FEAPpv - - A Finite Element Analysis Program

Personal Version 1.0 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

May 2001

Contents

1 Introduction **1**

2 Problem Definition **3**
2.1 Execution of FEAPpv and Input/Output Files. 4
2.2 Modification of Default Options 4

3 Manual Organization **6**

4 Input Records **7**
4.1 Constants 8
4.2 Parameters 8
4.3 Expressions 9
4.4 Functions 10
4.5 Include Commands in Mesh Input 11
4.6 Read and Save Commands in Mesh Input 12

5 Mesh Input Data **14**
5.1 Start of Problem and Control Information 14
5.2 Global Data 16
5.3 Nodal Coordinate and Element Connections 16
5.3.1 The COORdinate Command 17
5.3.2 The ELEMent Command 18
5.3.3 The BLOCk Command 19
5.3.4 The BLENd Command 22
5.4 Coordinate and Transformation Systems 25
5.4.1 Coordinate Transformation 27
5.5 Looping to Replicate Mesh Parts 28
5.6 Regions and Element Groups 31
5.7 Nodal Boundary Condition Inputs 31
5.7.1 Basic input form. 32
5.7.2 Edge input form. 33
5.7.3 Coordinate input form. 34
5.7.4 Hierarchy of input forms. 35

5.7.5 Time dependent load functions . . . 35
5.8 Surface Loading . . . 37

6 Element Library **40**

7 Material Models **46**
7.1 Orthotropic Linear Elastic Models . . . 46
7.2 Isotropic Linear Elastic Models . . . 48
7.3 Isotropic Finite Deformation Elastic Models . . . 49
7.3.1 St. Venant-Kirchhoff . . . 50
7.3.2 Neo-Hookean and Modified Neo-Hookean Models . . . 51
7.4 Rayleigh Damping . . . 52
7.5 Viscoelastic Models . . . 53
7.6 Plasticity Models . . . 54
7.7 Heat Conduction Material Models . . . 55
7.8 Mass Matrix Type Specification . . . 55
7.9 Element Cross Section and Load Specification . . . 56
7.9.1 Resultant formulations . . . 56
7.10 Miscellaneous Material Set Parameter Specifications . . . 57

8 End and Miscellaneous Commands **59**

9 Mesh Manipulation Commands **60**
9.1 The TIE Command . . . 60
9.2 The LINK Command . . . 61

10 Command Language Programs **62**
10.1 Problem Solving . . . 64
10.1.1 Solution of Non-linear Problems . . . 65
10.1.2 Solution of linear equations . . . 67
10.2 Transient Solutions . . . 68
10.2.1 Quasi-static solutions . . . 68
10.2.2 First order transient solutions . . . 70
10.2.3 Second order transient solutions . . . 71
10.3 Transient Solution of Linear Problems . . . 72
10.4 Time Dependent Loading . . . 73
10.5 Continuation Methods: Arclength Solution . . . 75
10.6 Augmented Solutions . . . 76
10.7 Time History Plots . . . 76
10.8 Viewing Solution Data: SHOW Command . . . 77
10.9 Reexecuting Commands: HISTORY Command . . . 78
10.10Solutions Using Procedures . . . 78

10.11Output of Element Arrays 80

11 Plot Outputs **81**
11.1 Screen Plots 81
11.2 PostScript Plots 84

A Mesh Manual **89**

B Mesh Manipulation Manual **162**

C Solution Command Manual **168**

D Plot Manual **229**

Chapter 1

INTRODUCTION

During the last several years, the finite element method has evolved from a linear structural analysis procedure to a general technique for solving non-linear partial differential equations. An extensive literature exists on the method describing the theory necessary to formulate solutions to general classes of problems. It is assumed that the reader is familiar with the finite element method as describe in popular reference books (e.g., *The Finite Element Method*, 5th edition, by O.C. Zienkiewicz and R.L. Taylor[1, 2, 3] and desires either to solve a specific problem or to generate new solution capabilities.

The Finite Element Analysis Program - Personal Version (*FEAPpv*) is a computer analysis system designed for:

1. Use in an instructional program 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 has been developed primarily for UNIX work station and personal computer (Windows PC) environments and includes an integrated set of modules to perform input of data describing a finite element model, construction of solution algorithms to address a wide range of applications, and graphical and numerical output of solution results.

A problem solution is constructed using a command language concept in which the solution algorithm is written by the user. Accordingly, with this capability, each user may define 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 features for model description and command language statements to meet specific applications requirements. These additions may be used to assist users in generating meshes for specific classes of problems or to import meshes generated by other systems.

The *FEAPpv* system includes a limited element library. Some elements are available to model one, two and three dimensional problems in linear and non-linear structural and solid mechanics and for linear heat conduction problems. Each available element uses a material model library. A few material models are provided for elasticity, viscoelasticity, plasticity, and heat transfer constitutive equations, however, it is assumed that users will want to add additional models for their specific application. Elements provide capability to generate mass and geometric stiffness matrices for structural problems and to compute output quantities associated with 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 *FEAPpv* system. Details on specific requirements to add an element as well as other optional features available are included in volume 1 and 2 of the books and in the *FEAP Programmers Manual*.[4]

The next several sections of this manual describe how to use existing capabilities in the *FEAPpv* system. The discussion centers on three different phases of problem solution using the system:

1. Mesh description options;
2. Problem solution options; and
3. Graphical display options.

The *FEAP Example Manual*[5] may be consulted for examples of some of the input and solution options available, however, users should consult this manual to ensure that all features needed are part of the *FEAPpv* system. Generally, problems defined for *FEAPpv* will also work with the full version of the program (*FEAP*), but those for *FEAP* may not work with *FEAPpv*.

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 values of its parameters in 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. In order to complete a problem specification it is necessary also to specify additional data, e.g., boundary conditions, loads, etc..

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 *FEAPpv*. The steps to define a mesh for *FEAPpv* are contained in Chapters 5 to 9 and the input data for several example problems is described in the *FEAP Example Manual*.[5] Each of the commands available for constructing mesh data for *FEAPpv* is described in Appendix A.

The second phase of a finite element analysis is to specify the solution algorithm for the problem. This may range from a simple linear static (steady state) analysis for one loading condition to a more complicated transient non-linear analysis subjected to a variety of loading conditions. *FEAPpv* permits the user to specify the solution algorithm utilizing a solution command language which is described in Chapter 10 and also illustrated in the book.[1, 2] Each solution command is also described in Appendix B of this report.

2.1 Execution of FEAPpv and Input/Output Files.

The execution of *FEAPpv* is initiated by issuing the command:[1]

```
FEAPpv
```

In PC use it is possilble to execute the program using standard windows options or to open an MS-DOS window and execute with the above command. If this is a first execution of the program it is necessary to provide names for the file containing the input data and those to receive output information. Upon a successful first execution of the program a file `feapname` will be written to disk to preserve the name for each of the input and output disk files. If it is desired to reinstall the program the `feapname` file should be deleted and the `FEAPpv` command then reissued.

For each subsequent execution of the program using the `FEAPpv` command, the analyst 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). The name of a default selection will also be indicated and may be accepted by pressing the return (enter) key without specifying any other data. Prior to running *FEAPpv* it is necessary to create the input data file using a standard text editor or word processing system. The other files are created by *FEAPpv*. A large part of the remainder of this manual is directed to 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 a problem.

