructions (see Appendix B). Elements in Region zero may not be deactivated.

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.[15]

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, or elastoplastic 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 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. 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 node of a 1-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);

10. 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.

11. 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).

12. 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 for information on adding a user element.

The first two records of the MATErial set must be:

```
MATErial ma
  type unum mset doflist
```

where `ma` is the material set number, `type` is one of the above element types (e.g., solid, truss, etc.), `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), and `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 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).

Each element requires additional input data as described below.

6.1 Thermal Elements

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

$$\boldsymbol{\nabla}^T \left[\mathbf{K} \, \boldsymbol{\nabla} T \right] + Q = \rho \, c \, \frac{\partial T}{\partial t}$$

where $\mathbf{K}$ are (constant) thermal conductivity values, $\boldsymbol{\nabla}$ is the gradient operator, Q is heat added per volume, ρ is mass density and c is specific heat. The degree of freedom for each *node-α* is a temperature, T^α, and, by default, is assigned to the first degree of freedom. For one-dimensional problems each element may be a 2 node or a 3 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 solid elements and each element is a tetrahedron with 4 or 10 nodes or a hexahedron (brick) with 8 or 27 nodes (see Section 3.3).

The *thermal elements* are all based on a standard displacement (Galerkin) formulation. The material behavior for a thermal analysis is a *linear* Fourier model given by

$$\mathbf{q} = -\, \mathbf{K} \, \boldsymbol{\nabla} T$$

where $\mathbf{q}$ are *thermal flux* values. Both isotropic and orthotropic models are available. 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
```

where $K - i$ are the principal conductivities. 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 only (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 q-2
```

where `q-1` is the order used to compute arrays and `q-2` the order for printed outputs. Similarly, the mass matrix type is *consistent* by default. The command

```
MASS CONSistent
```

explicitly gives this option or alternatively a command

```
MASS LUMP
```

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

```
MASS type a
```

in which a mass is computed using

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

In some cases a better or smoother answer may be obtained using an interpolated form (e.g., see pp. 476 [1] and [16]).

6.2 Convection Elements

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

$$q_n = q_0 + H\,(T^n - T_0^n)$$

where q_0 is a constant prescribed flux; 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.

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

```
MATErial
  CONVection
    SURFace  H T-0
    FLUX     q-0
    EXPOnent n            ! Default is 1
    PLANe or AXISymmetrix ! Default is PLANe
```

where `H` = H; `T-0` = T_0; `q-0` = q_0. The `PLANe` or `AXISymmetric` options are used to set the geometry type of the boundary.

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

```
MATErial
  CONVection
    SURFace  H T_0
```

and for the Stefan-Boltzman radiation condition by

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

N.B. The global setting of `AXIS`ymmetric 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 dimention one less than that for the thermal element; thus, it is a point, 1-2 line, or 2-d surface.

6.3 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 or 3 nodes. For two dimensional problems each element may be a triangle with 3 or 6 nodes or a quadrilateral with 4, 8, 9, or 16 nodes (enhanced formulation permits only 4-node quadrilaterals). 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 problems, u_x, coordinates are x; 1-D Axisymmetric problems without torsion, u_r, coordinates are 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;

For three dimensional problems each element is a 4 or 10 node tetrahedron or an 8, 20, 27 or 64 node hexahedron (brick) with degree-of-freedoms u_x, u_y, u_z (enhanced element may only be an 8 node hexahedron).

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.

6.3.1 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;

$$\boldsymbol{\epsilon} = \left[\begin{array}{cccccc} \epsilon_{xx}\,, & \epsilon_{yy}\,, & \epsilon_{zz}\,, & \gamma_{xy}\,, & \gamma_{yz}\,, & \gamma_{zx} \end{array}\right]^T$$

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, or elasto-plastic.

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-1,1 C-1,2 ... C-1,n
C-2,1 C-2,2 ... C-2,n
...
C-n,1 C-n,2 ... C-n,n
```

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 [1]). To designate another formulation type the commands are given as:

```
MATErial,1
  SOLId
    ELAStic ....
    MIXEd, ENHAnced,  or DISPlacement
     ....
```

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 [1]).

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

6.3.2 One dimensional formulations

The input options for one dimensional analysis are:

- `PLANe STRAin, PLANe STREss` or `AXISymmetric`;
- `SMALl` or `FINIte`;

- `DISPlacement`, `MIXEd`; `ENHAnced`; or `CONServing`;
- `MASS LUMPed`, `MASS CONSistent` or `MASS OFF`;

In one dimensional applications the displacement and the mixed formulation may be described by line elements with two (2) or three (3) 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 2π is not included in any applied loads, etc.

6.3.3 Two dimensional formulations

The input options for two dimensional analysis are:

- `PLANe STRAin, PLANe STREss, AXISymmetric` or `AXISymmetric TORSion`;
- `SMALl` or `FINIte`;
- `DISPlacement`, `MIXEd`; `ENHAnced`; `MIXEd ENHAnced` or `CONServing`;
- `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

Type: `SOLId`	Command Name	Option Name	Dimen. `NDM`	Node/Elm. `NEL`
Displacement	SMAL	DISP	1	2-3
Mixed	SMAL	MIXE	1	2-3
Enhanced	SMAL	ENHA	1	2
Displacement	SMAL	DISP	2	3-16
Mixed	SMAL	MIXE	2	3-16
Enhanced	SMAL	ENHA	2	4
Mixed-Enhanced	SMAL	MIXE ENHA	2	4
Engy-Momentum	SMAL	CONS	2	3-16
Displacement	SMAL	DISP	3	4-27
Mixed	SMAL	MIXE	3	4-27
Enhanced	SMAL	ENHA	3	8
Engy-Momentum	SMAL	CONS	3	4-27

Table 6.1: Options for Small Deformation Solid Elements

Type: `SOLId`	Command Name	Option Name	Dimen. `NDM`	Node/Elm. `NEL`
Displacement	FINI	DISP	1	2-3
Mixed	FINI	MIXE	1	2-3
Engy-Momentum	FINI	CONS	1	2
Displacement	FINI	DISP	2	3-16
Mixed	FINI	MIXE	2	3-16
Engy-Momentum	FINI	CONS	2	4
Displacement	FINI	DISP	3	4-27
Mixed	FINI	MIXE	3	4-27
Enhanced	FINI	ENHA	3	8
Engy-Momentum	FINI	CONS	3	8

Table 6.2: Options for Large Deformation Solid 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 2π is not included in any applied loads, etc.

6.3.4 Three dimensional formulations

The input options for three dimensional analysis are:

- `SMALl` or `FINIte`;
- `DISPlacement`, `MIXEd`; `ENHAnced`; `MIXEd ENHAnced` or `CONServing`;
- `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 (27) nodes. The elements described by this range are shown in Fig. 3.4. The enhanced strain and mixed-enhanced elements are 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.

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

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

6.4 Frame Elements

Frame elements can treat small and large displacement problems. for two and three dimensional geometries. A frame element is included using the commands:

```
MATErial,1
  FRAMe
      ....
```

All frame elements have two nodes.

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 with small rotation element (2nd order theory, no shear deformation); and a large displacement and large rotation element with shearing 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.

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	No	No	No
Euler-Bernoulli (Second order)	FINIte SHEAr OFF	Cubic	Yes	No	No	No
Timoshenko (Simo-Vu Quoc)	FINIte SHEAr ON MIXED	Linear	Yes	No	No	No
Timoshenko (Co-rotational)	FINIte SHEAr ON COROtate	Linear	Yes	No	No	No
Timoshenko (Ibrahimbegovic)	FINIte SHEAr ON	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

Two types of elements are provided for three dimensions: A small deformation element based on Euler-Bernoulli theory and a large displacement, large rotation element based on exact the kinematic formulation of Simo et. al. [17, 18] and Ibrahimbegovic [19]. The large displacement element includes elastic resultant and 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 [20].

The required data for frame elements is the material model, cross section data, 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 `REFErence` `VECTor`, `DIREction`, or `NODE`.

```
MATErial,1
  FRAMe
    REFErence VECTor or NODE or DIREction
      ....
```

`ELAStic` models are 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.

6.5 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,1
  TRUSs
      ....
```

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

6.6 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,1
  PLATe
      ....
```

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

6.7 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. [21] 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 TRANsient analysis with either the STATic or CONServing options (see chapter on transient solutions). Also, the tangent matrix is non-symmetric, thus to achieve quadratic rates of convergence the UTANgent solution command must also be employed. The shell element is included using the commands:

```
MATErial,1
  SHELl
      ....
```

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

6.8 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,1
  MEMBrane
      ....
```

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

6.9 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,1
  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.

6.10 Pressure: Follower loads

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

```
MATErial,1
  PRESsure
    LOAD p prop-ld
      ...
```

Loading is specified by options `LOAD` and, for follower loads by `FINIte` or `FOLLower`. Loading intensity may be associated with the proportional loading number `prop-ld`.

6.11 Gap Element

The *gap element* requires very little data to use. The material record is given as:

```
MATErial,1
  GAP
    DIREction,x-dir
    DEGRee,n-dof
    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 specifying a third node on the element. One degree of freedom from the third node (i.e., the dof `n-dof`) will be used to store the Lagrange multiplier value. 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 in which the profile (active column) solver is used – this is the default solver or 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.

6.12 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`. Accordingly, a typical set of data for a user element `elmt12` is given as:

```
MATErial,1
  USER     12              ! Use elmt12(...) module
  xxxxxx                   ! Additional data records
                           ! blank termination record
```

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

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

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.

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 Elastic Models

A linear elastic material model in *FEAP* is given by

$$\boldsymbol{\epsilon} = \mathbb{C}\,\boldsymbol{\sigma} + \boldsymbol{\epsilon}^{th} \tag{7.4}$$

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

For analysis purposes the model in inverted to:

$$\boldsymbol{\sigma} = \mathbb{D}\,\left[\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^{th}\right] = \mathbb{D}\,\boldsymbol{\epsilon} + \boldsymbol{\beta}^{th} \tag{7.5}$$

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

$$\mathbb{D} = \mathbb{C}^{-1} \tag{7.6}$$

and

$$\boldsymbol{\beta}^{th} = -\,\mathbb{D}\,\boldsymbol{\epsilon}^{th} \quad . \tag{7.7}$$

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:

$$\mathbb{C} = \begin{bmatrix} \frac{1}{E} & -\frac{\nu}{E} & -\frac{\nu}{E} & 0 & 0 & 0 \\ -\frac{\nu}{E} & \frac{1}{E} & -\frac{\nu}{E} & 0 & 0 & 0 \\ -\frac{\nu}{E} & -\frac{\nu}{E} & \frac{1}{E} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G} \end{bmatrix} \tag{7.8}$$

with the shear modulus G related through

$$G = \frac{E}{2\,(1 + \nu)} \quad . \tag{7.9}$$

For isotropic materials the thermal strain is given by

$$\boldsymbol{\epsilon}^{th} = \begin{bmatrix} \alpha \\ \alpha \\ \alpha \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta T \tag{7.10}$$

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 is given as:

```
MATErial 1
  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	HARDening	H_{iso}, H_{kin}
VISCoelastic		μ_i, τ_i
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
ANGLe		ψ
DENSity		ρ

Table 7.2: Material Model Data Inputs

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

$$\mathbb{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\nu) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-2\nu)/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & (1-2\nu)/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & (1-2\nu)/2 \end{bmatrix} . \tag{7.11}$$

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

$$\epsilon_{33} = -\frac{\nu}{E} (\sigma_{11} + \sigma_{22}) + \alpha \, \Delta T \tag{7.12}$$

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

Table 7.3: Small deformation models for solid elements

and

$$\mathbb{D} = \frac{E}{(1-\nu^2)} \begin{bmatrix} (1-\nu) & \nu & 0 & 0 & 0 & 0 \\ \nu & (1-\nu) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & (1-\nu)/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{7.13}$$

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

$$\hat{\boldsymbol{\epsilon}} = \hat{\mathbb{C}}\,\hat{\boldsymbol{\sigma}} + \hat{\boldsymbol{\epsilon}}^{th} \tag{7.14}$$

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:

$$\hat{\mathbb{C}} = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{12}}{E_1} & -\frac{\nu_{13}}{E_1} & 0 & 0 & 0 \\ -\frac{\nu_{21}}{E_2} & \frac{1}{E_2} & -\frac{\nu_{23}}{E_2} & 0 & 0 & 0 \\ -\frac{\nu_{31}}{E_3} & -\frac{\nu_{32}}{E_3} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{12}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{23}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{31}} \end{bmatrix} \tag{7.15}$$

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. The above sign convention corresponds to

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

and the definition of terms is identical to that given by Christensen [22] (except for shear modulus terms).

The thermal strain is given by:

$$\hat{\boldsymbol{\epsilon}}^{th} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta T = \hat{\boldsymbol{\alpha}}\,\Delta T \tag{7.16}$$

where

$$\Delta T \;=\; T \,-\, T_0\,, \tag{7.17}$$

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 7.12. Using this angle *FEAP* transforms the input material compliances to

$$\mathbb{C} = \mathbf{R}^T \, \hat{\mathbb{C}} \, \mathbf{R} \tag{7.18}$$

and converts the constitutive equation to the form given in Eqs. 7.5 to 7.7.

Material data is given by the command set:

```
MATErial 1
  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.

7.2.3 Transversly 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. (7.15) the parameters defining the transversely isotropic model satisfy the constraints:

$$E_1 = E_3 \;\; ; \;\; \nu_{12} = \nu_{23} \;\; ; \;\; G_{31} = \frac{E_1}{2(1+\nu_{31})} \;\text{ and }\; G_{12} = G_{23} \tag{7.19}$$

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

$$\alpha_1 = \alpha_3 \tag{7.20}$$

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 1
  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 7.12. 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 1
  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

$$\boldsymbol{\epsilon} = \mathbb{C}\,\boldsymbol{\sigma} \tag{7.21}$$

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

$$\boldsymbol{\sigma} = \mathbb{D}\, \boldsymbol{\epsilon} \tag{7.22}$$

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

To input moduli directly, the input sequence

```
MATErial 1
  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.