2.2 Modification of Default Options

At the time that the executable version of *FEAPpv* is created default values for several parameters may be set in file `feappv.f`. These default parameters may be changed without recompilation 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.

[1] Users may give a different name for the executable during the compilation of the full system – it is then necessary to specify this name instead of the one stated above.

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 reversed order.
		normal	Makes color 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
increment	value		Set increment value change to force reduction in array size.

Table 2.1: Options for Changing Default Parameters

Chapter 3

MANUAL ORGANIZATION

The user manual for *FEAPpv* is separated into several distinct parts. Each part describes the specific function and the input data required for the 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. Terminating mesh description (Chapter 8).
5. Manipulating mesh parts to tie and link degrees of freedom (Chapter 9).
6. Description of the solution command language (Chapter 10). This section of the manual includes basic solution algorithms to solve problems.
7. Plot features contained within the program (Chapter 11).

More information about each of the user manuals is contained in the following sections. The various options and parameters for each command to describe mesh input, problem solution, and plotting are included in the appendices to this manual. A *FEAP Programmer Manual*[4] describing the procedures to add features and elements is also available for users who wish to modify or extend the capabilities of *FEAPpv*.

Chapter 4

INPUT RECORDS SPECIFICATION

Data input specifications in *FEAPpv* 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. 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 below. Sets of records may also be designated as a *save* set and later *read* again for reuse.

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 *FEAPpv* 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 would yield data values of `4.0` and `-6.456`, respectively.

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 and 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.

4.2 Parameters

The use of parameters will simplify the data input required to define problems for a *FEAPpv* 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

```
a  = 3.567
e1 = 200.0e9
nu = 0.3
```

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 would use the value of the parameter a defined above

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

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.

4.3 Expressions

The most powerful form of data input in *FEAPpv* 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 this hierarchy proceed from left to right in each expression. At the present time only one level of parenthesis may appear in any expression. Accordingly, 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. Expressions may appear in any location in place of a constant or an expression. Accordingly, a force may be assigned as

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

Additionally, node and element numbers may also appear as expressions. This permits the input of *substructure* parts in a modular form. For example,

```
BLOCk
  CARTesian,4,4,n,e,m
  1,0.+x,0.+y
  2,5.+x,0.+y
  3,5.+x,5.+y
  4,0.+x,5.+y
                ! end with blank record
COOR
  n+25,0,5.5+x,2.5+y
                ! end with blank record
ELEM
  e+16,p,n+14,n+15
                ! end with blank record
```

could be used to input a block of nodes and elements (see Section 5.3.3). By specifying the values of the parameters `n,e,m,x,` and `y` a form of a *substructure* is defined. The part may be replicated using either the `INCLude` option or the name associated with `SAVE` and `READ` in the mesh data input statements.

4.4 Functions

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

```
abs   exp,  int,  log,   sqrt,
```

```
sin,  cos,  tan,  atan,  asin,  acos,
sind, cosd, tand, atand, asind, acosd,
cosh, sinh, tanh,
```

The trigonometric and inverse trigonometric functions which end in *d* involve values of angles in degrees; whereas, the ones without involve values in radians.

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

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

will compute the value of π and assign it to the parameter p. Internal computations are all preformed in double precision arithmetic (e.g., as `REAL*8` variables). Again note that the function parenthesis count as one level, hence

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

is not a legal expression at the present time. It should be replaced by the pair of statements

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

4.5 Include Commands in Mesh Input

Any data input records may be placed in a separate file and read using the `INCLude` command. The form for the include is a single record

```
INCLude,filename
```

where `filename` is the name of the file containing the input data. 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 *FEAPpv* data using:

[1] Upper and lower case letters are treated as different on workstations but the same on PCs

```
COORdinates
INCLude,Blockxy
                    ! blank termination record
```

The information in each file must always be in the format required by *FEAPpv*. 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*.

4.6 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
PARAmete
  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.

Chapter 5

MESH INPUT DATA SPECIFICATIONS

The description of the mesh data for a problem to be solved by *FEAPpv* 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. A start/title record which must have as the first four non-blank characters *FEAPpv* (either upper or lower case letters may be used with the remainder used as a problem title.

2. The second recond contains problem size information consisting of:

   (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

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

As noted above, input records for *FEAPpv* are in free format. Each data item is separated by a comma, equal sign or 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 or comma).

For standard input options *FEAPpv* can automatically determine the number of nodes, elements, and material sets. Thus, on the control record the values of NUMNP, NUMEL, and NUMMAT may be omitted (i.e., specified as zero). When using automatic numbering it is generally advisable to use mesh input options which avoid direct specification of a node or element number. Specification of nodal loads (forces), nodal displacements (displacements), and boundary condition restraint codes have options which begin with `E` and `C` for *edge* and *coordinate* related options, respectively. It is recommended these be used whenever possible.

The use of the automatic determination of data requires a the mesh data to be read twice: Once to do the counts and once to input the data. For problems with a large number of data records, this may result in significant time lapse during the input data phase. The need for a second read may be avoided by inserting a `NOCOunt` record *before* the `FEAPpv` record and providing the actual number of nodes, elements and material sets on the control record.

We next consider commands used to describe the remainder of the finite element mesh. In *FEAPpv* each data set starts with a command name of which only the first four characters are used as identifiers. Appendix A describes options for each mesh input and manipulation command. Immediately following each command is the data to be processed. Where a variable number of records is needed to define the data set a blank line is used as a termination record. Extra blank lines before or after data sets are ignored.

Commands may be in any order. If there is any order dependence *FEAPpv* will transfer the input data to temporary files and process it after the mesh specification is terminated by the `END` command. Thus, information will not necessarily occur in the output file in the order which data is specified in the input file.

All data from a mesh input is written to the output file by default. For very large problems the size of the output file may become large. Once a mesh has been checked for correctness it may not be necessary to retain this information in 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. Insertion of a `NOPRint` record before any data set (not within a data set) suspends writing the data to the output file until another `PRINt` command is encountered.

5.2 Global Data

FEAPpv 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 masss input using the `MASS` command using the command

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

5.3 Nodal Coordinate and Element Connections

The basic mesh for *FEAPpv* 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). 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 5.4 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.

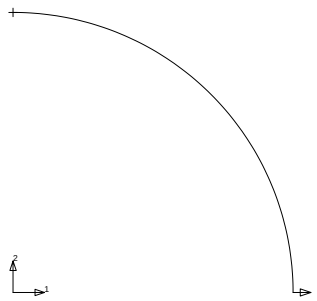

Figure 5.1: Curved Beam

5.3.1 The COORdinate Command

The coordinates of nodes may be specified using the COORdinate command. For example, the commands to generate polar coordinates for an eleven node mesh of a circular beam with radius 5 are given by:

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

These coordinates may then be converted from polar to cartesian form using the POLAr command. For the coordinate input shown above this is given as:

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

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

After the COORinate command individual records defining each nodal point and its coordinates are specified 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 corresponding 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.

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

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

The missing data is generated from M to N in increments of MG; that is the first generated node will be M+MG. Linear interpolation of the coordinates is used to define the 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 a non-zero MG will be adjusted to ensure that generation is in the correct direction.

Coordinate data is processed to determine the total number of nodes in a mesh. Nodal coordinates may also be defined using the BLOCk or the BLENd commands.

5.3.2 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=30,NEN)
```

That is, each record must contain no more than 16 items of data as mentioned in Chapter 4.

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

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

where 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 5.2

Element data is processed to determine the total number of elements in a mesh. Element data may also be defined using the BLOCk and BLENd commands.

5.3.3 The BLOCk Command

Regular patterns of nodes and element may be input using the BLOCk command. The block command can input patches of line elements (truss or frames), triangles or quadrilaterals, or three dimensional hexahedral (brick) or tetrahedral elements.

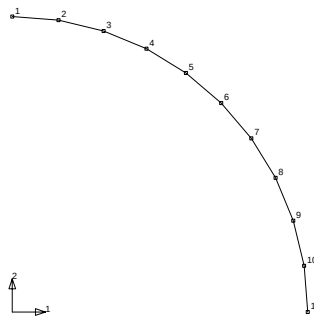

Figure 5.2: Mesh for Curved Beam. 10 Elements

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

```
BLOCk
  type,r-inc,,node1,elmt1,mat,r-skip
  1,X_1,Y_1,Z_1
     ...
  N,X_N,Y_N,Z_N
           ! Termination record
```

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

```
BLOCk
  type,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
```

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

```
BLOCk
  type,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
```

where the parameters are defined as:

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

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 hexahera (bricks).
=11: 4-node tetrahedra.

An example mesh input using the `BLOCk` command is the line elements shown in Figure 5.2. 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 *BLEN*d options with the ELEM option wher 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 values for x_0, y_0, and z_0. 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.

5.3.4 The BLENd Command

A block of nodes and elements also may be generated using a blending function approach (e.g., see [6], pp 181 ff. or similar information in [1]). In *FEAPpv* the blending function meshes are created from a set of control points - call super-nodes - (SNODe command), edges (SIDE command) and the BLENd command. Meshes may be created as SURFaces in two and three dimensions or as SOLIds in three dimensions. The two dimensional blended mesh shown in Figure 5.3 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 5.4.

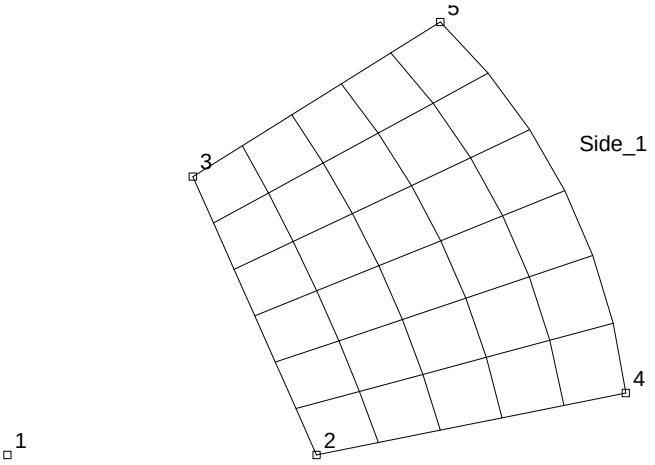

Figure 5.3: Two-dimensional Blended Mesh

```
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 5.4: Two-dimensional blended mesh data

The coordinates for super-nodes *always* are given in Cartesian form. Also, only the edges for non-straight or non-uniformly spaced increments need be given. *FEAPpv* will automatically add all straight uniformly spaced edges not given as input data. The specification of edges using the `SIDE` command is given by the general form:

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

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

1. `Type = CARTesian`: For Lagrange interpolation in cartesian coordinates. The `Vi` values are the 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.

2. `Type = POLAr`: For Lagrange interpolation in polar coordinates. The interpolations are given as:

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

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

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

3. `Type = SEGMent`: For 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.

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). 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. 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 number of generation increments and other parameters are given in Table 5.1 for surface generations and in Table 5.2 for solid generations.

Type	- Blend type (`SURF`ace.
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 elements in patch (default = 1)

Table 5.1: Surface Blend Parameters

A blended region for a three dimensional mesh is shown in Figure 5.5 and generated using the data shown in Figure 5.6.

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 command.

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)

Table 5.2: Three-dimensional Solid Blend Parameters

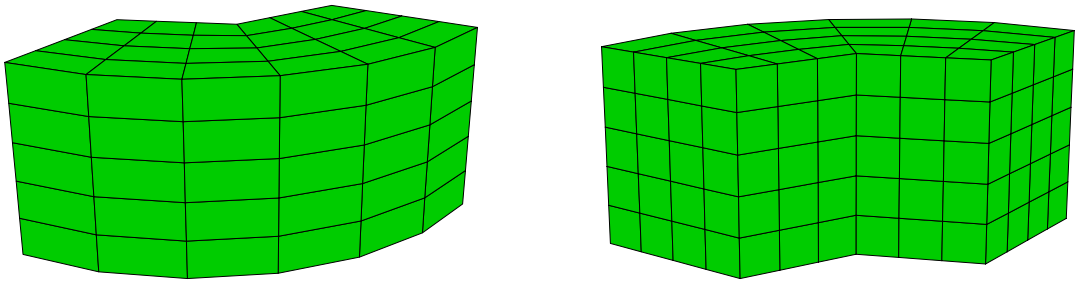

Figure 5.5: Three-dimensional Blended Mesh

5.4 Coordinate and Transformation Systems

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

```
POLAr
  NODE,n1,n2,inc
```

```
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 5.6: Three-dimensional blended mesh data

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