7.3 Finite Deformation

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

$$\mathbf{x} = \boldsymbol{\varphi}(\mathbf{X}, t) \tag{7.23}$$

The deformation gradient is then defined as

$$\mathbf{F} = \frac{\partial \boldsymbol{\varphi}}{\partial \mathbf{X}} \ . \tag{7.24}$$

Additional measures of deformation are given by the right Cauchy-Green deformation tensor

$$\mathbf{C} = \mathbf{F}^T \mathbf{F} \tag{7.25}$$

and the left Cauchy-Green deformation tensor

$$\mathbf{b} = \mathbf{F}\,\mathbf{F}^T \tag{7.26}$$

A measure of strain is provided by the Green strain

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{C} - \mathbf{1}\right) \tag{7.27}$$

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 capabilitites 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		
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

Table 7.5: Finite deformation 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

$$\mathbf{S} = \frac{\partial W(\mathbf{C})}{\partial \mathbf{C}} \tag{7.28}$$

where W is a *stored energy* function. Stress in the current configuration may be deduced by transformation (pushing) the stress. Accordingly

$$\sigma = \frac{1}{J} \mathbf{F}\,\mathbf{S}\,\mathbf{F}^T \; . \tag{7.29}$$

Isotropic models may be expressed in terms of the invariants of the deformation tensor. Accordingly, the three principal invariants given by

$$I_C = \operatorname{tr} \mathbf{C} \tag{7.30}$$

$$II_C = \frac{1}{2}\left(I_C^2 - \operatorname{tr} \mathbf{C}^2\right) \tag{7.31}$$

and

$$III_C = \det \mathbf{C} = J^2 \tag{7.32}$$

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

$$\mathbf{C} = \sum_{A=1}^{3} \lambda_A^2 \, \mathbf{N}_A \otimes \mathbf{N}_A \; . \tag{7.33}$$

The invariants are then given by

$$I_C = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \tag{7.34}$$

$$II_C = \lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2 \tag{7.35}$$

and

$$III_C = \lambda_1^2\lambda_2^2\lambda_3^2 \; . \tag{7.36}$$

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

$$\mathbf{F} = \mathbf{F}_{vol}\,\mathbf{F}_{dev} \tag{7.37}$$

where

$$\mathbf{F}_{vol} = J^{1/3}\mathbf{1} \tag{7.38}$$

and

$$\mathbf{F}_{dev} = J^{-1/3}\mathbf{F} \tag{7.39}$$

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

$$\mathbf{C}_{dev} = J^{-2/3}\mathbf{C} = \mathbf{F}^T_{dev}\mathbf{F}_{dev} \quad . \tag{7.40}$$

and again has unit volume change. The stored energy function may then be written in terms of the invariants

$$\bar{I}_C = \operatorname{tr}\mathbf{C}_{dev} = J^{-2/3}\, I_C \tag{7.41}$$

$$\begin{aligned} \bar{II}_C &= \frac{1}{2}\left(\bar{I}^2_C - \operatorname{tr}\mathbf{C}^2_{dev}\right) \\ &= \frac{1}{2}\, J^{-4/3}\left(I^2_C - \operatorname{tr}\mathbf{C}^2\right) \end{aligned} \tag{7.42}$$

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:

$$\mathbf{S} = \mathbb{D}\,\mathbf{E} \tag{7.43}$$

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

$$W = \frac{1}{2}\,\mathbf{E}^T\mathbb{D}\,\mathbf{E} \; . \tag{7.44}$$

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 1
  SOLId
  ELAStic STVE(or STVK) E nu
                           ! blank termination record
```

for an isotropic material where a `FINIte` statement is optional; or by for an orthotropic material where the `FINIte` is required to distinguish from the small deformation case.

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 1
  SOLId
  FINIte
  ELAStic CONServing E nu
                           ! blank termination record
```

Recall that the location of the `FINI`te command is not order dependent so that the commands can also be.

```
MATErial 1
  SOLId
  ELAStic CONServing E nu
  FINIte
                           ! blank termination record
```

Optionally the finite deformation designation also may be given for all elements as `GLOB`al 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

$$W(\mathbf{E}) = C \, \exp(\mathbf{E}^T \mathbf{A} \, \mathbf{E}) \; . \tag{7.45}$$

Here the array **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 1
  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.

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

Table 7.6: Isotropic Finite Deformation Elastic Material Models and Inputs

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 $\mathbf{F}$, multiplied by a material parameter. The volumetric function in *FEAP* is taken from one of the following:

$$\begin{aligned} 1: \quad U(J) &= \frac{1}{4}\,(J^2 - 1 - 2\,lnJ) \\ 2: \quad U(J) &= \frac{1}{2}\,(J-1)^2 \\ 3: \quad U(J) &= \frac{1}{2}\,ln^2 J \end{aligned} \tag{7.46}$$

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

$$W = \left(K - \frac{2}{3}G\right)\,U(J) + \frac{1}{2}\,G\,\left(I_C - 3\right) \tag{7.47}$$

where I_C is defined in Eq. (7.30). 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

$$K = \frac{E}{3\,(1-2\nu)} \quad ; \quad G = \frac{E}{2\,(1+\nu)}\ . \tag{7.48}$$

The data set to use this form is given by

```
MATErial 1
  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. (7.46).

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 (7.37) to (7.39). Accordingly, the stored energy function is given as

$$\begin{aligned} W &= K\,U(J) + \frac{1}{2}\,G\,\left(J^{-2/3} I_C - 3\right) \\ &= K\,U(J) + \frac{1}{2}\,G\,\left(\bar{I}_C - 3\right) \end{aligned} \tag{7.49}$$

where $\bar{I}_C$ is defined by Eq. (7.41).

The parameters K and G are again specified by their small strain equivalent E and ν defined in Eq. 7.48.

The data set to use the modified form is given by

```
MATErial 1
  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

$$W = K\,U(J) + \frac{1}{2}\,G\left[(1-c)\,(I_C - 3 - 2\,\ln J) + c\,(II_C - 3 - 4\,\ln J)\right] \tag{7.50}$$

where I_C and II_C are defined in (7.30) and (7.31), 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. 7.46.

A modified form of the Mooney-Rivlin model is given as

$$W = K\,U(J) + \frac{1}{2}\,G\left[(1-c)\left(\bar{I}_C - 3\right) + c\left(\bar{II}_C - 3\right)\right] \tag{7.51}$$

where $\bar{I}_C$ and $\bar{II}_C$ are defined in (7.41) and (7.42), respectively. The input data for the Mooney-Rivlin model is given by

```
    MATErial 1
      SOLId
      FINIte <VOLUme n>
      ELAStic MOONey E nu c
                                 ! blank termination record
```

for the standard model and

```
    MATErial 1
      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. 7.48.

7.3.5 Ogden Model

FEAP also contains a model for hyper-elastic behavior which is expressed directly in terms of *deviatoric* principal stretches, $\tilde{\lambda}_A$. This model has a stored energy function expressed in the form:

$$W = K\,U(J) + \sum_{A=1}^{3} w(\tilde{\lambda}_A, J) \tag{7.52}$$

and is based on the *Valanis-Landel hypothesis* ([23, 24]). The deviatoric principal stretches are defined as

$$\tilde{\lambda}_A = J^{-1/3}\lambda_A \tag{7.53}$$

and used to write the scalar stored energy functions as

$$w(\tilde{\lambda}_A) = \sum_j \frac{C_j}{a_j}\left(\tilde{\lambda}_A^{a_j} - 1\right) \tag{7.54}$$

where, in *FEAP*, j can range from 1 to 3 terms. The data input for the Ogden model is given as

```
    MATErial 1
      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 Logarithmic Stretch Model

An alternative principal stretch model is defined by strains expressed as

$$\epsilon_A = \log \lambda_A \ . \tag{7.55}$$

The stored energy function for this form is identical to the small strain isotropic model expressed in principal strains. Accordingly,

$$W(\lambda_A) = \frac{1}{2}\left(K - \frac{2}{3}G\right)\left(\sum_{A=1}^{3} \epsilon_A\right)^2 + G \sum_{A=1}^{3} \epsilon_A^2 \ . \tag{7.56}$$

The stress-strain behavior for principal stresses σ_A and principal strains ϵ_A is given by

$$\left\{ \begin{array}{c} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{array} \right\} = \frac{E}{(1+\nu)(1-2\,\nu)} \left[\begin{array}{ccc} (1-\nu) & \nu & \nu \\ \nu & (1-\nu) & \nu \\ \nu & \nu & (1-\nu) \end{array} \right] \left\{ \begin{array}{c} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{array} \right\} . \tag{7.57}$$

The transformation to the global stresses is carried out as defined in Ogden [24].

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 hyper-elastic 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:

$$\boldsymbol{\sigma} \; = \; \sigma_{vol}\,\mathbf{1} + \boldsymbol{\sigma}_{dev} \tag{7.58}$$

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

$$\boldsymbol{\epsilon} \; = \; \frac{1}{3}\,\theta\,\mathbf{1} + \boldsymbol{\epsilon}_{dev} \tag{7.59}$$

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

$$\boldsymbol{\sigma}_{dev} \; = \; 2\,G\int_{-\infty}^{t}\mu(t-\tau)\,\frac{d\boldsymbol{\epsilon}_{dev}}{d\tau}\,d\tau \tag{7.60}$$

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)

$$\mu(t) \; = \; \mu_0 + \sum_i \mu_i\,\exp -t/\tau_i \; . \tag{7.61}$$

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

$$\mu_0 \; = \; 1 - \sum_i \mu_i \; . \tag{7.62}$$

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 1
  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 [25]). 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. 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 1
  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.

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

$$\begin{aligned} \boldsymbol{\sigma}^\star &= \boldsymbol{\sigma}_\Re + i\,\boldsymbol{\sigma}_\Im \\ \boldsymbol{\epsilon}^\star &= \boldsymbol{\epsilon}_\Re + i\,\boldsymbol{\epsilon}_\Im \end{aligned} \tag{7.63}$$

with $i = \sqrt{-1}$. With this representation we may then write the viscoelastic material response as

$$\boldsymbol{\sigma}^\star(\omega) = \mathbb{D}^\star(\omega)\,\boldsymbol{\epsilon}^\star(\omega) \tag{7.64}$$

where $\mathbb{D}^\star(\omega)$ are frequency dependent complex moduli.

If we split the stress into volumetric and deviatoric components as

$$\begin{aligned} \boldsymbol{\sigma}^\star &= \sigma^\star_{vol}\,\mathbf{1} + \boldsymbol{\sigma}^\star_{dev} \\ \boldsymbol{\epsilon}^\star &= \theta^\star\,\mathbf{e} + \boldsymbol{\epsilon}^\star_{dev} \end{aligned} \tag{7.65}$$

and consider isotropic materials only we can write the response in terms of two complex modulus functions as

$$\sigma^{\star}_{vol} = K^{\star}\,\theta^{\star} \quad \text{and} \quad \boldsymbol{\sigma}^{\star}_{dev} = 2\,G^{\star}\,\boldsymbol{\epsilon}^{\star}_{dev} \tag{7.66}$$

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

$$\begin{aligned} K^{\star} &= K_{\Re} + i\,K_{\Im} \\ G^{\star} &= G_{\Re} + i\,G_{\Im} \quad . \end{aligned} \tag{7.67}$$

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

$$G(t) = G\left[\mu_0 + \sum_{i}^{n} \mu_i\, \exp -t/\tau_i\right] \quad . \tag{7.68}$$

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

$$\mu_0 + \sum_{i}^{n} \mu_i = 1 \quad \text{with} \mu_0\, \mu_i > 0 \quad . \tag{7.69}$$

The complex shear modulus for this representation has real and imaginary parts given by

$$\begin{aligned} G_{\Re} &= G\left[\mu_0 + \sum_{i}^{n} \mu_i \left(\frac{\omega^2 \tau_i^2}{1 + \omega^2 \tau_i^2}\right)\right] \\ G_{\Im} &= G \sum_{i}^{n} \mu_i \left(\frac{\omega \tau_i}{1 + \omega^2 \tau_i^2}\right) \quad . \end{aligned} \tag{7.70}$$

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
   ...
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. The 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. 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

$$\epsilon = \epsilon^{el} + \epsilon^{pl} \ . \tag{7.71}$$

An associative flow rule is assumed so that the plastic strain rate may be computed from a *yield function*, F, as

$$\dot{\epsilon}^{pl} = \dot{\gamma} \, \frac{\partial F}{\partial \sigma} \; . \tag{7.72}$$

The relation may be integrated in time using a backward Euler (implicit) time integration to compute a discrete form of the problem.

Isotropic and kinematic hardening are also added to the model. The kinematic hardening is limited to a linear form where it is assumed that

$$\alpha = H_{kin} \, \epsilon^{pl} \tag{7.73}$$

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

$$Y(e^{pl}) = Y_\infty + (Y_0 - Y_\infty) \exp(-\beta \, e^{pl}) + H_{iso} \, e^{pl} \tag{7.74}$$

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

$$e^{pl} = \int_0^t \dot{\gamma} \, d\tau \; . \tag{7.75}$$

In *FEAP* the discrete problem is solved using a closest point return map algorithm (e.g., see [26, 27, 28]).

Input properties for a simple material with no saturation hardening and linear isotropic hardening is given by:

```
MATErial 1
  SOLId
  ELAStic ISOTropic 30e+06 0.3
  PLAStic MISEs     30e+03
  PLAStic HARDening 3000   0
                           ! blank termination record
```

7.6 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:

$$\mathbf{M}_{interp} \;=\; (1\,-\,a)\,\mathbf{M}_{cons} \;+\; a\,\mathbf{M}_{lump}\;. \tag{7.76}$$

The type of mass and, where required, the parameter a are input using the `MASS` command as shown in Table 7.7 and the elements which are affected by the command are indicated in Table 7.8.

Command	Type	Parameters
MASS	LUMPed	
MASS	CONSistent	
MASS	OFF	
MASS		a

Table 7.7: Material Model Mass Related Inputs

Command	Solid	Truss	Frame	Plate	Shell	Membrane	Thermal
MASS	X	X	X	X	-	-	X

Table 7.8: Mass Command vs. Element Types

7.7 Rayleigh Damping

The effects of damping may be included in transient solutions assuming a damping matrix in the form

$$\mathbf{C} \;=\; a_0\,\mathbf{M} \;+\; a_1\,\mathbf{K}\;. \tag{7.77}$$

This defines a form called *Rayleigh Damping.* The input for this form of damping is given by:

```
MATERIAL
  .....
    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 14.4.2).

7.8 Element Cross Section and Load Specification

7.8.1 Resultant formulations

The plane stress and structural elements require specification of cross-section information. For the plane stress, plate, and shell elements this 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. For the truss and frame elements it is necessary to provide cross-sectional property for area, and for the frame elements, flexural effects as indicated in Table 7.9.

Element loads for surface pressure and body force are input using the `LOAD`, `BODY` or `GROU`p force commands as shown in Table 7.9.

The types of elements affected by the `THICkness`, `LOAD` and `BODY` commands is indicated in Table 7.10.

Command	Type	Parameters
THICkness		h, κ
CROSS	section	$A, I_{xx}, I_{yy}, I_{xy}, J_{zz}, \kappa_x, \kappa_y$
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

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 pressent, 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

$$f_i = b_i + pr_i(t)\, g_i$$

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.

7.8.2 Section integration formulations

Structural element behavior may also be defined by numerical integration over the cross section using the `SECTion` command.

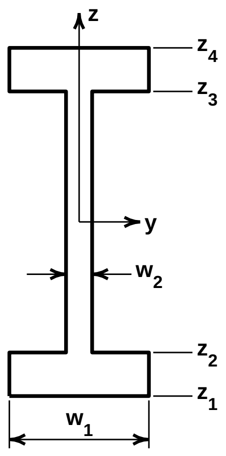


Figure 7.1: Cross-section for frame element

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.

Three dimensional frame sections

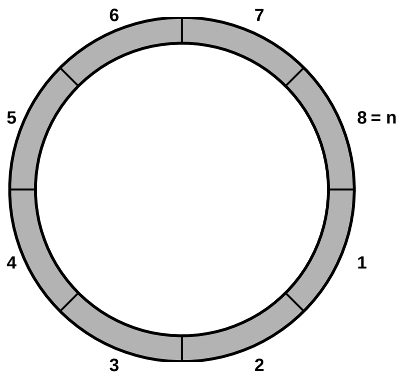
For three-dimensional truss and frame elements the cross section may be defined by alternate forms which include: `TUBE`, a thin circular tube; `RECT`angle, a rectangular solid section; `WIDE` flange, a wide flange composite section; `CHAN`nel, a channel composite section; `ANGL`e, an angle composite 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)
```

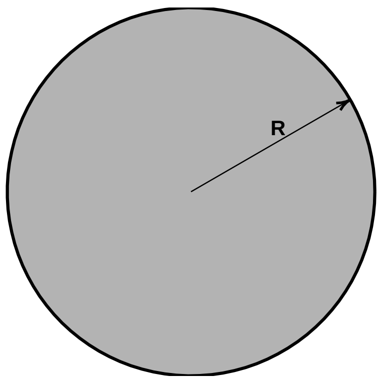
The data parameters `EV` for each type are summarized in Table 7.11 where all the parameters except the quadrature order are shown in Fig. 7.2.

In Table 7.11 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.

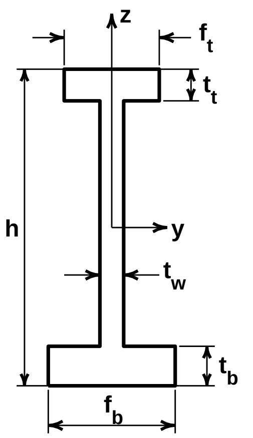
The cross-section shown in Fig. 7.1 can be input for a three-dimensional frame element using the command set:



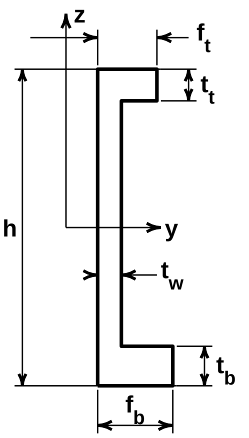
(a) Tube section $n = 8$



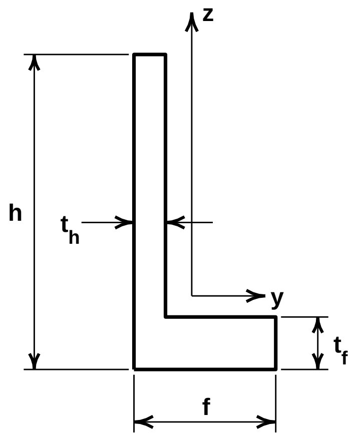
(b) Circle section



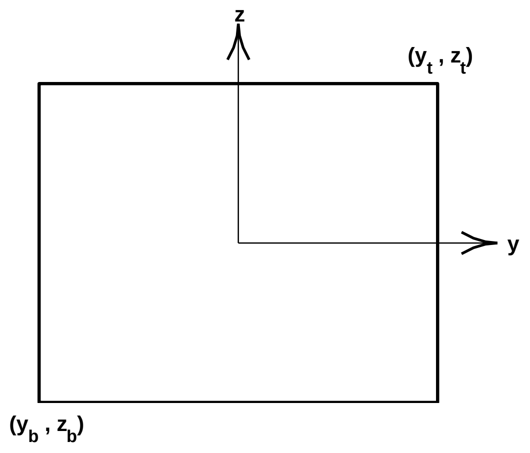
(c) Wide flange section



(d) Channel section



(e) Angle section



(f) Rectangle section

Figure 7.2: Cross-sections types for 3-dimensional frame analysis

TYPE	EV(1)	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 7.11: Types and data for integrated cross-sections.

```
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*.

7.9 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 7.12 and the affected element types in Table 7.13. 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.

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 7.12 and the affected element types in Table 7.13.

Command	Type	Parameters
QUADrature		n_{array}, n_{output}
PENAlty		k_{pen}
ADAPtive	ERROr	η
TEMPerature		T_{dof}
SMALl	deformation	
FINIte	deformation	
NONLinear		
DISPlacment		
MIXEd		
ENHAnced	strain	
PLANe	STREss	
PLANe	STRAin	
AXISymmetric		

Table 7.12: 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 7.13: Miscellaneous Material Commands vs. Element Types

7.10 Global Data

FEAP uses the `GLOBal` command to specify data which is common to all elements. For example, 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 example problem is to be solved as a plane strain problem, the global data is specified as:

```
GLOBal
  PLANe STRAin
                     ! 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. 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.

A problem in solid mechanics may be designated as `SMALl` or `FINIte` deformation using global commands. In addition, the variable used for temperature in a coupled thermo-mechanical analysis and the `REFErence` vector or node for three dimensional problems using structural frame elements may be defined globally. Options also exist for users to add their own global options.

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
  GROUnd factors f_1 f_2 ... f_ndf
                     ! blank termination record
```

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 7.7). 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 7.7. The global damping value may also be used for modal solutions as described in Section 14.4.2.

Chapter 8

NODAL MASS, DAMPERS AND SPRINGS

FEAP has options to add discrete mass, damping, and stiffness terms to a 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

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
```

[1]Upper and lower case letters are different in UNIX or LINUX environment but the same in a Windows one

```
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 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 alphanumerical 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 Looping 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.

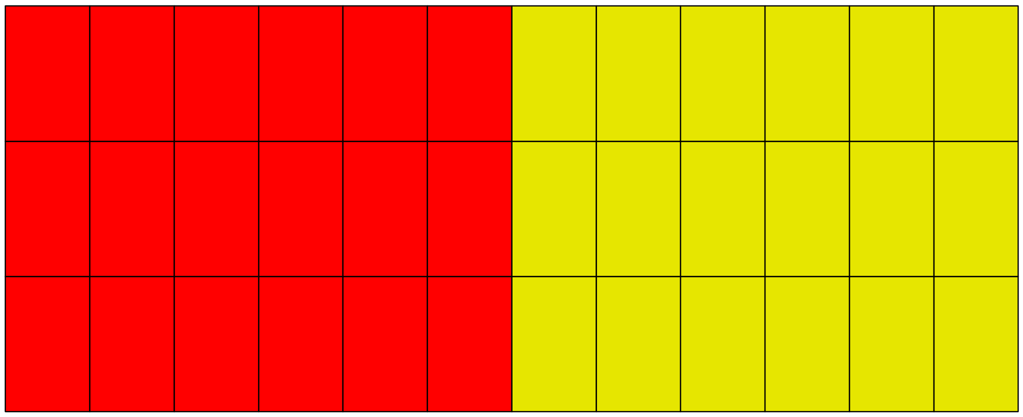


Figure 9.1: Two blocks using `LOOP-NEXT` commands

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 shown in Fig. 9.1 and is trivial (also not much is gained over a construction using two block commands directly).

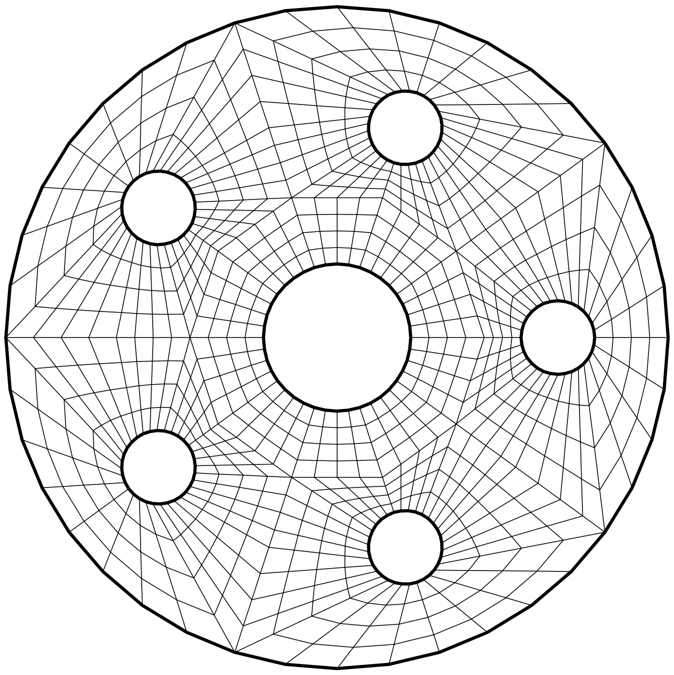


Figure 9.2: Disk with holes

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`

```
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

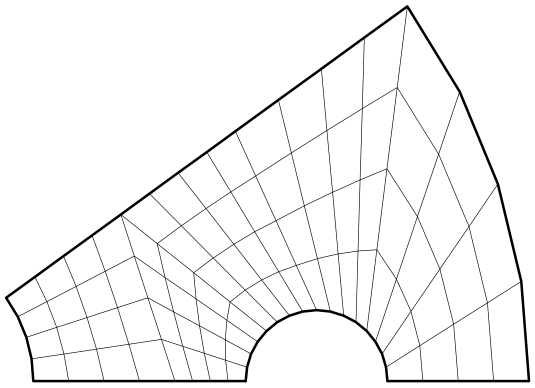


Figure 9.3: Mesh segment for disk with holes

```
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

or `SIDE` commands within any looping instructions. In this case a correct structure is:

```
SNODe
  1   ...
      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.

9.4 Node and Element Numbers: `*NOD` and `*ELE`

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 the `*NODe` and `*ELEment` option.

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:

```
ELEMent
   5 0 1 32 45
   ....
```

would add the current value of the `*NODe` to the input values `32` and `45` to produce new values of the nodes for this element.

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 `*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
```

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` and/or `ELINk` 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 The 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.

11.2 The LINK and ELINk 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.

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).

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 interpretted 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`).

11.3 The 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
  P1,P2,P3,...
```

is inserted after the mesh `END` command and before the first solution command. Values for `P-i` must be between 1 and 4 and must be specified for all active degree-of-freedoms (i.e., the total number specified on the control record described in Section 5.1).

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 is 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 next.

11.4 The 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 14.3.3. 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 *tieing* 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 tieing of interfaces is accomplished using a *dual mortar* Lagrange multipler 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
    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 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
      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 `ELEM`ent 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 `CSUR`face mesh command. The data for 2-node boundary segments is given as

```
  SURFace number
    LINE
      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.

12.1.3 Blend Specification

The `BLENd` option is analogous to the way sides are generated for blending function mesh generation. At present only two dimensional surfaces may be defined by the contact blend option. The data input is

```
SURFace number
  LINE
    BLENd SEGMent
      type sn-1 sn-2 sn-3 ....
                   ! termination record
```

where `type` is selected from `CARTesian`, `POLAr`, or `SEGMent`.

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

$$r(t) = R + u\, p_i(t) \quad .$$

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 Contact Material Models

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.

12.3.1 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 Example for Contact Input

As a simple example, consider the definition of a contact interaction between two blocks as shown in Figure 12.1. 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.

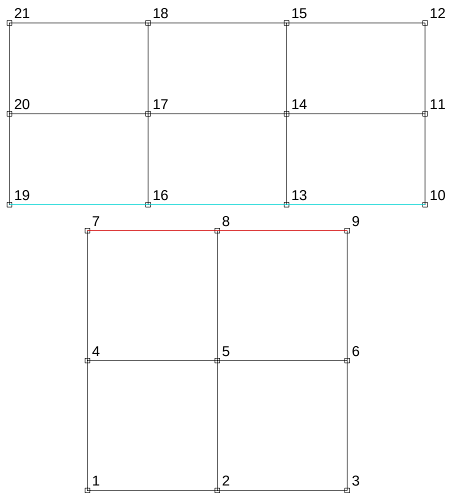


Figure 12.1: Contact analysis between two bodies

Chapter 13

RIGID BODY ANALYSIS

The rigid body capabilities are split into two classes. One for small displacement problems where translation and rotation parameters are linear and one for the large displacement case where the rotational parameters appear in a non-linear form.

13.1 Small Displacement Analyses

For the small displacement case the treatment of rigidity may be 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 `SURF`ace 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 Large Displacement Analyses

In performing a rigid-flexible body analysis for problems which undergo large motions and rotations 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.

13.2.1 Flexible or Rigid Groups

FEAP permits the use of both flexible and rigid finite elements. By default all elements are flexible. If it is desired to designate an element as rigid the command

```
        RIGId,number
```

must be inserted in the mesh data just before the elements belonging to rigid body `number` are input or generated using the `ELEM`ent, `BLOC`k, or `BLEN`d commands.

To designate elements as flexible the command

```
        FLEXible
```

must be inserted immediately before element groups which are to remain deformable. It is not necessary to include this statement if all elements are flexible.

The current release of FEAP does not fully support all rigid body options. Problems may be solved using the energy conserving algorithms; however, other algorithms may not converge quickly.

13.2.2 Activation

As noted above *FEAP* permits 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 (`BATC`h or `INTE`ractive). 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).

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 commands which can be used to solve problems using standard algorithms, such as linear static and transient methods and Newton's method to solve non-linear problems. Appendix D of the manual describes all the programming commands which are included in the current system. These commands are combined to define the solution algorithm desired.

To enter the solution 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 inserging a command `INTEractive` in the input file. Thus, 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. 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.

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 14.2 with examples of algorithms that can be constructed using the command language approach.

14.1 Basic Solution Commands

The set of *basic* solution commands is:

```
ACCE   CAPT   CHEC   DEBU   DISP   DT     EIGE   EPRI
FORM   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 B. All commands available in an installed program may be displayed during an interactive mode of solution by issuing the command `MANUal,,3` followed by a `HELP` command. 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 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[1]. 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

[1]The check part of user developed elements must be implemented for the check command to work properly

output file. If all the volumes (or areas) are negative most of the system element routines will perform a resequencing of the nodes and repeat the check. If the resequencing 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 Chapter 11 in Volume 1 of Zienkiewicz and Taylor 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 resequences 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 below).

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
   ..... ! 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>
```

where the parameter `ON` is optional. Debug prings 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.

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 incrment, 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 spurrious results.

14.1.8 `EPRInt` command

The `EPRINt` command is used to display the values in the last computed element tangent and residual arrays (i.e., the element with highest number). The command should be issued immediately after a `TANG` or `UTAN` command. The command is given by

```
        EPRInt
```

14.1.9 `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).

14.1.10 `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
```

and has the same input structure as the displacements.

In interactive mode the input is given immediatly after the `INIT` command (although this is generally not recommended as an input form).

14.1.11 `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.

14.1.12 `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.

14.1.13 `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.

14.1.14 `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
```

14.1.15 NEXT command

The `NEXT` command must be used with a matching preceding `LOOP` command (see Sect. 14.1.12 above). Thus, it must always have the structure

```
LOOP,,n
  ....
NEXT
```

14.1.16 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.

14.1.17 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 `.opt` extender added. The file only includes the mesh coordinates, element connections, boundary restraint codes, and nodal force and displacement values. It is necessary to append the material set data and any 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.

14.1.18 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 use of the parameter command requires additional data to define the parameters to be given after the batch `END` command. Thus the general form is:

```
BATCh
  ...
  PARAmeter
  ...
END
  .... ! Parameter values
  LIST ! List parameters in output file
```

14.1.19 `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.

14.1.20 `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 stiffnes, symmetric tangent, unsymmetric tangent or residual, respectively.

14.1.21 `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

$$p_j(t) = a_1 + a_2\,(t - t_min) + a_3\,(\sin a_4(t - t_min))^k$$

which remains valid only in the time range $t_min \leq t \leq t_max$.

A table input may be input using the command set:

```
BATCh
  PROP,,j
END
  2  np
  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 $t_i \leq t \leq t_i + 1$.