```
POLAr
  ALL
```

The transformation is given by

$$x_1 = x_0 + r \cos\theta$$

$$x_2 = y_0 + r \sin\theta$$

and

$$x_3 = z_0 + z$$

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

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

By default x_0, y_0, z_0 are zero.

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

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

Transformations use the relations

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

$$x_2 = y_0 + r \sin\theta \sin\phi$$

and

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

5.4.1 Coordinate Transformation

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

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

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

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

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

The TRANsform command may be used as many times as needed. In particular, it may be used with a portion of a mesh (substructure) in an include file to replicate repeated parts of meshes. When a reflection is performed, *FEAPpv* 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 - $\hat{x}_i$).

5.5 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. *FEAPpv* 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 *FEAPpv* 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.

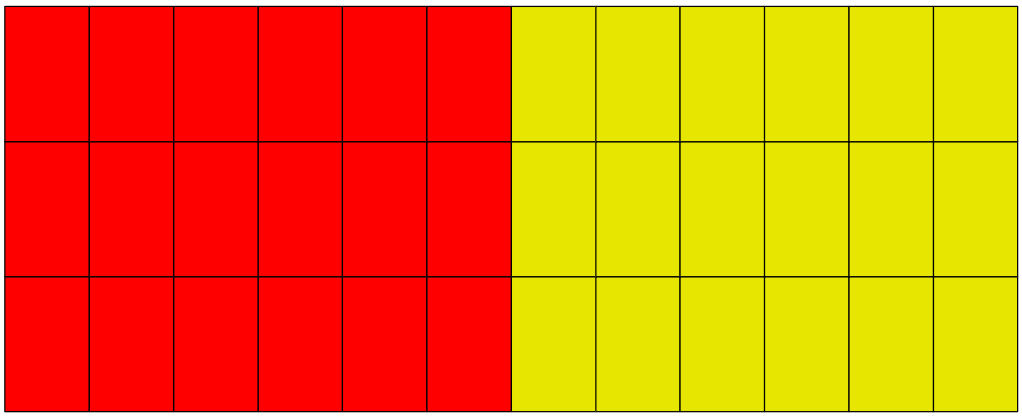

Figure 5.7: Two blocks using LOOP-NEXT commands

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 5.3 where we note the use of the loop using the `TRANsform` command. The above example produces the mesh shown in Fig. 5.7 and is trivial (also not much is gained over a construction using two block commands directly).

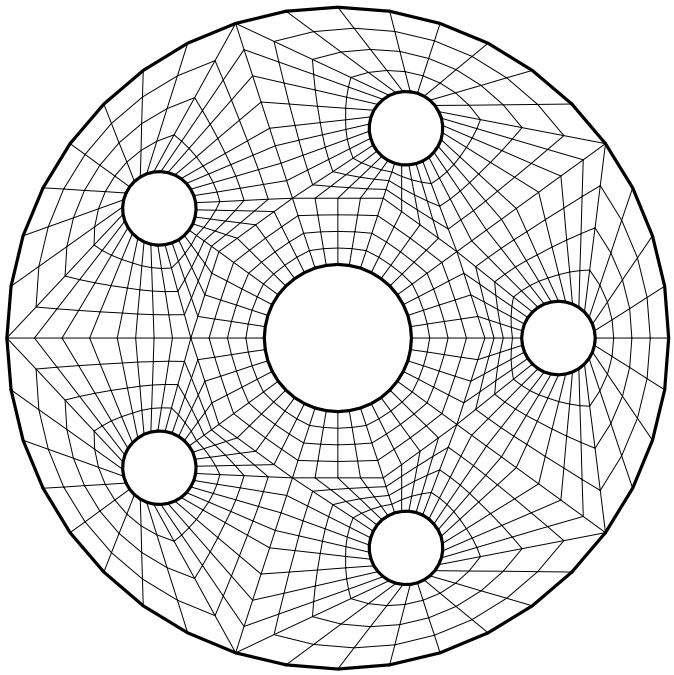

Figure 5.8: Disk with holes

A more involved example is shown in Fig. 5.8 for a disk containing circular holes. This example was constructed using the commands shown in Table 5.4. The file `Iwseg` contains the mesh for one part of the repeating mesh as shown in Fig. 5.9.

Many more involved meshe constructs may be considered using the `LOOP-NEXT` commands.

```
FEAPpv * * 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 5.3: `LOOP-NEXT` mesh construction

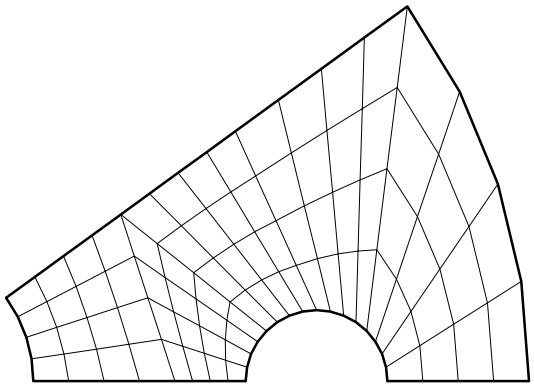

Figure 5.9: Mesh segment for disk with holes

```
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 5.4: `LOOP-NEXT` disk mesh construction

5.6 Regions and Element Groups

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

```
REGIon,number
```

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

The use of regions facilitates merging groups of elements whose nodes should be common but have different numbers (e.g., those defined using `BLOCk` commands). An illustration of this option is used in Example 4 above.

5.7 Nodal Boundary Condition Inputs

The basic *FEAPpv* 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 interpretted 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 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: Node numbers; Nodal coordinate values; or Edge coordinate values.

5.7.1 Basic input form.

The basic options to input the nodal quantities associated with boundary conditions is 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 opther 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 5.5: Nodal Boundary Condition Quantity Inputs

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

```
FORCe
  19  0    0.0     10.0
                      ! Termination record
```

The input records for basic `FORCe`, `DISPlacement`, `BOUNdary` condition and `ANGLe` commands are similar to those for `COORinates` with the node and generation increment in the first two fields and the list of values for each degree-of-freedom in the remaining field. 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
```

The value of a force or displacement will be selected based on the *boundary restraint code* value. 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 only the non-zero displacement will be used. During execution it is possible to change the boundary restraint codes to then use the non-zero force.

To use the basic input option it is a users responsibility to determine the correct number for each node - often the graphics capability of *FEAPpv* 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 the nodal values.

5.7.2 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 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, above we are indicating the first degree-of-freedom is to have specified displacements and the second is to have specified forces. *FEAPpv* locates all nodes which are within a small tolerance of the specified coordinate *after the mesh input is completed*.

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 conditions edge options are specified as:

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

By the 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 would have only its second degree-of-freedom restrained.

5.7.3 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 in the y-direction 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 5.8).

Once again, 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.

5.7.4 Hierarchy of input forms.

The input of the nodal boundary data is performed by *FEAPpv* in a specific order. Data input in the basic form is interpretted 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 interpretted 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 interpretted 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 interpretted 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 *FEAPpv* 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.

5.7.5 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 loadings active at a particular time instant.

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.

5.8 Surface Loading

FEAPpv uses the `CSURface` 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.

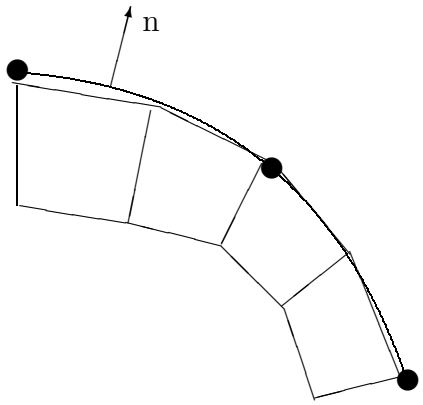

Figure 5.10: Two-Dimensional Surface Loading

FEAPpv 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 5.10) 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.

Use of the `POLAr` 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.

Chapter 6

ELEMENT LIBRARY

FEAPpv contains a library of standard elements and material models which may be employed to solve a wide range of problems in solid and structural mechanics, and heat transfer analysis. In addition, users may program and add new elements to the program. The type of element to be employed in an analysis is specified as part of the `MATErial` data sets. The first record of each material data set also contains the material property number. Each material property number is an integer ranging from one (1) to the maximum number of material sets specified on the control data record (which immediately follows the `FEAPpv` start/title record); however, as noted earlier, the maximum number on material sets on the control data may be specified as zero and *FEAPpv* will automatically compute the maximum number of material sets from the input data. The second record of the material set data defines the type of element to be used. The library of standard elements includes the following types:

1. SOLId - A solid element is used to solve continuum problems with either small or large deformations. Options exist to use finite elements based on displacement or mixed formulations. Small deformation elements contains a library of models for elastic, viscoelastic, or elastoplastic constitutive equations. Finite deformation elements contain only elastic material models. For two dimensional problems each element is a quadrilateral with between 4 and 9-nodes. The two dimensional displacement formulation also permits use of 3 or 6-node triangular elements. The degrees of freedom on each node are displacements, u_i, in the coordinate directions. The degrees of freedom are ordered as: 2-D Plane problems, u_x, u_y, coordinates are x, y; 2-D Axisymmetric problems, u_r, u_z, coordinates are r, z. For three dimensional problems only a displacement form is included and each element is an 8-node hexahedron (brick) or a 4-node tetrahedron with degree-of-freedoms u_x, u_y, u_z. No finite deformation element is included for three dimensional solids.