14.1.22 `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.

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

```
        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.

14.1.23 SHOW Command: Viewing Solution Data

The `SHOW` command permits users to display the problem size and values for some of the solution parameters 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 `DICT`ionary, *array name*, or be omitted. When omitted the `SHOW` command displays values for basic solution 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 ineractive 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*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.

14.1.24 `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[29] 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).

14.1.25 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 onto nodes and outputs the values for nodes `n1` to `n2` at increments of `inc`. 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.

14.1.26 SUBSpace command

The subspace command is 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 stiffnes 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 stiffnes 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 `gl` is not explicitly specified, *FEAP* sets it to the maximum of `nl` 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. 14.4.1below to solve this case).

14.1.27 `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

$$\mathbf{K}^\star_T = c_1\,\mathbf{K}_T + c_2\,\mathbf{C} + c_3\,\mathbf{M}$$

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. 14.4.1).

14.1.28 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.

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

$$t_n = t_{new} \quad .$$

14.1.29 TOL command

The `TOL` command is used to set the solution tolerance used to detect convergence in algorithms. The command is given by

```
TOL,,val
```

where `val` is the convergence tolerance value. It is not necessary to use this command unless the default value of 10^{-16} needs to be changed. *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 product of an increment becomes less than 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.

14.1.30 `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              ! Tplot end
```

where `type` options and parameters are shown in Table 14.1. 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
```

Type	Number	Point	Coordinates
DISP	n1	n2	$x\,y\,z$
VELO	n1	n2	$x\,y\,z$
ACCE	n1	n2	$x\,y\,z$
REAC	n1	n2	$x\,y\,z$
STRE	n1	n2	$x\,y\,z$
ELEM	n1	n2	$x\,y\,z$
USER	n1	n2	$x\,y\,z$
CONT	n1	n2	$x\,y\,z$
ARCL	n1	n2	
RSUM	n1	n2	
SUMS	n1	n2	

Table 14.1: Time history plot options

```
DISP 5 2
SHOW
```

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).

14.1.31 `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 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

$$\mathbf{R}(\mathbf{u}, \dot{\mathbf{u}}) = \mathbf{0}$$

Method	Description	ODE Order
`OFF`	Quasi-static method	0
`STAT`	Midpoint static method	0
`BACK`	Backward Euler implicit method	1
`BDF2`	Backward difference formula method	1
`GEN1`	Generalized midpoint method	1
`NEWMark`	Newmark method	2
`ALPHa`	HHT alpha-method	2
`CONServe`	Energy-momentum conserving method	2
`EXPLicit`	Newmark explicit method ($\beta = 0$)	2
`CENTral`	Central difference explicit method	2
`USER`	User programmed method	-

Table 14.2: Transient integration methods in *FEAP*

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 14.2 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 depenedent. 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` and `CENTral` should work with all material models. The remaining methods work only for elastic and thermal models.

14.1.32 `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.

14.1.33 `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 nodes.

14.2 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

```
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 B 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.

14.2.1 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.

14.2.2 Solution of linear equations

The use of Newton's method results in a set of linear algebraic equations which are solved to give the incremental displacements. *FEAP* includes several options for solving linear equations. The default solution scheme is the variable band, profile scheme discussed in Chapter 20 of *The Finite Element Method, Vol 1*, 5th edition[1] (or in Chapter 15 of *The Finite Element Method, Vol 1*, 4th edition[13]). 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 `DIRE`ct,`SPAR`se before the first use of the `TANG`ent command. *WARNING: If the sparse solver requires more memory than available, the program will '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`. 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.

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 `TANGent` command may be used. The PCG with diagonal preconditioner is requested by the command `ITERation` before the first `TANGent` command. A block nodal preconditioner may be requested using the command `ITER,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).

FEAP has also been interfaced to the scalar form of the SuperLU equation solver. Information on downloading the interface and the source for SuperLU is available at `www.ce.berkeley.edu/~rlt/feap`.

14.3 Transient Solutions

FEAP provides several alternatives to construct transient solutions. To solve a non-linear time dependent problem using Newton's method with a time integration method 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

`VELO`city and `ACCE`leration (see Appendix B).

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 14.2 shows the methods and solution order of the differential equation to be solved for each method available in *FEAP*.

14.3.1 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 `TRAN`sient 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.

14.3.2 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.

14.3.3 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 **P** vector is given by:

$$\mathbf{P} \;=\; \mathbf{P}_\sigma \;+\; \mathbf{C}\,\dot{\mathbf{u}} \;+\; \mathbf{M}\,\ddot{\mathbf{u}} \tag{14.26}$$

with **C** the damping and **M** the mass matrix.

The solution options for second order problems are:

1. A Newmark method [30] 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:

   $$\mathbf{u}_{n+1} \;=\; \mathbf{u}_n \;+\; \Delta t\,\mathbf{v}_n \;+\; \Delta t^2\,[\,(0.5-\beta)\,\mathbf{a}_n \;+\; \beta\,\mathbf{a}_{n+1},\;] \tag{14.29}$$

   and

   $$\mathbf{v}_{n+1} \;=\; \mathbf{v}_n \;+\; \Delta t\,[\,(1-\gamma)\,\mathbf{a}_n \;+\; \gamma\,\mathbf{a}_{n+1}\,] \tag{14.30}$$

   in which β and γ are parameters controlling stability and numerical dissipation. The command

   ```
   TRANsient NEWMark beta gamma
   ```

   is used to select this integration scheme. Optionally, the command

   ```
   TRANsient,,beta gamma
   ```

   also selects the Newmark algorithm. Default values are $\beta = 0.25$ and $\gamma = 0.5$.

   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
    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`, `ACCE`leration 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 `INIT`ial command. The initial command requires additional data which in a `BATC`h 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 [31] 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

$$\mathbf{u}_{n+\alpha} \;=\; (1-\alpha)\,\mathbf{u}_n \;+\; \alpha\,\mathbf{u}_{n+1} \tag{14.32}$$

$$\mathbf{v}_{n+\alpha} \;=\; (1-\alpha)\,\mathbf{v}_n \;+\; \alpha\,\mathbf{v}_{n+1} \tag{14.33}$$

$$\mathbf{a}_{n+\alpha} \;=\; \mathbf{a}_{n+1}\ . \tag{14.34}$$

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 $\alpha = 0.5$.

3. An energy conserving form of the alpha method [32, 33, 34] (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.

4. 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)
LOOP,time,nstep
  TIME
  FORM
  EXPLicit
  ... Outputs
NEXT
```

5. 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

$$\Delta t_m = \frac{1}{2}\left(\Delta t_{n+1} + \Delta t_{n+2}\right) \; .$$

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

$$\mathbf{v}_{n+1} = \frac{u_{n+1} - u_n}{\Delta t_{n+1}} \quad .$$

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).

14.3.4 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. 11.4):

```
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. 11.3) 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 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.

14.4.1 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

$$[\mathbf{K}_{uu}]\,[\boldsymbol{\Phi}] \;=\; [\mathbf{M}_{uu}]\,[\boldsymbol{\Phi}]\,[\boldsymbol{\Lambda}] \tag{14.45}$$

where

$$[\boldsymbol{\Phi}] \;=\; \begin{bmatrix} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_n \end{bmatrix} \tag{14.46}$$

is the set of *normal modes* and

$$[\boldsymbol{\Lambda}] \;=\; \begin{bmatrix} \omega_1^2 & 0 & \cdots & 0 \\ 0 & \omega_2^2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \omega_n^2 \end{bmatrix} \tag{14.47}$$

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 B for specifying additional options).

For example, if the modes for an *unsupported structure* (which will contain zero eigen-values) are desired, the tangent matrix is *singular* and the subspace method will fail to converge or an error may result during the construction of the triangular factors of the **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:

$$\left([\mathbf{K}_{uu}] \; - \; \chi \, [\mathbf{M}_{uu}] \right) [\mathbf{\Phi}] \; = \; [\mathbf{M}_{uu}] \, [\mathbf{\Phi}] \left[\tilde{\mathbf{\Lambda}} \right] \tag{14.51}$$

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.

14.4.2 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, [35]. 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 7.7).

14.4.3 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
```

14.4.4 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:

$$\begin{bmatrix}\mathbf{u}_u\\ \mathbf{u}_s\end{bmatrix} = \begin{bmatrix}\mathbf{\Phi}\\ \mathbf{0}\end{bmatrix}[\mathbf{v}] + \begin{bmatrix}\mathbf{\Psi}\\ \mathbf{I}\end{bmatrix}[\mathbf{w}] \tag{14.60}$$

where the arrays $\mathbf{\Phi}$ and $\mathbf{\Psi}$ 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

$$\mathbf{K}_{uu}\,\mathbf{\Psi} + \mathbf{K}_{us}\,\mathbf{I} = \mathbf{0}\;. \tag{14.61}$$

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 $\mathbf{\Phi}^T$ to give

$$\begin{aligned}&\mathbf{\Phi}^T\mathbf{M}_{uu}\mathbf{\Phi}\,\ddot{\mathbf{v}} + \mathbf{\Phi}^T\mathbf{M}_{uu}\mathbf{\Phi}\,\dot{\mathbf{v}} + \mathbf{\Phi}^T\mathbf{K}_{uu}\mathbf{\Phi}\,\mathbf{v}\\ &\qquad = \mathbf{G} - \mathbf{\Phi}^T\left[\mathbf{M}_{uu}\mathbf{\Psi} + \mathbf{M}_{us}\right]\ddot{\mathbf{w}} - \mathbf{\Phi}^T\left[\mathbf{C}_{uu}\mathbf{\Psi} + \mathbf{C}_{us}\right]\ddot{\mathbf{w}}\;.\end{aligned} \tag{14.62}$$

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 `FPRO`portional 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 $\boldsymbol{\Psi}$ 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 `TANG`ent command. Since base supports are provided, no *shift* should be included on the tangent command.

14.5 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. At present, the implementation in *FEAP* restricts the force specified during input to be real. Accordingly,

$$\mathbf{F}_r \;=\; \Re(\hat{\mathbf{F}}) \tag{14.75}$$

$$\mathbf{F}_i \;=\; \Im(\hat{\mathbf{F}}) = \mathbf{0}\ . \tag{14.76}$$

The real part of the force may be input using the mesh commands `FORC`e, `CFOR`ce, `EFOR`ce, and/or `CSUR`face. 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:

$$\left[-\,\omega^2\,\mathbf{M} \;+\; i\,\omega\,\mathbf{C} \;+\; \mathbf{K}\right]\,\hat{\mathbf{u}}(\omega) \;=\; \hat{\mathbf{F}}(\omega) \tag{14.78}$$

which may be solved for each specified frequency and load to give a solution for the $\hat{\mathbf{u}}(\omega)$.

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

$$\hat{\mathbf{A}} \;=\; -\,\omega^2\,\mathbf{M} \;+\; i\,\omega\,\mathbf{C} \;+\; \mathbf{K} \tag{14.79}$$

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, only a real part may currently 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. 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 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
```

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 specifications of the frequency ω.

After a solution is available the usual *FEAP* commands may be used to output or plot the solution. For example, the command

```
DISPlacement,,k1 k2 k3
```

outputs the real part of the displacement for nodes `k1` to `k2` at increments of `k3`. 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.

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.

14.6 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 and in the *FEAP* Programmers Manual.[12]

The third option is based on an error estimator for the transient response. Knowing the step-size for the current solution (Δt_{n+1}) a new step-size for the next solution is sellected based on a predictor error defined by

$$\mathbf{e}_{n+1}^{pred} = \mathbf{u}_{n+1} - \tilde{\mathbf{u}}_{n+1} \tag{14.84}$$

where $\tilde{\mathbf{u}}_{n+1}$ is a quadratic extrapolation for the initial guess to a Newton solution. The predictor error may be approximated by

$$\mathbf{e}_{n+1}^{pred} = \mathbf{u}[t_{n+1}, t_n, t_{n-1}, t_{n-2}](\Delta t_{n+1} + \Delta t_n)(\Delta t_{n+1} + \Delta t_n + \Delta t_{n-1}) \tag{14.85}$$

In (14.85) the divided difference is computed from the recursion

$$\mathbf{u}[t_{n+1}, t_n, \cdots, t_{n-i-1}] = \frac{\mathbf{u}[t_{n+1}, t_n, \cdots, t_{n-i}] - \mathbf{u}[t_n, t_{n-1}, \cdots, t_{n-i-1}]}{t_{n+1} - t_{n-i-1}} \tag{14.86}$$

and for

$$\mathbf{v}_k = \frac{\mathbf{u}_k - \mathbf{u}_{k-1}}{\Delta t_k} \tag{14.87}$$

yields

$$\mathbf{u}[t_{n+1}, t_n, t_{n-1}, t_{n-2}] = \frac{1}{\Delta t_{n+1} + \Delta t_n + \Delta t_{n-1}} \left[\frac{\mathbf{v}_{n+1} - \mathbf{v}_n}{\Delta t_{n+1} + \Delta t_n} - \frac{\mathbf{v}_n - \mathbf{v}_{n-1}}{\Delta t_n + \Delta t_{n-1}} \right] \tag{14.88}$$

By assuming

$$\mathbf{e}^{pred}_{n+2} = \mathbf{e}^{pred}_{n+1} \tag{14.89}$$

a new time step may be estimated from the solution of the cubic equation

$$\Delta t^{tr}_{n+2} \, (\Delta t^{tr}_{n+2} + \Delta t_{n+1})(\Delta t^{tr}_{n+2} + \Delta t_{n+1} + \Delta t_n) = \frac{1}{C} \tag{14.90}$$

where

$$C^2 = \sum_{i=1}^{neq} (u_i[t_{n+1}, t_n, t_{n-1}, t_{n-2}])/U_i^2 \tag{14.91}$$

with

$$U_i = tol_1 \, |u_{i,n+1}| + tol_2 \, u_{max} \quad . \tag{14.92}$$

where tol_1 and tol_2 are two relative and absolute tolerences on error and

$$u_{max} = \max(\bar{u}_{n+1}, \bar{u}_n, \bar{u}_{n-1}, \bar{u}_{n-2}) \tag{14.93}$$

with $\bar{u}_k$ the maximum absolute value of a component in the solution at time t_k. Typical values for the tolerances are

$$tol_1 = 5 \times 10^{-3} \quad \text{and} \quad tol_2 = 5 \times 10^{-5} \quad . \tag{14.94}$$

The time step is not allowed to grow rapidly by imposing the constraint

$$\Delta t_{n+2} = \min(\Delta t^{tr}_{n+2}, 4 \, \Delta t_{n+1}) \tag{14.95}$$

14.7 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 ACTIvated or DEACtivated 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).