2. FRAMe - The frame element is used to model structural members which include axial, bending, and 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 $u_x, u_y, u_z, \theta_x, \theta_y, \theta_z$;

3. TRUSs - The truss element is used to model structural members which include axial deformations and axial 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 may be used in one, two and three dimensional problems. The degrees of freedom on 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

4. PLATe - The plate element is used to model structural behavior of planar (flat) 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 on 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;

5. 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 is for small deformations only and 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. In two-dimensions the element is a 2-node ring sector whereas in three-dimensions each element is a quadrilateral with 4-nodes. 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_r, u_z, θ in two dimensions and $u_x, u_y, u_z, \theta_x, \theta_y, \theta_z$ (6-dof) in three dimensions.

6. MEMBrane- The membrane element is used to model structural behavior of curved bodies which are thin and carry in-plane loading only. The element is generally 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 only in a three dimensional problem. The degrees of freedom on each node are: Displacements, u_i. The degrees of freedom are ordered as: u_x, u_y, u_z;

7. THERmal - The thermal element is used to compute temperatures in solid bodies or truss elements. The element solves the Fourier heat conduction equation. For two dimensional problems each element is a quadrilateral with between 4 and 9-nodes or a triangle with 3 or 6-nodes. For three dimensional problems each element is a brick with 8-nodes or a tetrahedron with 4-nodes. The degree of freedom on each node is temperature, T, and, by default, is placed in the first position in the unknowns (i.e., first degree of freedom). If the element is combined with a solid element to perform thermo-mechanical analyses it is necessary to relocate the temperature degree of freedom using the option on the material set element type record (see the `MATE`erial set command description in the *FEAPpv* Mesh User Manual in Appendix A).

8. USER - Each user element must be developed and added to the program. Provisions are included which permit the addition of up to 5 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 Reference [1] or the *FEAP* Programmers Manual[4] for information on adding a user element.

Each element requires additional input data to describe the specific constitutive model, the finite element formulation to be used, loading applied to elements, etc. As an example consider an analysis of a two dimensional continuum (solid) with a single material and constrained to a plane strain deformation state. The problem is to be modeled by an elastic material with isotropic properties.

```
MATErial,1
  SOLId
  ELAStic ISOTropic 30e+06 0.3
                            ! blank termination record
```

The property `ELAStic` is required for all types of `SOLId` elements. For two dimensional problems in small deformations there are two element types: (1) A displacement model and (2) A mixed $\mathbf{u} - p - \theta$ model. In finite deformations only the two dimensional displacement form exists. The material records for an analysis which includes linear elastic solid elements is shown above.

In two dimensional applications the displacement and the mixed formulation may be described by elements with between four (4) and nine (9) nodes. 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. In three dimensional applications the element is described by an eight (8) node hexahedron. The displacement model may also describe wedge (by coallescing appropriate nodes of the hexahedron) and a 4-node tetrahedron.

The material models and other options available for use with the solid elements are described in the next chapter.

In two-dimesions, the *frame elements* can treat small and large displacement problems. The small displacement element is restricted to elastic behavior and includes effects of bending and axial deformations. Cubic interpolations are used. The finite deformation frame element are based on the exact kinematic formulation of Ibrahimbegovic[7]. The element includes an elastic resultant model. Interpolations are linear along the beam axis. All elements have two nodes. To definie the orientation of the cross section for a three dimensional analysis it is necessary to define a `REFE`rence `VECT`or, `DIRE`ction, or `NODE`. A frame element is included using the commands:

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

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. Typically, `ELAStic` models are required and the cross section data given by `CROSs` section properties. The geometric data for orienting cross section axes is given by `REFErence VECTor` or `REFErence NODE` options.

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. Optional data is the type of inelastic constitutive model.

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). 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.

The *shell element* is capable of only small deformation analysis. The model includes bending and membrane strains only - no transverse shearing deformation is included - thus restricting application to thin shell problems only. A resultant elastic formulation is provided. A shell element is included using the commands:

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

Required data is the elastic material model (e.g., `ELAStic`) and the thickness given by the `THICk` option. Additional data for loading is also available.

A *membrane element* is derived from the shell element by deleting the bending deformations, thus leaving only the in-plane strain deformation terms. Elements for small displacements are included and are restricted 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.

The *thermal elements* are all based on a standard Galerkin (irreducible, displacement) formulation.[1] The element shapes available are identical to those for the displacement form of a solid element.

At present the finite deformation material models for the solid elements do not permit a coupled thermo-mechanical analysis. The small deformation models for elasticity do permit coupled thermo-mechanical analyses to be performed using an iterative solution algorithm.[1] The material behavior for the thermal analysis is a linear Fourier model. Both isotropic and orthotropic models are available. The thermal element is included using the commands:

[1] While the model is linear, there is no tangent terms for the coupling between temperature and displacement. Thus, solution requires added iterations

```
MATErial,1
  THERmal
    FOURier ...
```

Required data is the material model given by the `FOURier` option. Note also that it is necessary to specify the glogal degree of freedom for the temperature when displacements are present (see below).

The specification of *user elements* must contain a number of an element module which has been added to *FEAPpv*. Each user developed element module is designated as `subroutine elmtnn(...)`, where `nn` ranges from `01` to `05`. Accordingly, a typical set of data for a user element `elmt02` is given as:

```
MATErial,1
  USER      2              ! Use elmt02(...) module
  xxxxxx                   ! Additional data records
                           ! blank termination record
```

The first two records of the `MATE`rial set always must be:

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

where `ma` is the material set number, `type` is the element type (e.g., solid, truss, etc.), `unum` is the user element number, `mset` is the material set number given for each element (by default it is 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 *FEAPpv* it is one needs only the `type` parameter unless degrees of freedom are to be relocated (e.g., for thermal analysis).

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 Orthotropic Linear Elastic Models

The orthotropic linear elastic material model in *FEAPpv* is given by

$$\hat{\boldsymbol{\epsilon}} \;=\; \hat{\boldsymbol{C}}\,\hat{\boldsymbol{\sigma}} \;+\; \hat{\boldsymbol{\epsilon}}^{th} \tag{7.1}$$

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

$$\hat{\boldsymbol{C}} \;=\; \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{12}}{E_1} & -\frac{\nu_{13}}{E_1} & 0 & 0 & 0 \\ -\frac{\nu_{21}}{E_2} & \frac{1}{E_2} & -\frac{\nu_{23}}{E_2} & 0 & 0 & 0 \\ -\frac{\nu_{31}}{E_3} & -\frac{\nu_{32}}{E_3} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{12}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{23}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{31}} \end{bmatrix} \tag{7.2}$$

with E_i elastic moduli in principal directions, ν_{ij} Poisson ratios for strains measured in 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[8] (except for shear modulus terms).

The thermal strain is given by:

$$\hat{\boldsymbol{\epsilon}}^{th} \;=\; \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta T \;=\; \hat{\boldsymbol{\alpha}}\, \Delta T \tag{7.3}$$

where

$$\Delta T \;=\; T \,-\, T_0\,, \tag{7.4}$$

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 7.1 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 7.8. Using this angle *FEAPpv* transforms the input material compliances to

$$\boldsymbol{C} = \boldsymbol{R}^T\, \hat{\boldsymbol{C}}\, \boldsymbol{R} \tag{7.5}$$

and converts the constitutive equation to the form

$$\boldsymbol{\sigma} \;=\; \boldsymbol{D}\, \boldsymbol{\epsilon} \;+\; \boldsymbol{\beta}^{th} \tag{7.6}$$

where

$$\boldsymbol{C} = \boldsymbol{C}^{-1} \tag{7.7}$$

and

$$\boldsymbol{\beta}^{th} = -\, \boldsymbol{D}\, \boldsymbol{\epsilon}^{th} \tag{7.8}$$

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
                           ! blank termination record
```

Additional data options and parameters are defined in Table 7.1.

Command	Type	Parameters
ELAStic	ORTHotropic	$E_1, E_2, E_3, \nu_{12}, \nu_{23}, \nu_{13}, G_{12}, G_{23}, G_{31}$
ELAStic	ISOTropic	E, ν
ELAStic	TRANsverse	$E_1, E_2, \nu_{12}, \nu_{13}, G_{31}$
DAMPing	RAYLeigh	a0 , a1
PLAStic	MISEs	Y_0, Y_∞, β
PLAStic	HARDening	H_{iso}, H_{kin}
VISCoelastic		μ_i, τ_i
THERmal	ORTHotropic	$\alpha_1, \alpha_2, \alpha_3, T_0$
THERmal	ISOTropic	α, T_0
FOURier	ORTHotropic	K_1, K_2, K_3, c
FOURier	ISOTropic	K, c
DENSity		ρ
ANGLe		ψ

Table 7.1: Material Model Data Inputs

7.2 Isotropic Linear Elastic Models

The isotropic models require less data since now only two independent elastic parameters are needed to define $\hat{\boldsymbol{C}}$. These are taken as Young's modulus, E, and Poisson's ratio, ν and for an isotropic material the elastic compliance array is

$$\hat{\boldsymbol{C}} = \begin{bmatrix} \frac{1}{E} & -\frac{\nu}{E} & -\frac{\nu}{E} & 0 & 0 & 0 \\ -\frac{\nu}{E} & \frac{1}{E} & -\frac{\nu}{E} & 0 & 0 & 0 \\ -\frac{\nu}{E} & -\frac{\nu}{E} & \frac{1}{E} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G} \end{bmatrix} \tag{7.9}$$

with the shear modulus related through

$$G = \frac{E}{2\,(1+\nu)} \tag{7.10}$$

For thermally isotropic materials the expansion coefficient is constant in all directions,thus

$$\hat{\boldsymbol{\epsilon}}^{th} = \begin{bmatrix} \alpha \\ \alpha \\ \alpha \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta T \tag{7.11}$$

The data input for the isotropic models is input using the ELAStic,ISOTropic and `THERmal,ISOTropic` commands as shown in Table 7.1. For an isotropic material it is not necessary to perform transformation of the elastic arrays since $\boldsymbol{C} = \hat{\boldsymbol{C}}$.

The types of elements for which elastic material models may be specified is indicated in Table 7.3.

7.3 Isotropic Finite Deformation Elastic Models

Finite deformation hyperelastic models are provided in *FEAPpv* 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 $\boldsymbol{X}$, and positions in the current configuration, denoted by $\boldsymbol{x}$. The motion of a point from the reference to the current configuration at time t is expressed as

$$\boldsymbol{x} = \boldsymbol{\varphi}(\boldsymbol{X}, t) \tag{7.12}$$

The deformation gradient is then defined as

$$\boldsymbol{F} = \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{X}} \tag{7.13}$$

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

$$\boldsymbol{C} = \boldsymbol{F}^T \boldsymbol{F} \tag{7.14}$$

and the left Cauchy-Green deformation tensor

$$\boldsymbol{b} = \boldsymbol{F}\, \boldsymbol{F}^T \tag{7.15}$$

A measure of strain is provided by the Green strain

$$\boldsymbol{E} = \frac{1}{2}\left(\boldsymbol{C} - \boldsymbol{1}\right) \tag{7.16}$$

The hyperelastic model expressed in terms of the strain energy function as a function of $\boldsymbol{C}$ is given by

$$\boldsymbol{S} = \frac{\partial W(\boldsymbol{C})}{\partial \boldsymbol{C}} \tag{7.17}$$

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

$$\boldsymbol{\sigma} = \frac{1}{J}\,\boldsymbol{F}\,\boldsymbol{S}\,\boldsymbol{F}^T \tag{7.18}$$

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

$$I_C = \operatorname{tr}\boldsymbol{C} \tag{7.19}$$

$$II_C = \frac{1}{2}\left(I_C^2 - \operatorname{tr}\boldsymbol{C}^2\right) \tag{7.20}$$

and

$$III_C = \det\boldsymbol{C} = J^2 \tag{7.21}$$

where J is $\det\mathbf{F}$ may be used to write the stored energy function.

In *FEAPpv* the isotropic elastic moduli are defined to match results from the small strain isotropic elastic models. Accordingly, they only require specification of the elastic modulus, E, and Poisson ratio, ν or, equivalently, the bulk modulus, K, and shear modulus, G.

7.3.1 St. Venant-Kirchhoff

The simplest model is a St. Venant-Kirchhoff model given by:

$$\boldsymbol{S} = \boldsymbol{D}\,\boldsymbol{E} \tag{7.22}$$

where $\boldsymbol{S}$ is the second Piola-Kirchhoff stress, $\boldsymbol{E}$ is the Green strain, and $\boldsymbol{D}$ are the elastic moduli. This model may be deduced from the stored energy function

$$W = \frac{1}{2}\,\boldsymbol{E}^T\boldsymbol{D}\,\boldsymbol{E} \tag{7.23}$$

For isotropy the model may be written in terms of the invariants of $\boldsymbol{E}$; however, the $\boldsymbol{D}$ will have the same structure as in an isotropic linear elastic material (see above).

The St. Venant-Kirchhoff model should not be used for problems in which very large compressive deformations are expected. The model gives identical results to the small deformation isotropic model if deformations are truly infinitesimal. It is also a good model to use if the displacements are large, but strains remain small. For situations where large elastic deformations are involved the `NEOHookean` or `MNEOhookean` models should be used.

Command	Type	Parameters
ELAStic	NEOHookean	E, ν
ELAStic	MNEOhookean	E, ν
ELAStic	STVK	E, ν

Table 7.2: Finite Deformation Elastic Material Data Inputs

7.3.2 Neo-Hookean and Modified Neo-Hookean Models

The stored energy functions for the finite deformation hyperelastic 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 $U(J)$ multiplied by a material parameter. The volumetric function is taken as

$$U(J) = (lnJ)^2 \tag{7.24}$$

The neo-Hookean hyperelastic model is deduced from the stored energy function

$$W = \left(K - \frac{2}{3}G\right) U(J) + \frac{1}{2} G \, (I_C - 3) \tag{7.25}$$

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

$$K = \frac{E}{3\,(1-2\nu)} \quad ; \quad G = \frac{E}{2\,(1+\nu)} \tag{7.26}$$

The data set to use this form is given by

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

A modified form to the neo-Hookean model is also available. The modified form defines a stored energy function which splits the two terms into pure volumetric behavior and pure deviatoric behavior. To accomplish the construction the deformation gradient is split into the product of a volumetric and deviatoric form as

$$\boldsymbol{F} = \boldsymbol{F}_{vol}\, \boldsymbol{F}_{dev} \tag{7.27}$$

where

$$\boldsymbol{F}_{vol} = J^{\frac{1}{3}}\mathbf{1} \tag{7.28}$$

and

$$\boldsymbol{F}_{dev} = J^{-\frac{1}{3}}\boldsymbol{F} \tag{7.29}$$

The stored energy function is then given as

$$W = K\,U(J) + \frac{1}{2}\,G\,\left(J^{-\frac{2}{3}}I_C - 3\right) \tag{7.30}$$

The parameters K and G are again specified by their small strain equivalent E and ν defined in Eq. 7.26.

The data set to use the modified form is given by

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

7.4 Rayleigh Damping

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

$$\boldsymbol{C} \;=\; a_0\,\boldsymbol{M} \;+\; a_1\,\boldsymbol{K} \tag{7.31}$$

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.

7.5 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).[9] At present, the implementation in *FEAPpv* is restricted to isotropic viscoelasticity for small deformation applications only in which time effects are included for the deviatoric stress components only. If we split the stress as:

$$\boldsymbol{\sigma} \;=\; \sigma_{vol}\,\mathbf{1} + \boldsymbol{\sigma}_{dev} \tag{7.32}$$

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

$$\boldsymbol{\epsilon} \;=\; \frac{1}{3}\,\theta\,\mathbf{1} + \boldsymbol{\epsilon}_{dev} \tag{7.33}$$

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

$$\boldsymbol{\sigma}_{dev} \;=\; 2\,G\int_{-\infty}^{t} \mu(t-\tau)\,\frac{d\boldsymbol{\epsilon}_{dev}}{d\tau}\,d\tau \tag{7.34}$$

where $mu(t)$ is a relaxation function. The term $Gmu(t)$ is called the relaxation modulus. In *FEAPpv* the relaxation function is represented by a Prony series

$$\mu(t) \;=\; \mu_0 + \sum_i \mu_i\,\exp\frac{-t}{\tau_i} \tag{7.35}$$

The τ_i are time parameters defining the relaxation times for the material and the μ_i are constant terms. Currently, *FEAPpv* limits the representation to three (3) exponential terms. The value of μ_0 is computed from

$$\mu_0 \;=\; 1 - \sum_i \mu_i \tag{7.36}$$

Thus, the elastic modulus G represents the instantaneous elastic response and $G\,\mu_0$ the equilibrium, or long time, elastic modulus. Only positivef 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 followin 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 integraion over each time step and leads to a simple recursion for each exponential term (e.g., see [10]). Additional details are also given in the *FEAP* Theory manual.[9]

7.6 Plasticity Models

Classical elasto-plastic material models are included in *FEAPpv* for small deformation problems. The stress for an elasto-plastic material may be computed by assuming an additive split of the strain as

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}^{el} + \boldsymbol{\epsilon}^{pl} \tag{7.37}$$

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

$$\dot{\boldsymbol{\epsilon}}^{pl} = \dot{\gamma}\,\frac{\partial F}{\partial \boldsymbol{\sigma}} \tag{7.38}$$

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

$$\boldsymbol{\alpha} = H_{kin}\,\boldsymbol{\epsilon}^{pl} \tag{7.39}$$

where $\boldsymbol{\alpha}$ is the back stress and H_{kin} is the kinematic hardening modulus. The isotropic hardening is taken in a linear and saturation form as

$$Y(e^{pl}) = Y_\infty + (Y_0 - Y_\infty)\exp(-\beta\, e^{pl}) + H_{iso}\, e^{pl} \tag{7.40}$$

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

$$e^{pl} = \int_0^t \dot{\gamma}\, d\tau \tag{7.41}$$

In *FEAPpv* the discrete problem is solved using a closest point return map algorithm (e.g., see [11, 12, 13]).

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
```