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

$$\mathbf{F}(t) = p_j(t)\,\mathbf{F}_j \tag{14.96}$$

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; and

3. The proportional loading function $p_j(t)$.

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

$$p(t) = a_0 + a_1 t + a_2[\sin a_3(t - t_{min})]^k \quad ; \quad t_{min} \le t \le t_{max} \tag{14.97}$$

and is input by the statements

```
PROP,,J1
 ...
END
1 K T-min T-max A0 A1 A2 A3
```

This is called Type 1 loading and requires a 1 in the first column defining the parameters. *A blank record* is considered to be a Type 1 loading with default parameters:

$$t_{min} = 0 \ ; \ t_{max} = 10^6 \ ; \ a_1 = 1 \ ; \ k = a_0 = a_2 = a_3 = 0 \ . \tag{14.98}$$

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:

$$\begin{array}{llllll} 2 & \text{n} & & & & \\ t_1 & p_1 & t_2 & \cdots & t_n & p_n \\ t_{n+1} & p_{n+1} & \cdots & \cdots & t_{2n} & p_{2n} \\ \cdots & \cdots & & & & \end{array}$$

by default $n = 1$ and the values appear as time-value pairs on each record. *Input terminates with a blank record.*

14.8 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 algorithm for an arclength solution is given by:

```
ARCLength
DT,,delta-t
LOOP time n-steps
  TIME
  LOOP newton n-iters
    TANGent,,1
  NEXT iteration
   ..... (outputs, etc.)
NEXT,time
```

Remark: It is *not* permitted to use a PROPortional loading command with the arclength procedure.

14.9 Augmented Solutions

FEAP has options to employ penalty method solutions to enforce CONStraints. A penalty method is used as an option of the GAP element, to enforce incompressibility constraints in some of the continuum elements and to enforce the gap in satisfaction of contact constraints. 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 on on pp. 323 of *The Finite Element Method, Vol 1*, 5th ed., by Zienkiewicz & Taylor [1] (or on pp. 358 of *The Finite Element Method, Vol 1*, 4th ed., by Zienkiewicz & Taylor[13]). 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 `TANG`ent (or `UTAN`) command. If convergence is achieved in this loop execution passes to the `AUGM`ent command and another augmentation is performed until the `n-augm` augmentation iterations are performed.

14.10 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.

14.11 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 14.3. 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 14.3 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: stored energy; 8: kinetic energy; 9: dissipated energy; and 10: total energy.

The components output by the STREss option in the TPLOt command are indicated in Table 14.4.

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 14.3: 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	q	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}	
5	σ_{23}	M	M_2		M_{22}	M_{22}	S_{23}	
6	σ_{31}	χ	M_3			M_{12}		
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 14.4: Components for `STREss` option

14.12 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.

14.13 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.

14.14 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 14.9 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.

14.15 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. For example, after a `TANG`ent command the use of `EPRI`nt would display the element tangent matrix (e.g., stiffness) and residual vector for this element. This option works for both symmetric and unsymmetric 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	ESTR	EWIR	EXNO	EYES	FACT	FILL
FLUX	FRAM	FULL	HIDE	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	SCAL	SCRE
SHOW	SIZE	SNOD	SPHE	SPLO	STRE	SWEE	SWIR
SYMM	TEXT	TIME	TITL	TORS	UNDE	UPLO	VELO
VWIR	WIND	WIPE	XYPL	ZOOM			

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.

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 Clear graphics screen

The plot screen may be erased at any time using the command

```
WIPE
```

Alternatively, the central portion of the screen may be erased using the command

```
CLEAr
```

15.1.2 Mesh plots

The command

```
PLOT MESH
```

will display a wireframe 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.

15.1.3 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.

15.1.4 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.

15.1.5 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 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.

15.1.6 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 `DISPlacement`.

15.2.1 Displacement contours

Contours of the displacements within each element may be constructed using

```
        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.

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.

To construct contours the stress values are first projected to the nodes. 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 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 `SUBS`pace solution 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 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 `feapx.eps`, where `x` is a letter between the lower case `a` and the upper case `Z` (52 files may be made - only 26 in PC mode since the difference in upper and lower cases is ignored). 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.

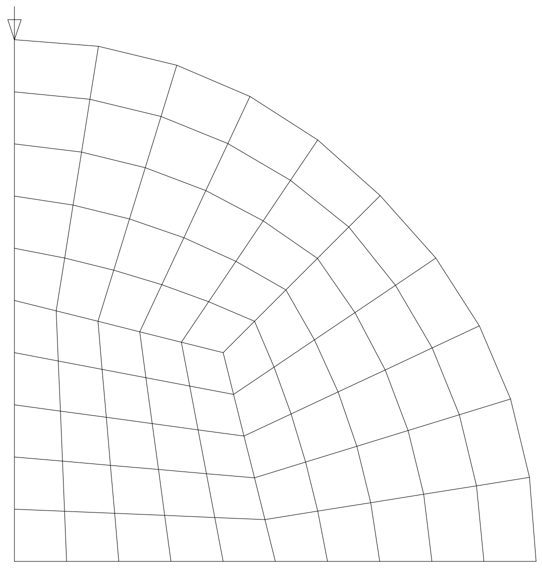


Figure 15.1: Mesh for Circular Disk. 75 Elements

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 15.1 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 `CONT`our 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 `CONT`our command is replaced by `VELO`city or `ACCE`leration, 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.
```

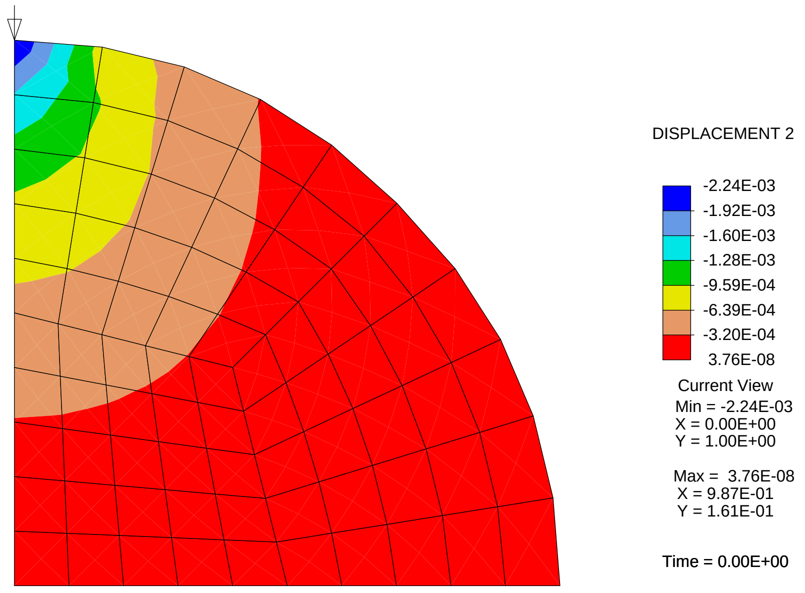


Figure 15.2: Contours of Vertical Displacement for Circular Disk

```
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

```
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.

[1]Note: The `DEFAult` command is at the intermediate level and will not appear if the `HELP` command is given at the basic level (i.e., `MANUal`= 0).

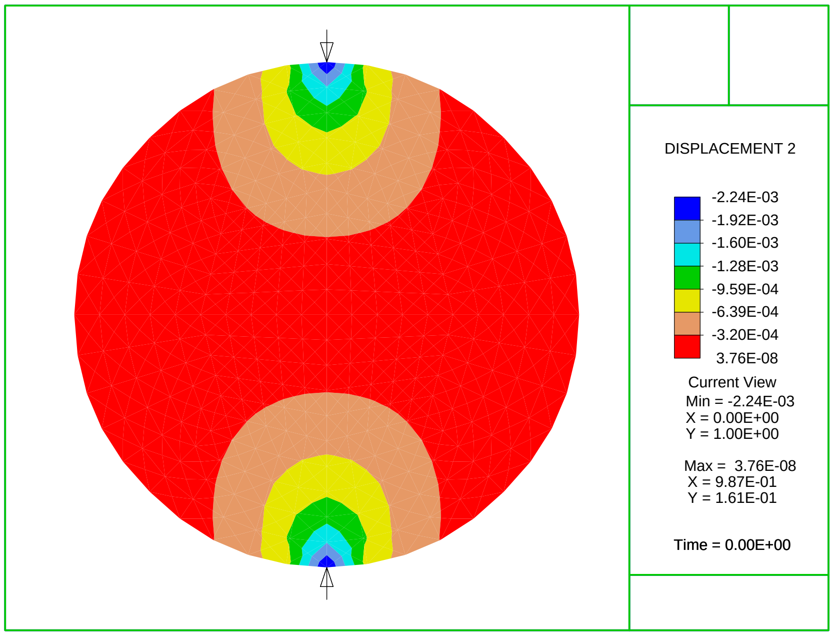


Figure 15.3: Contours of Vertical Displacement for Circular Disk

The need to specify parameters depends on settings of these parameters at installation time.

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 twenty plus years. 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. The basic strategies for solving non-linear problems resulted from contributions by Juan during our years of interactions. Element technology for finite deformation solid elements for the mixed and enhanced strain are based on the insights of Juan, 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 Juan and further developed by Dr. Alecia Chen. Juan also added the rotational update routines involving quaternions to support the structural elements and the rigid body work.

Additional improvements to *FEAP* resulted from contributions by present and former students, visiting scholars, and users of earlier versions of the program. Listed in alphabetical order the contributors were: Ferdinando Auricchio (University of Pavia, Italy), Ushnish Basu (Structural Mechanics, University of California, Berkeley), David Bindel(Computer Science, University of California, Berkeley), Jerry Goudreau (Lawrence Livermore National Laboratory), Anna Haraldsson (Hannover University of Technology, Germany), Peter Helnwein, (Institute for Strength of Materials, Technical University Vienna, Austria), Tom Hughes (The University of Texas, Austin), Eric Kasper

(California State Polytechnic University, San Luis Obispo), Tod Larsen (Duke University), Barham Nour-Omid, Karl Schweitzerhof (Institute for Mechanics, University of Karlsruhe, Germany), Tom Spelce, Prashanth Vijalapura (Structural Mechanics, University of California, Berkeley), Peter Wriggers (Hannover University of Technology, Germany), Giorgio Zavarise (University of Torino, Italy),

Many additional contributions and suggestions for improvements have been made by Berkeley colleagues Francisco Armero, Jon Bray, Greg Fenves, Filip Filippou, Sanjay Govindjee, and Panayiotis (Panos) Papadopoulos are gratefully acknowledged.

The careful reading of the Version 7.4 manual by Dr. Damir Lazarevic of the University of Zagreb in Croatia has led to several improvements and corrections to this manual. 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 Professor Olek Zienkiewicz during the last thirty plus years. His insights and contributions have 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
4 August 2003

Bibliography

[1] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method: The Basis*, volume 1. Butterworth-Heinemann, Oxford, 5th edition, 2000.

[2] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method: Solid Mechanics*, volume 2. Butterworth-Heinemann, Oxford, 5th edition, 2000.

[3] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method: Fluid Mechanics*, volume 3. Butterworth-Heinemann, Oxford, 5th edition, 2000.

[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 edition, 2001.

[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. *FEAP - A Finite Element Analysis Program, Programmer Manual.* University of California, Berkeley. http://www.ce.berkeley.edu/~rlt.

[13] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method*, volume 1. McGraw-Hill, London, 4th edition, 1989.

[14] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method*, volume 2. McGraw-Hill, London, 4th edition, 1991.

[15] R.L. Taylor. *FEAP - A Finite Element Analysis Program, Theory Manual.* University of California, Berkeley. http://www.ce.berkeley.edu/~rlt.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] R.M. Christensen. *Theory of Viscoelasticity: An Introduction.* Academic Press, New York, 1971 (Reprinted 1991).

[23] 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.

[24] R.W. Ogden. *Non-linear Elastic Deformations.* Ellis Horwood, Limited (reprinted by Dover, 1997), Chichester, England, 1984.

[25] 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.

[26] 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.

[27] 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.

[28] J.C. Simo and T.J.R. Hughes. *Computational Inelasticity*, volume 7 of *Interdisciplinary Applied Mathematics*. Springer-Verlag, Berlin, 1998.

[29] H. Matthies and G. Strang. The solution of nonlinear finite element equations. *International Journal for Numerical Methods in Engineering*, 14:1613–1626, 1979.

[30] N. Newmark. A method of computation for structural dynamics. *J. Engineering Mechanics, ASCE*, 85:67–94, 1959.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] A.K. Chopra. *Dynamics of Structures.* Prentice-Hall, Upper Saddle River, N.J., 1995.

Appendix A

Mesh Manual Pages

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
   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 FEAP MESH INPUT COMMAND 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 $ndf \times nen + nad$.
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,PROF`ile 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):

$$node1,\ node1{+}ngen1,\ node1{+}2{\times}ngen1,\ \ldots.\ ,\ node2$$

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.

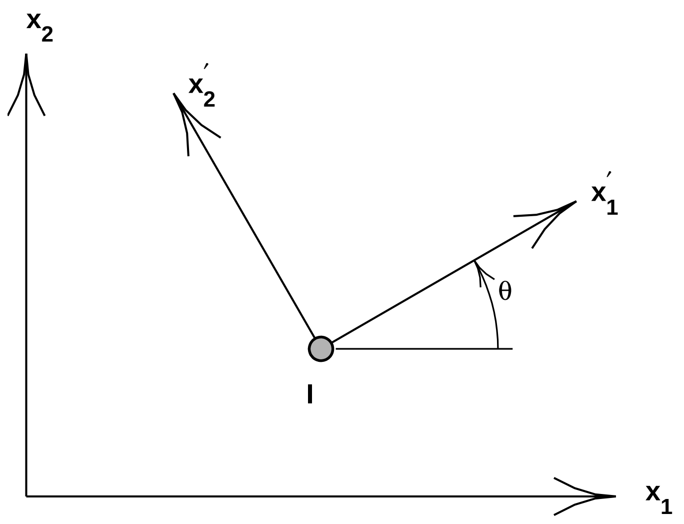


Figure A.1: Coordinate rotation for nodes

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
   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):

$$node1,\ node1+ngen1,\ node1+2\times ngen1,\ ....\ ,\ node2$$

```
      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 finite element book, 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 `BLOC`k 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 `BLEN`d command.

Examples for two and three dimensional blends are illustrated in the *FEAP* User Manual.

Since the description of the `BLEN`d command depends on existence of `SNOD`e 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.

```
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 `BLOC`k data input segment is used to generate:

1. 2-node line elements in 1, 2, or 3-D.
2. 4 to 9-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-node hexahedra (bricks) in 3-D.
5. 4-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 `BLOC`k 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 `BLOC`k 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/side nodes are optional, as is the center node (numbering for nodes is shown in Figures A.3, A.4 and A.5).

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.

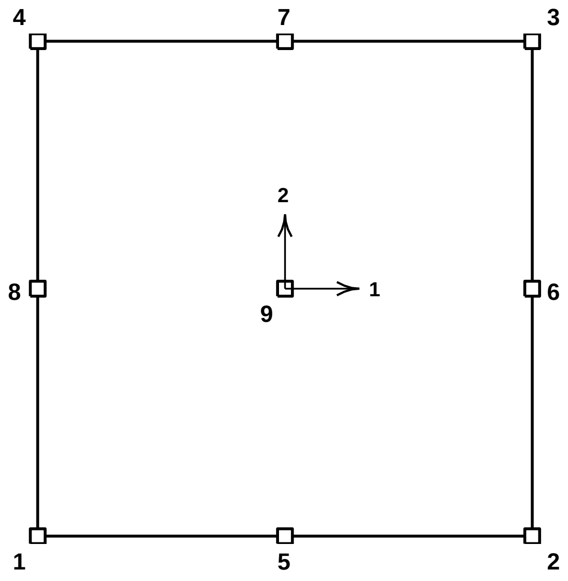


Figure A.2: Node Specification on 2D Master Block.

The data parameters are defined in Tables A.1 and A.2.

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 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.

type	– Master node coordinate type (*cart, pola,* 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. 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 Numbering Data

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);
	=10:	8-node hexahedra (bricks).
	=11:	4-node tetrahedra.

Table A.2: Block Type Data

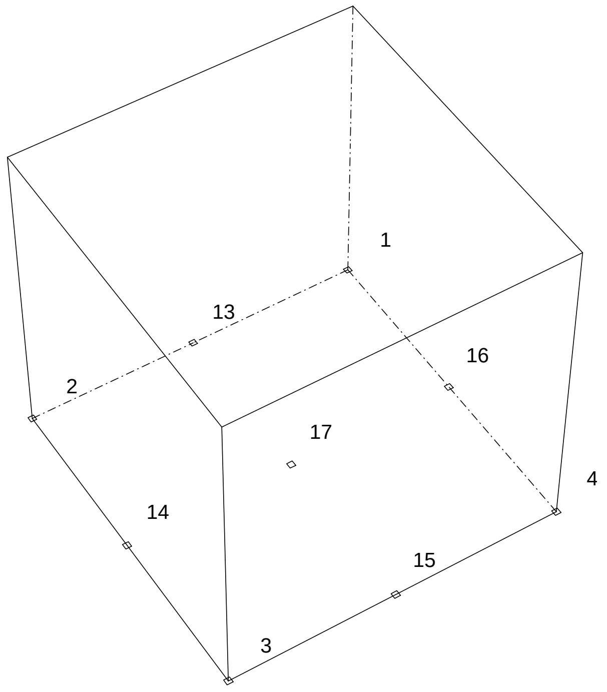


Figure A.3: Node Specification on 3D Master Block.

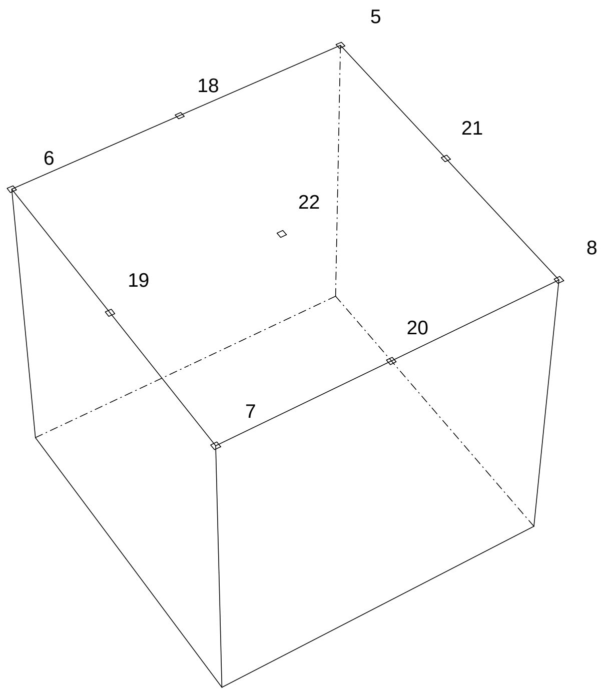


Figure A.4: Node Specification on 3D Master Block.

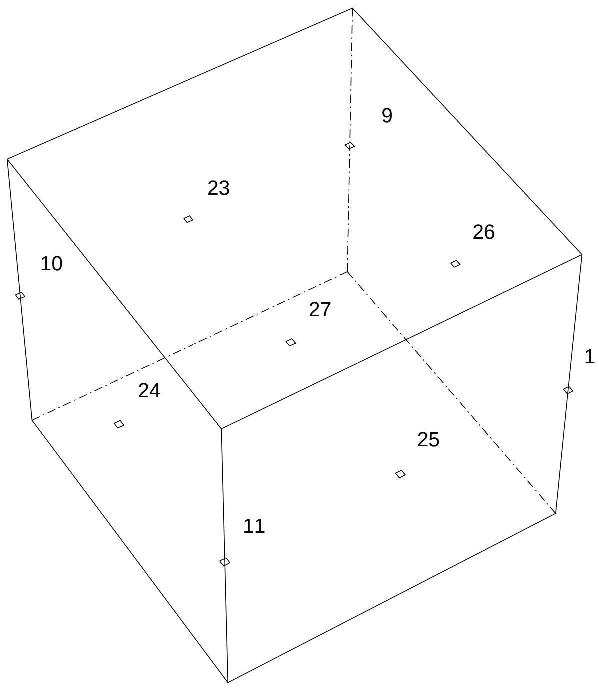


Figure A.5: Node Specification on 3D Master Block.

```
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:

$$\begin{array}{lcll} \textit{id(i,node)} & = & 0 & \text{a force will be an applied load to dof (default).} \\ \textit{id(i,node)} & \neq & 0 & \text{a displacement will be imposed to dof.} \end{array}$$

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown at top):

$$node1,\ node1+ngen1,\ node1+2\times ngen1,\ ....\ ,\ node2$$

The values for each boundary restraint will be as follows:

$$\begin{array}{lcl} \textit{id(i,node1)} = & \text{0 or positive} & \rightarrow \text{id(i,node1+ngen1)} = \ \ 0 \\ \textit{id(i,node1)} = & \text{negative} & \rightarrow \text{id(i,node1+ngen1)} = -1 \end{array}$$

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 `EBOUnd` or `CBOUnd` 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,x1,t1
    2,x2,t2
  etc.,until all 'nodes' records are input
```

The `BTEM`perature data input segment is used to generate 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). If thermal problems are solved by *FEAP* temperatures are *generalized displacements*. Boundary temperatures should then be specified using `DISP`, `EDIS` and/or `CDIS` commands. Initial conditions are specified using the `INIT,DISP` solution command.

The temperatures using `BTEM` are generated by interpolating specified nodal temperatures using the standard isoparametric interpolation:

$$T = N_I(\boldsymbol{\xi})\, T_I$$

where $N_I(\boldsymbol{\xi})$ are the shape functions, $\boldsymbol{\xi}$ are the natural coordinates (ξ_1, ξ_2, ξ_3), and T_I is the temperature at node-I.

For two dimensions, the patch of nodes defined by `BTEM`perature is developed from a master element which is defined by an isoparametric 4-9 node mapping function in terms of the natural coordinates r (for ξ_1) and s (for ξ_2). The node numbers on the master element of each patch defined by `BTEM` are specified according to Figure A.2 in the `BLOC`k 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 `BLOC`k 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 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 (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:

```
CANGle
  QUADratic
    1,x1,y1,angle1
    2,x2,y2,angle2
    3,x3,y3,angle3
```

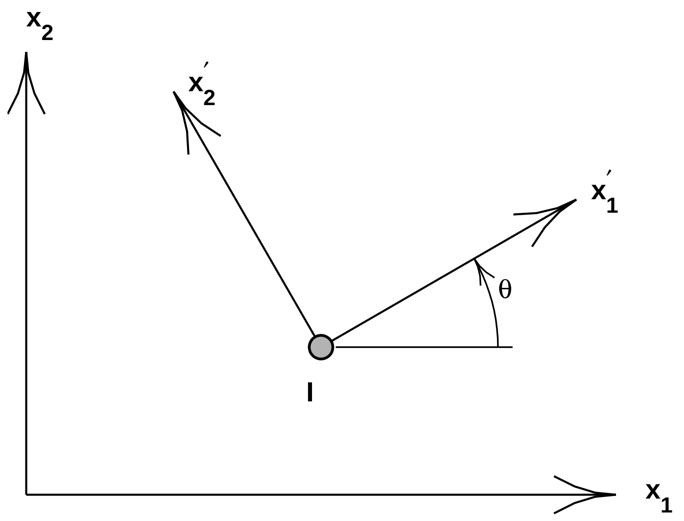


Figure A.6: Coordinate rotation for node I

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,BOUN`dary 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
```

```
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.

```
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 `DIS`Placement 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.

```
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 `FORC`e processed first, `EFOR`ce processed second and the `CFOR`ce data processed last. The value defined last is used for any analysis.

The `CFOR`e 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.

CPROportional FEAP MESH INPUT COMMAND MANUAL

```
    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 `FPRO`portional processed first, `EPRO`portional processed second and the `CPRO`portional 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.

```
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):

$$node1,\ node1+ngen1,\ node1+2\times ngen1,\ ....\ ,\ node2$$

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.

```
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,component
   normal
   tangential
   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 or displacements 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*, *normal*, or *tangential* options. These specify that inputs will be a specific displacement component, normal tractions (pressures), or tangential tractions (shears), respectively. The default is *normal* loading. For displacement inputs the component to be generated is specified immediately after the *displacement* command.

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 segment to which the quantities are applied. The segment is specified as a *surface*. The data is 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):

$$node1,\ node1+ngen1,\ node1+2\times ngen1,\ ....\ ,\ node2$$

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 INPUT 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):

$$node1,\ node1{+}ngen1,\ node1{+}2{\times}ngen1,\ ....\ ,\ node2$$

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.

```
ebou,[set,add]
   i-coor,xi-value,(ibc(j),j=1,ndf)
   <etc.,terminate with a blank record>
gap value
```

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 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 `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: EBOUndary

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.

```
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 `FORCe` 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: 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.

```
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,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 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 `ELEM`ent input or during a `BLOC`k and `BLEN`d generation. During `ELEM`ent input it is necessary to input the last element in generation sequences.

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 `*ELE` value.

END FEAP MESH INPUT 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`, `ELIN`k, `PART`ition `ORDE`r, `RIGI`d and `JOIN`t should be given prior to initiation of a problem solution using `BATC`h and/or `INTE`ractive.

```
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 `FPRO`p and `CPRO`p commands. The data is order dependent with data defined by `FPRO`p processed first, `EPRO`p processed second and the `CPRO`p 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 `EREG`ion 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

$$elem1,\ elem1{+}ngen1,\ elem1{+}2{\times}ngen1,\ ....\ ,\ elem2$$

The generated element are assigned to *reg1*.

Region numbers may also be assigned to element groups using the `REGI`on mesh command.

```
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 `FORC`e 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:

$$node1,\ node1+ngen1,\ node1+2\times ngen1,\ ....\ ,\ node2$$

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 `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.

Force conditions may also be specified using the `EFOR`ce 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: 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 `FPRO`portional 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
   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)
```

The GLOBal command is used to set parameters which apply to all elements. 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.

Global parameters may be superseded by specifying a different condition during input of MATErial commands.

INCLude FEAP MESH INPUT COMMAND MANUAL

```
    incl,filename
```

The `INCL`ude 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.

```
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 INPUT 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):

$$node1,\ node1{+}ngen1,\ node1{+}2{\times}ngen1,\ ....\ ,\ node2$$

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 `MATE`rial 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:

SOLId	Continuum solid mechanics element (2 or 3-D).
THERmal	Continuum thermal element (2 or 3-D).
FRAMe	2-Node frame element (2 or 3-D).
TRUSs	2-Node truss element (1, 2, or 3-D).
PLATe	2-d Plate bending element.
SHELl	3-d Shell element.
MEMBrane	3-d Membrane element.
GAP	n-d Gap element.
PRESsure	3-d Pressure load element (dead or follower).

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 `ELEM`ent or `BLOC`k 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

```
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.

NEXT FEAP MESH INPUT COMMAND MANUAL

```
    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 INPUT 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 `NOPR`int 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 `PRIN`t will cause the mesh information to again be reported in the output file. The default value is `PRIN`t at the start of each problem execution.

```
para
   x = expression
```

The use of the `PARA`meter command may be used to assign values to letter parameters. A letter parameter is defined immediately following the `PARA`meter 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 ˆ. 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 `PARA`meter 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 `PARA`meter 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 INPUT COMMAND MANUAL

```
pars
```

The `PARS`e 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 `NOPA`rse command. If subsequent data must be parsed, a `PARS`e command may be required to produce the correct results.

POLAr FEAP MESH INPUT COMMAND MANUAL

```
pola
   node,node1,node2,inc
   all
   <terminate with blank record>
```

The `POLA`r 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 `SHIF`t 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 `POLA`r command. Execution terminates with a blank record.

The option *all* perform the operation on all currently defined nodes.

PRINt FEAP MESH INPUT COMMAND MANUAL

```
    prin
```

The use of the `PRIN`t command will cause the description of most information produced during the mesh description to be placed in the *FEAP* output file. The use of `NOPR`int will discontinue the output of mesh information (except for data printed in elements). The default value is `PRIN`t.

REACtion FEAP MESH INPUT COMMAND MANUAL

```
reac,<filename>
```

The use of the `REAC`tion command permits the retrieval of reaction data which was saved using the `REAC,filename` 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.

```
    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.

REGIon FEAP MESH INPUT COMMAND MANUAL

```
    regi,nreg
```

The `REGI`on 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 `ACTI`vate or `DEAC`tivate parts of the mesh during execution.

```
    rese
```

Use of the `RESE`t command will reinitialize all the boundary condition codes to have no restraints imposed on the degrees-of-freedom. Thus, all the degrees-of-freedom become unknowns for the problem. The command is useful when boundary conditions are to be changed from *displacement* to *force* states during execution. After the use of the `RESE`t command, boundary conditions for specified *displacement* conditions may be reimposed using `BOUN`dary, `EBOU`dary, or `CBOU`ndary 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\_body
  ELEMents
    1,.....
                  ! Blank terminator
```

To specify an additional rigid body another `RIGI`d 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,<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 `SBLO`ck 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 `SBLO`ck 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 `SBLO`ck 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 `SBLO`ck. 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)

SHIFt FEAP MESH INPUT COMMAND MANUAL

```
shif
  x0,y0,z0
```

The `SHIF`t command is used to specify the values for the origin of polar and spherical coordinate transformations (used by commands `POLA`r, `SPHE`rical, or `BLOC`k). The input of *x0*, *y0*, and, for three dimensional problems, *z0* are in cartesian values based on the reference mesh coordinate distances.

```
    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 finite element book, 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
   (iel(i),i=1,ns),(p(i),i=1,nv)
   <etc.,terminate with blank record>
```

The `SLOA`d 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.

```
    snod
      snode1,(x(i,snode1),i=1,ndm)
      snode2,(x(i,snode2),i=1,ndm)
      <etc.,terminate with blank record>
```

The `SNOD`e 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 finite element book, 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 `SPHE`rical 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 `SHIF`t 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 `SPHE`rical 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):

$$node1,\ node1+ngen1,\ node1+2\times ngen1,\ ....\ ,\ node2$$

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.