7.7 Heat Conduction Material Models

For thermal analysis a linear heat conduction capability is included in *FEAPpv*. The constitutive equation is given by a linear Fourier model in which the heat flux $\boldsymbol{q}$ is related to the thermal gradient $\boldsymbol{h}$ by the relation

$$\boldsymbol{q} \;=\; -\,\boldsymbol{K}\,\boldsymbol{h} \tag{7.42}$$

where, in the principal directions,

$$\boldsymbol{K} \;=\; \begin{bmatrix} K_1 & 0 & 0 \\ 0 & K_2 & 0 \\ 0 & 0 & K_3 \end{bmatrix} \tag{7.43}$$

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

Command	Solid	Truss	Frame	Plate	Shell	Membrane	Thermal
ELAStic	X	X	X	X	X	X	-
PLAStic	S	X	-	-	-	-	-
VISCoelastic	S	X	-	-	-	-	-
THERmal	S	-	-	X	-	X	-
FOURier	S	-	-	-	-	-	X
DENSity	X	X	X	X	X	X	X
ANGLe	X	-	-	X	X	X	X

Table 7.3: Material Commands vs. Element Types. X=all, F=finite, S=small.

7.8 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 *FEAPpv*

uses a lumped matrix. If $\boldsymbol{M}_{cons}$ is the consistent matrix and $\boldsymbol{M}_{lump}$ is the diagonal lumped matrix, the interpolated matrix is defined as:

$$\boldsymbol{M}_{interp} \;=\; (1 \,-\, a)\, \boldsymbol{M}_{cons} \;+\; a\, \boldsymbol{M}_{lump} \tag{7.44}$$

The type of mass and, where required, the parameter a are input using the `MASS` command as shown in Table 7.4 and the elements which are affected by the command are indicated in Table 7.5.

Command	Type	Parameters
MASS	LUMPed	
MASS	CONSistent	
MASS		a

Table 7.4: Material Model Mass Related Inputs

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

Table 7.5: Mass Command vs. Element Types

7.9 Element Cross Section and Load Specification

7.9.1 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 7.6. 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 7.6.

Element loads for surface pressure and body force are input using the `LOAD` and `BODY` force commands as shown in Table 7.6.

The types of elements affected by the `THICkness`, `LOAD` and `BODY` commands is indicated in Table 7.7.

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
LOAD	normal	q

Table 7.6: 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	-
LOAD	-	-	-	X	X	X	-

Table 7.7: Geometry and Loads vs. Element Types

7.10 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 7.8 and the affected element types in Table 7.9. 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, *FEAPpv* 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.

FEAPpv includes capabilities to solve finite deformation problems using only the 2-d `SOLId`, `FRAMe`, and `TRUSs` 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) or `MIXEd` types. The data options for these are indicated in Table 7.8 and the affected element types in Table 7.9.