```
temp
   node1,ngen1,t(node1)
   node2,ngen2,t(node2)
   <etc.,terminate with blank record>
```

The `TEMP`erature command is used to specify the values for nodal temperatures. 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 temperature for node.

When generation is performed, the node number sequence will be (for *node1-node2* sequence shown at top):

node1, node1+ngen1, node1+2$\times$*ngen1, ... , node2*

The values for each temperature will be a linear interpolation between *node1* and *node2*.

```
     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.

```
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.

```
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 `TRIB`lock 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 `TRIB`lock 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 `TRIB`lock 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 `TRIB`lock. 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, 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.

```
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 `ELIN`k command may be used for this purpose.

To use the `ELIN`k option the complete mesh must first be defined. After the `END` command for the mesh definition and before the `BATC`h, or `INTE`ractive command for defining a solution algorithm, the use of a `ELIN`k 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 (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 `MAST`er 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 `surf`ace 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
```

```
part
  part_1,part_2, ...  part_ndf
```

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 `PART`ition command between the mesh `END` command and the first problem solution command `BATC`h or `INTE`ractive.

The `PART` command is followed by a record which denotes the partition number for each DOF. 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
  2  2  1
```

Note that a DOF can belong to only one partition; thus, general staggering algorithms may not be considered in the present implementation in *FEAP*.

```
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).

```
rdis
   body,comp_1,comp_2, ...  ,comp_ndf
```

This command is currently inactive.

Users may specify displacements to be applied to each body using the `RDIS` command which must appear in the data file after the `END` mesh command and before the first `BATC`h or `INTE`ractive solution command.

The displacement components will be interpreted according to the boundary restraint codes set for each rigid body (see manual page for `RBOU`). Components will be ignored if the restraint code associated with the DOF is zero.

Rigid bodies may be interconnected using joints (see `JOIN`t mesh manipulation command). They may also be loaded and restrained at their center of mass (see `RLOA`d and `RBOU`ndary 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 `RIGI`d 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 `RIGI`d, which must appear in the data file after the `END` mesh command and before the first `BATC`h or `INTE`ractive solution comomand.

Rigid bodies may be interconnected using joints (see `JOIN`t mesh manipulation command). They may also be loaded and restrained at their center of mass (see `RLOA`d and `RDIS`placement 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).

```
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 `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.

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 INPUT 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`, `ELIN`k, `PART`ition `ORDE`r, `RIGI`d and `JOIN`t can be given prior to initiation of a problem solution using `BATC`h and/or `INTE`ractive.

```
    mate number
       standard
         friction coulomb value                user coulomb value
```

The `MATE`rial command is used to define properties for contact interaction. Only one option is currently available: the standard option denoted by the data `STAN`dard. If the `MATE`rial command is not included as part of the contact data a standard model without friction is assumed. Friction may be added by including the `FRIC`ion command record with the parameters `COUL`omb and a *value* of the frictional coefficient.

```
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
```

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 `SURF`ace 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 `MATE`rial 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 `TOLE`rance 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.

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.

```
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 `SURF`ace 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.

```
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.

ACCEleration FEAP COMMAND INPUT COMMAND MANUAL

```
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.

ACTIvate FEAP COMMAND INPUT COMMAND MANUAL

```
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 following table of input options are allowed.

xxxx	`kfl`	`lfl`
	0 to 5	0 or 1
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

$$\boldsymbol{u} \leftarrow \boldsymbol{u} + \frac{\mid \boldsymbol{u} \mid}{\mid \boldsymbol{Ev}_{n1} \mid \, \tau} \boldsymbol{Ev}_{n1} \tag{D.1}$$

where $\boldsymbol{u}$ is the current solution and $\boldsymbol{Ev}_{n1}$ is the $n1$-eigenvector.

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.

ARCL 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 `PROP`ortional load using default parameters; the actual load level is computed by `ARCL`.

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

$$\tau = 100 \frac{(\boldsymbol{u} \cdot \boldsymbol{E}\boldsymbol{v}_{n1})}{\mid \boldsymbol{u} \mid \mid \boldsymbol{E}\boldsymbol{v}_{n1} \mid} + 1 \tag{D.2}$$

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
```

AUGMent FEAP COMMAND INPUT COMMAND MANUAL

```
    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,imin,imax,maxr
 auto,dt,dtmin,dtmax
auto,mate                            auto,off
```

The `AUTO` command provides for automatic time step control, based on iteration properties or on a material response indicator.

The form based on iteration limits is given as:

```
AUTO,TIME,IMIN,IMAX,MAXR
```

where the parameter `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 the time step is reduced, whereas when the iteration falls below the lower limit the time step is increased.

Example use:

```
LOOP,,500
  TIME,,50.0
  AUTO,time,4,9,5
  LOOP,,15
    TANG,,1
  NEXT
NEXT
```

Use of the command:

```
AUTO,DT,DTMIN,DTMAX
```

limits the range of the auto time stepping 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.

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 FEAP COMMAND INPUT COMMAND MANUAL

```
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 FEAP COMMAND INPUT COMMAND MANUAL

```
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.

CAPTion FEAP COMMAND INPUT COMMAND MANUAL

```
capt,name
```

Use of the `CAPTion` command will allow the label of a subsequent contour plot to be set as the label `NAME`. The command must be given *before* each contour plot, even if the plot type is to be repeated.

Example:

```
....
CAPTION Mises_Stress
PLOT PSTRE 6
....
```

In an interactive mode of execution the caption may also be set using the `PLOT CAPTion` command.

CHECk mesh FEAP COMMAND INPUT COMMAND MANUAL

```
chec
```

The `CHEC`k 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 `CHEC`k 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.

COMMent FEAP COMMAND INPUT COMMAND MANUAL

```
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.

CONTact FEAP COMMAND INPUT COMMAND MANUAL

```
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.

```
    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.

```
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 COMMAND INPUT 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.

DEACtivate FEAP COMMAND INPUT COMMAND MANUAL

```
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 FEAP COMMAND INPUT 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 the `DEBUg,OFF` command.

```
dire
dire,bloc,v1
dire,spar
```

The `DIRE`ct command sets the mode of solution to direct for the linear algebraic equations generated by a `TANG`ent or a `UTAN`gent 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 `DIRE`ct,`BLOC`k,`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 using memory from the blank common. 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 `ITER`ative command language manual page).

```
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 `DIS`Placement 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.

DT FEAP COMMAND INPUT 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.

EIGElement FEAP COMMAND INPUT COMMAND MANUAL

```
eige
eige,vect
```

The use of the `EIGE`lement command permits the computation of the eigenvalues associated with the last computed element tangent array. It is assumed that the array is symmetric and has real eigenvalues. This option is useful during element development to study the spectral properties of the element, including number of zero eigenvalues or those associated with some parameter. Use of `EIGE,VECT` reports both the eigenvalues and eigenvectors for the last element.

```
eigv,nn,<n1,n2,n3>
eigv,coor,idir,xi,nn
eigv,list,n1,nn
eigv,all,n1,nn
eigv,dofs,<list>
```

The command `EIGV`ector 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 FEAP COMMAND INPUT COMMAND MANUAL

```
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 FEAP COMMAND INPUT COMMAND MANUAL

```
    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.

```
    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`.

EPRInt FEAP COMMAND INPUT COMMAND MANUAL

```
    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 `ERRO`r 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 FEAP COMMAND INPUT COMMAND MANUAL

```
    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.

EXPLicit FEAP COMMAND INPUT COMMAND MANUAL

```
expl
```

The use of the `EXPL`icit 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 `TRAN`sient 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. 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.

EXPOrt FEAP COMMAND INPUT COMMAND MANUAL

```
    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.

FORM FEAP COMMAND INPUT COMMAND MANUAL

```
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 `EXPL`icit option is present *FEAP* computes a solution to the equations of motion (momentum equations) using an explicit solution option. Prior to using the `FORM,EXPL`icit command it is necessary to specify the explicit solution option using the `TRAN`sient,`EXPL`icit 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.

FUNCtion FEAP COMMAND INPUT COMMAND MANUAL

```
    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.

HELP FEAP COMMAND INPUT COMMAND MANUAL

```
help
```

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.

HISTory FEAP COMMAND INPUT COMMAND MANUAL

```
hist,<clab,n1,n2>
```

The use of the `HIST`ory 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.

IDENtity FEAP COMMAND INPUT COMMAND MANUAL

```
    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 COMMAND INPUT 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`.

IMPOrt FEAP COMMAND INPUT COMMAND MANUAL

```
    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 `INIT`ial 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.

ITERative FEAP COMMAND INPUT COMMAND MANUAL

```
iter
iterbpcg,v1
iterppcg,v1
```

The `ITER`ative command sets the mode of solution to iterative for the linear algebraic equations generated by a `TANG`ent. Currently, iterative options exist only for symmetric, positive definite tangent arrays, consequently the use of the `UTAN`gent command must be avoided. An iterative solution requires a sparse matrix form of the tangent matrix to fit within the blank common array dimensioned in the main program (see programmer manual for procedures to reset the size of the blank common array).

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 `ITER`ative,`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 `ITER`ative,`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 non-linear (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 `DIRE`ct command language manual page).

LIST FEAP COMMAND INPUT COMMAND MANUAL

```
mate list,,n1
   <values>
```

The command `LIST` is used to specify lists of nodes for outputs. It is possible to specify up to three different lists where the list number corresponds to `n1` (default = 1). The list of nodes to be output follows with 8 values per record. The input terminates when less than 8 values are specified.

List outputs are then 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
```

LOOP FEAP COMMAND INPUT COMMAND MANUAL

```
loop,<xxxx>,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 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.

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 COMMAND INPUT 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 FEAP COMMAND INPUT COMMAND MANUAL

```
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.

MATErial FEAP COMMAND INPUT COMMAND MANUAL

```
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).

MESH FEAP COMMAND INPUT COMMAND MANUAL

```
    mesh
```

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 provided *FEAP* is solving problems in a `CHAN`ge mode.

```
    moda
```

The solution of transient linear problems may be performed using either the time stepping algorithms defined by the `TRAN`sient command language statement or using mode superposition using the `MODA`l 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 `INIT`ial 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 `TPLO`t) by inserting the command:

```
      TPLOt
```

before the first time loop.

MONOlithic solution FEAP COMMAND INPUT 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`.

NEWForce FEAP COMMAND INPUT COMMAND MANUAL

```
newf
newf,zero
```

The use of the `NEWF`orce 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:
$$\texttt{f0}(\texttt{i},\texttt{1}) = \texttt{f}(\texttt{i},\texttt{1}) * \texttt{prop}(\texttt{t}) + \texttt{f0}(\texttt{i},\texttt{1}) \tag{D.6}$$
2. For degree-of-freedoms where displacements are specified:
$$\texttt{f0}(\texttt{i},\texttt{2}) = \texttt{u}(\texttt{i}) \tag{D.7}$$
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.

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 `NEWF`orce had been used in generating the results.

The `f0(i,n)` may be reset to zero using the `NEWF`orce,`zero` command (values are not updated). N.B. This only affects the current partition degree of freedoms.

NEXT FEAP COMMAND INPUT COMMAND MANUAL

```
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 COMMAND INPUT COMMAND MANUAL

```
    nopr
```

The use of the `NOPR`int command will discontinue most output of commands. Plot results and element outputs will normally still be reported. The use of `PRIN`t will cause the output of execution descriptions to again be reported. The default value is `PRIN`t at start of command language program execution.

NTANgent FEAP COMMAND INPUT COMMAND MANUAL

```
ntan,mate,<n1>
ntan,elem,<n1>
ntan,off
```

Numerically compute the tangent matrix using residuals. Use of the `MATEerial` 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.

The `OFF` option is used to discontinue use of numerical computations of tangents.

```
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)

OUTMesh FEAP COMMAND INPUT COMMAND MANUAL

```
outm,<bina>
```

The use of the `OUTM`esh command writes an output file which contains some of the mesh data. Two modes of output are possible. Using the `OUTM`esh 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 `OPTI`mize 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,`BINA`ry 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 `OPTI`mize have been used. It also may be used on very large models which are time consuming to generate the input data.

PARAmeters FEAP COMMAND INPUT COMMAND MANUAL

```
para
 < After end command >
   letter = expression
   list
```

The use of the `PARA`meter 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.

PARTition FEAP COMMAND INPUT COMMAND MANUAL

```
part,,n1
part
```

The command PARTition 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 n1,n2,n3, etc.
```

where the ni are between 1 and 4 and denote the partition the degree of freedom to be assigned. 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 1,1,2
```

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 FEAP COMMAND INPUT COMMAND MANUAL

```
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.

PRINt FEAP COMMAND INPUT COMMAND MANUAL

```
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.

```
prop,,<n1>
prop,,<n1,n2>                    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:

   $$Prop(t) = a_1 + a_2\,(t - t_{min}) + a_3\,\left(\sin(a_4\,(t - t_{min}))\right)^k \tag{D.8}$$

   for all time values between `t_min` and `t_max`. The value of `k` must be a positive integer all other parameters are real.

   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.

User Function

The optional input `PROP,USER` requires modification of the program to function. It is necessary to write and compile into the program a subprogram:

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: Parameters for user proportional load

```
      SUBROUTINE UPROP(J,IE,TMIN,TMAX,A, T, UPRLD, ISW)
```

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. Users are also 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:

$$uprld(t) = C_0 \exp[-C_1\,(t - t_{min})]$$

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

```
      subroutine uprop(j,ie,tmin,tmax,a, t, uprld, isw)

c     User proportional load function

      implicit   none

      integer    j,ie, isw
      real*8     tmin,tmax,a(5), t, uprld

c     Function 1

      if(j.eq.1) then

c       Output parameters

        if(isw.eq.1) then
          write(iow,2000) tmin,tmax,a(1),a(2)

c       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

c     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: User proportional load

QUIT FEAP COMMAND INPUT COMMAND MANUAL

```
    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

$$\mathbf{C} = a_0\,\mathbf{M} + a_1\,\mathbf{K} \tag{D.9}$$

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 `.ren` (starting from `re0`). These may be used as input in Mesh (see Mesh `REAC`tion command).

READ FEAP COMMAND INPUT COMMAND MANUAL

```
read,xxxx
```

The `READ` command may be used to input the values of displacements and nodal stresses previously computed and saved using the `WRIT`e command - it is primarily used for plots related to deformations or nodal stresses. It is not intended for a restart option (see `REST`art) 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 `WRIT`e command which has identical options for `xxxx` except for the backspace option.

RENUmber FEAP COMMAND INPUT COMMAND MANUAL

```
    renu
```

The use of the `RENU`mber command writes to the output the renumbering map from `OPTI`mize together with the nodal coordinates.

RESTart FEAP COMMAND INPUT COMMAND MANUAL

```
rest,<fileext>
```

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 `fileext` option is used to restart with files generated using the `SAVE` command with the same specified `fileext`.

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 COMMAND INPUT COMMAND MANUAL

```
save,<fileext>
```

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 `fileext` is optional and may be any 1-4 alphanumeric characters and is appended to the name of the current restart save file which was named when the problem was started if specified. In interactive mode should a name be given which already exists the user receives a prompt for an alternate name or given the opportunity to write over the old file.

SCREen FEAP COMMAND INPUT COMMAND MANUAL

```
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`.

```
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.

SOLVe FEAP COMMAND INPUT COMMAND MANUAL

```
    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.

STREss FEAP COMMAND INPUT COMMAND MANUAL

```
stre,,<n1,n2,n3>
stre,all
stre,coor,idir,xi
stre.<node,n1,n2,n3>
stre,erro
```

The `STRE`ss 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 `STRE`ss,`ALL` is used.
2. For solid elements results may be reported at nodes using the `STRE`ss,`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.

With the `ERRO`r option `STRE`ss computes element sizes for adaptive mesh refinement. N.B. The error option does not function with all elements.

SUBSpace FEAP COMMAND INPUT COMMAND MANUAL

```
subs,<prin,n1,n2,stol>
```

The `SUBS`pace 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 `SUBS`pace command must be preceded by the specification of the tangent stiffness array using a `TANG`ent 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 `TANG`ent 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 `PRIN`t 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 `TANG`ent 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 `SUBS`pace 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`, `SOLV`e, `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 `TANG`ent,`LINE` plus options invokes the line search (it may also be used in conjunction with `SOLV`e,`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!

TIME FEAP COMMAND INPUT COMMAND MANUAL

```
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.

TOLerance FEAP COMMAND INPUT COMMAND MANUAL

```
tol,,v1
tol,ener,v1
tol,emax,v1
```

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 the subspace eigenpair solution which is measured in terms of the change in subsequent eigenvalues computed.

The default value of `TOL` is 1.0d-16.

The tol command also permits setting a value for the energy below which convergence is assumed to occur. The command is issued as `TOL,ENER`gy,`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 `TOL`erance value times the value of the energy from the first iteration. 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.

Finally, the tol command permits resetting 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. Since the `TIME` command sets the maximum energy to zero, the value of `EMAX`imum must be reset after each time step. Thus, a set of commands:

```
LOOP,time,n
  TIME
  TOL,EMAX,5.e+3
  LOOP,newton,m
    TANG,,1
  NEXT
  etc.
NEXT
```

is necessary to force convergence check against a specified maximum energy. The above two forms for setting the convergence are nearly equivalent; however, the `ENER`gy tolerance form can be set once whereas the `EMAX`imum form must be reset after each time command.

TPLOts FEAP COMMAND INPUT COMMAND MANUAL

```
     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
      sums,dof,dir,x,xtol
      ener
      show
```

The `TPLO`t 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 `TPLO`t 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 int 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 `DIS`placement 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 and each programmer must decide what is to be saved. Indeed the components need not be stresses, they may be strains, internal variables, etc.

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 `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. The Newmark-beta step-by-step integration of the equations of motion.
2. An Euler-backward difference method for first order ordinary differential equations such as heat transfer, etc.
3. Hilber-Hughes-Taylor alpha method for second order systems.
4. An explicit implementation of Newmark.
5. An energy conserving generalized mid-point method for second order systems.
6. A generalized mid-point method for static problems.
7. A generalized mid-point method for first order systems.

The `OFF` option 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. Newmark Method (`name` is `newm` or blank)

   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.

2. Backward Euler (`name` is `back`)

   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. HHT Alpha Method (`name` is `alph`)

   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 papers) in momentum equation:

   $$\boldsymbol{M}\,\boldsymbol{a}(t_{n+1}) + \boldsymbol{N}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha}) \tag{D.10}$$

   where $\boldsymbol{N}$ 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.

4. Explicit Newmark Method (`name` is `expl`)

   This option permits the explicit form of the Newmark method to be implemented. The input parameter is only

v2 =	gamma (default = 0.5)

5. Conserving Alpha Method (`name` is `cons`)

   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,

   $$\frac{1}{\Delta t}\boldsymbol{M}(\boldsymbol{v}(t_{n+1}) - \boldsymbol{v}(t_n)) + \boldsymbol{N}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha}) \tag{D.11}$$

   where $\boldsymbol{N}$ 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.

6. Static Alpha Method (`name` is `stat`)

   The method requires the specification of the alpha parameter for the momentum equation

   $$\boldsymbol{N}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha}) \tag{D.12}$$

   where $\boldsymbol{N}$ is the nonlinear internal force term.

v3 =	alpha (default = 0.5) Alpha should be between 0.5 and 1.

7. Alpha Method for First Order Systems(`name` is `gen1`)

   The method requires the specification of the alpha parameter for the equation,

$$\boldsymbol{M}\boldsymbol{v}(t_{n+\alpha}) + \boldsymbol{N}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha}) \tag{D.13}$$

   where $\boldsymbol{N}$ is the nonlinear internal force term.

v3 =	alpha (default = 0.5) Alpha should be between 0.5 and 1.

It is possible to specify nonzero values for the initial velocity in second order system integrators using the command `INIT`ial (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 `UTAN`gent 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 `SUBS`pace algorithm, which is restricted to symmetric tangents only (see `TANG`ent 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 `SOLV`e, 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 `UTAN`gent 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 COMMAND INPUT COMMAND MANUAL

```
writ,xxxx
```

The `WRIT`e 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 `REST`art 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 `WRIT`e output is reinput using the `READ` command which has nearly identical options for `xxxx`.

ZERO FEAP COMMAND INPUT COMMAND MANUAL

```
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 FEAP COMMAND INPUT COMMAND MANUAL

```
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 INPUT 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.

AXIS FEAP PLOT INPUT COMMAND MANUAL

```
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 INPUT 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.

BORDer FEAP PLOT INPUT COMMAND MANUAL

```
bord,n1 number
bord,on number
bord,off number
```

The `BOR`Der 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 INPUT 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 FEAP BPLOt INPUT COMMAND MANUAL

```
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 INPUT 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 INPUT 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).

CENTer FEAP PLOT INPUT COMMAND MANUAL

```
    cent,x,y
```

The `CENT`er 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 $0 < \mathrm{x} < 1.4$ and $0 < \mathrm{y} < 1.0$.

CLEAr FEAP PLOT INPUT COMMAND MANUAL

```
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 FEAP PLOT INPUT COMMAND MANUAL

```
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.

COLOr FEAP PLOT INPUT COMMAND MANUAL

```
    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.

CONTour FEAP PLOT INPUT COMMAND MANUAL

```
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.

```
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 INPUT COMMAND MANUAL

```
defo,v1,n2
```

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.

An undeformed option is specified using the `UNDE`formed command.

```
disp,,n2
```

Plot nodal generalized displacements as vectors at each node. Vector lengths will be scaled in proportion to the maximum displacement, accordingly some vectors may be too small to be visible. If `n2` is nonzero the vector tip will appear next to the node; whereas, if `n2` is zero the tail of the load vectors are on the nodes. Default: n2 = 0.

```
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 `DPLO`t 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 `CONT`our 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 `DPLO`t command may be combined with `SPLO`t to show all quantities along selected lines. Currently, this command works for 2-D problems displayed in a `CART`esian mode only.

EIGElement FEAP PLOT INPUT COMMAND MANUAL

```
    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 (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
1	11-stress
2	22-stress
3	33-stress
4	12-stress
5	23-stress
6	31-stress
7	1-heat flux
8	2-heat flux
9	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.

EXNOde FEAP PLOT INPUT COMMAND MANUAL

```
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 FEAP PLOT INPUT COMMAND MANUAL

```
    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.

FACTor FEAP PLOT INPUT COMMAND MANUAL

```
    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.

FRAMe FEAP PLOT INPUT COMMAND MANUAL

```
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 `FRAMe` may be cleared independently for some devices using a `WIPE,n1` command.

FULL FEAP PLOT INPUT COMMAND MANUAL

```
    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 FEAP PLOT INPUT COMMAND MANUAL

```
    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.

IMAGinary FEAP PLOT INPUT COMMAND MANUAL

```
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.

LABEl FEAP PLOT INPUT COMMAND MANUAL

```
labe
```

The LABEl 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 NOLAbel command. The default mode is enabled. This command may be used to produce contour PostScript output files without the labels.

LINE FEAP PLOT INPUT COMMAND MANUAL

```
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).

LOAD FEAP PLOT INPUT COMMAND MANUAL

```
    load
```

The `LOAD` command may be used to display the applied nodal loads on the system. Prior to any solution steps all loads are shown; however, after any solution step only the current non-zero loading are given.

LOGO FEAP PLOT INPUT 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 INPUT 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 INPUT 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`.

MATErial FEAP PLOT INPUT COMMAND MANUAL

```
    mate,n1
```

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 number, and a value of zero indicates all materials are to be displayed. (Default: n1 = 0).

MESH FEAP PLOT INPUT COMMAND MANUAL

```
    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).

NODE FEAP PLOT INPUT 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.

NOFUll FEAP PLOT INPUT COMMAND MANUAL

```
    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 INPUT COMMAND MANUAL

```
nola
```

The `NOLA`bel 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 `LABE`l command. The default mode is enabled. This command may be used to produce contour PostScript output files without the labels.

NOPRint FEAP PLOT INPUT COMMAND MANUAL

```
    nopr
```

The `NOPR`int command will suppress the print mode for interactive plot prompts. A `PRIN`t command will enable prints. The default is `PRIN`t.

NORAnge FEAP PLOT INPUT COMMAND MANUAL

```
    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*.

OUTLine FEAP PLOT INPUT COMMAND MANUAL

```
    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 FEAP PLOT INPUT COMMAND MANUAL

```
    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.

PBOUndary FEAP PLOT INPUT COMMAND MANUAL

```
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 INPUT 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).

PELEment FEAP PLOT INPUT COMMAND MANUAL

```
    pele
```

The `PELM`ent 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*.

PERSpective FEAP PLOT INPUT COMMAND MANUAL

```
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.

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.

In batch mode, the quantities `inew`, etc. must appear after the command language `END` command and ordered so that reads are performed at the execution of the correct instruction. In interactive mode prompts will be given for each input item.

```
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 FEAP PLOT INPUT COMMAND MANUAL

```
    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.

PNODe FEAP PLOT INPUT COMMAND MANUAL

```
    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 INPUT COMMAND MANUAL

```
    post,n1,n2
```

The POSTScript command will enable the output of a PostScript file for later use in producing hard copy plots. The sequence is initiated by the first POSTScript 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 POSTScript 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.

PRAXes FEAP PLOT INPUT COMMAND MANUAL

```
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 INPUT 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 INPUT 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 INPUT 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 `PROM`pt 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.

```
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 $= 0$.

QUADrant FEAP PLOT INPUT COMMAND MANUAL

```
quad
```

The QUADrant command may be used in combination with the SYMMetry 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 INPUT 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 INPUT COMMAND MANUAL

```
reac,,n2
```

Plot nodal reactions for current solution state. 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 `n2` is non-zero the vector tip will appear next to the node; whereas, if `n2` is zero the tail of the load vectors are on the nodes.

REAL FEAP PLOT INPUT COMMAND MANUAL

```
    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 INPUT COMMAND MANUAL

```
    refr
```

The `REFR`esh command will redisplay all plot segments which are in the current X-window buffer. No action is taken for other display types.

REGIon FEAP PLOT INPUT COMMAND MANUAL

```
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).

SCALe FEAP PLOT INPUT 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 INPUT COMMAND MANUAL

```
    scre,<on,off>
```

The SCREen 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 INPUT COMMAND MANUAL

```
    show
```

The SHOW command will output the state of for several plot parameters to the text window.

SIZE FEAP PLOT INPUT 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.

SNODe FEAP PLOT INPUT COMMAND MANUAL

```
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 INPUT 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.

STREss FEAP PLOT INPUT COMMAND MANUAL

```
stre,n1,n2,n3
```

Plot contours of 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	11-stress
2	22-stress
3	33-stress
4	12-stress
5	23-stress
6	31-stress
7	1-heat flux
8	2-heat flux
9	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 SYMMetry 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).

TEXT FEAP PLOT INPUT COMMAND MANUAL

```
    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 FEAP PLOT INPUT COMMAND MANUAL

```
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 INPUT 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.

UNDEformed FEAP PLOT INPUT COMMAND MANUAL

```
    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 FEAP PLOT INPUT COMMAND MANUAL

```
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.

WINDow FEAP PLOT INPUT 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 FEAP PLOT INPUT COMMAND MANUAL

```
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 FEAP PLOT INPUT COMMAND MANUAL

```
    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 7.4. A summary of the changes is given below:

- The memory for all major arrays is now dynamically allocated using calls to subprograms `malloc` and `free`. The use of a blank common is still used to provide base addresses for the pointers to real and integer data. Thus, the current version does not require specification of a total amount of data memory – with problem size controlled only by the memory available in each computer.
- The *FEAP* log file has been revised to provide additional information on solution effort and to list problems encountered during execution. This file has the name `Lxxxx` where `xxxx` is the name extracted from the name of the input file by deleting the first character.
- The user material model interface has been revised to permit easier addition of new models (see the Programmer manual for details).
- The mesh manipulation user module interface also has been revised.
- The user plot module interface also has been revised to allow for user provided modules (see Programmer manual).
- A BDF2 time integration method has been added for first order PDE.
- Serendipity shape functions have been added to displacement model brick elements.
- A new solution command, `MONOlithic`, has been added to permit full coupled solution during partitioned analyses.
- Graphics outputs have been improved to permit contour plots for 6-node triangular facets

- A node to rigid surface option has been added to contact. The rigid surface is allowed to change uniformly for cylinders, planes, and spheres.
- The user contact element interface has an option to permit adding a user selected name for the interface. This is now consistent with user added mesh, manipulation, solution, and plot modules.
- The material property input module name has been changed from `inpt2d.f` to `inmate.f` and new options added.
- A Mooney-Rivlin finite elastic model has been added in terms of invariants. Previously, it was necessary to use the Ogden model to obtain solutions.
- New 3-D frame element options are provided for *non-transient* solution. The options are: Crisfield-Jelenic co-rotational form and Simo-Vu Quoc form. Both permit 3-node elements in addition to 2-node ones.
- Body forces now are permitted in all element forms.
- A torsion motion is added to axisymmetric elements of displacement and mixed type. Not available in the enhanced strain element.
- Three dimensional elements of type displacement may be 4-node linear tetrahra, 10-node quadratic tetrahedra, 8-node tri-linear brick, 20-node Serendipity quadratic, 27 node Lagrangian tri-quadratic brick or 64-node Lagrangian tri-cubic brick form.
- Three dimensional elements of type thermal may be 4-node linear tetrahra, 10-node quadratic tetrahedra, 8-node tri-linear brick, 20-node Serendipity quadratic, 27 node Lagrangian tri-quadratic brick or 64-node Lagrangian tri-cubic brick form.
- Two dimensional elements of type displacement may be 3-node linear triangular, 6-node quadratic triangular, or 4-node bilinear to 16-node bi-cubic quadrilaterals.
- Two dimensional elements of type mixed may be 4-node bi-linear to 16-node bi-cubic quadrilaterals.
- One dimensional elements for solid, thermal, and convection form added.
- An unsymmetric option to form damping and mass matrices has been added.

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 properaly. As always, please inform us of these or desired new features by sending comments by e-mail to:

```
feap-help@vulture.ce.berkeley.edu
```

Index