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

Table 7.8: Miscellaneous Material Model Inputs

Command	Solid	Truss	Frame	Plate	Shell	Membrane	Thermal
QUADrature	X	-	-	-	X	X	X
PENAlty	-	-	-	-	-	-	-
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 7.9: Miscellaneous Material Commands vs. Element Types

Chapter 8

END AND MISCELLANEOUS BASIC MESH COMMANDS

The above set of commands are part of the basic mesh input commands available in *FEAPpv* 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 *FEAPpv* 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 *FEAPpv* 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 9 and procedures to solve and plot results are presented in Chapters 10 and 11, respectively.

Chapter 9

MESH MANIPULATION COMMANDS

Once an initial mesh is completely defined it may be further processed to: (1) merge nodes with the same coordinates using the `TIE` command, or (2) force a sharing of degrees-of-freedom using the `LINK` command. 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 *FEAPpv* later executes the commands in a definite order.

9.1 The TIE Command

The ability to merge nodes which have the same coordinates permits the generation of a mesh in separate parts 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. A range of node numbers to search may be specified also. For example, if the merge is to be done only for nodes numbered between 34 and 65 the command is issued as:

```
TIE,, 34, 65
```

It is however, not possible to merge nodes from two different ranges of numbers.

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
TIE MATErial 2  2
```

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`). The basic command to merge parts in *Region m* to those in *Region n* is

```
TIE REGIon m  n
```

The parameters `m` and `n` 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 plot 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 output file. This is different than creating common solution values for degree of freedoms associated with two nodes using the `LINK` option described below. In a link option the node is not deleted, and thus projections may create a different solution at the two nodes.

9.2 The LINK Command

The link option 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.[1] The link is performed based on node numbers using the `LINK` command.

Chapter 10

COMMAND LANGUAGE PROGRAMS

FEAPpv 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 C 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 *FEAPpv* the user issues the command BATCh or for an interactive execution mode the command INTEractive. A solution is terminated by the command END (QUIT or just Q also may be used in interactive mode).

Thus, the input file must contain at least one set of

```
BATCh
 ....  ! Solution specification steps
END
```

or

```
INTEractive
```

for any solution process to be possible.

More than one BATCh-END and/or INTEractive-END sequence may be used during the solution process.

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 a basic set of commands quite sophisticated solution algorithms may be constructed. Each of the commands may be issued in a lower or upper case mode. For example, 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 *FEAPpv* 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 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.

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

10.1 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 *FEAPpv* 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 tangent and residual 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 `PRINt` and `NOPRint` 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 C 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 11.

10.1.1 Solution of Non-linear Problems

The solution of non-linear problems is often performed using Newton's method which solves the problem

$$\mathbf{R}(\mathbf{u}) = \mathbf{0} \tag{10.1}$$

using the iterative algorithm

1. Set initial solution

$$\mathbf{u}^0 = \mathbf{0} \tag{10.2}$$

2. Solve the set of equations

$$\mathbf{K}\,\Delta\mathbf{u}^i = \mathbf{R}(\mathbf{u}^i) \tag{10.3}$$

   where

$$\mathbf{K} = -\frac{\partial \mathbf{R}}{\partial \mathbf{u}} \tag{10.4}$$

3. Update the solution iterate

$$\mathbf{u}^{i+1} = \mathbf{u}^i + \Delta\mathbf{u}^i \tag{10.5}$$

The steps are repeated until a norm of the solution is less than some tolerance.

FEAPpv 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

$$E^i = \Delta \mathbf{u}^i \cdot \mathbf{R}^i \tag{10.6}$$

with convergence assumed when

$$E^i < tol\, E^0 \tag{10.7}$$

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, *FEAPpv* transfers to the statement following the `NEXT` command. If convergence is not achieved in 10 iterations, *FEAPpv* 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 *FEAPpv*, 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 *FEAPpv* 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 *FEAPpv* 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.

10.1.2 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. *FEAPpv* includes several options for solving linear equations. The default solution scheme is the variable band, profile scheme discussed in Chapter 16 of *The Finite Element Method, Vol 1*, 5th edition.[1] 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.

An iterative option is available based on a preconditioned conjugate gradient scheme (PCG method). The PCG method is applicable to symmetric, positive definite coefficient arrays only. Thus, only the `TANG`ent command may be used. The PCG with diagonal preconditioner is requested by the command `ITER`ation before the first `TANG`ent command. 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 beam; 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).

10.2 Transient Solutions

FEAPpv 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 `DISP`lacement, transient algorithms permit the output of `VELO`city and `ACCE`leration (see Appendix C).

FEAPpv 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. *FEAPpv* solves three basic types of transient formulations:

10.2.1 Quasi-static solutions

The governing equation to be solved by the quasi-static option is expressed as:

$$\mathbf{R}(t) \;=\; \mathbf{F}(t) \;-\; \mathbf{P}(\mathbf{u}(t)) \;=\; \mathbf{0} \tag{10.8}$$

where, for example, the $\mathbf{P}$ vector is given by the stress divergence term of a solid mechanics problem as:

$$\mathbf{P}(\mathbf{u}(t)) \;=\; \mathbf{P}_\sigma \;=\; \int_\Omega \mathbf{B}^T \boldsymbol{\sigma}\, dV \tag{10.9}$$

The solution options for this form are:

1. The default algorithm which solves

   $$\mathbf{R}(t_{n+1}) \;=\; \mathbf{F}(t_{n+1}) \;-\; \mathbf{P}(\mathbf{u}(t_{n+1})) \;=\; \mathbf{0} \tag{10.10}$$

   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

   $$\mathbf{R}(t_{n+\alpha}) \;=\; \mathbf{F}(t_{n+\alpha}) \;-\; \mathbf{P}(\mathbf{u}(t_{n+\alpha})) \;=\; \mathbf{0} \tag{10.11}$$

   where

   $$\mathbf{u}(t_{n+\alpha}) \;=\; \mathbf{u}_{n+\alpha} \;=\; (1-\alpha)\,\mathbf{u}_n \;+\; \alpha\,\mathbf{u}_{n+1} \tag{10.12}$$

   and

   $$\mathbf{F}(t_{n+\alpha}) \;=\; \mathbf{F}_{n+\alpha} \;=\; (1-\alpha)\,\mathbf{F}_n \;+\; \alpha\,\mathbf{F}_{n+1} \tag{10.13}$$

   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.

10.2.2 First order transient solutions

The governing equation to be solved for first order transient solutions is expressed as:

$$\mathbf{R}(t) \;=\; \mathbf{F}(t) \;-\; \mathbf{P}(\mathbf{u}(t), \dot{\mathbf{u}}(t)) \;=\; \mathbf{0} \tag{10.14}$$

where, for example, $\mathbf{u}$ are the nodal temperatures $\mathbf{T}$ and the $\mathbf{P}$ vector is given by:

$$\mathbf{P} \;=\; \int_{\Omega} (\nabla N)^T \, \mathbf{q} \, dV \;+\; \mathbf{C}\,\dot{\mathbf{T}} \tag{10.15}$$

with $\mathbf{C}$ the heat capacity matrix.

The solution options for this form are:

1. A backward Euler method which solves the problem

   $$\mathbf{R}(t_{n+1}) \;=\; \mathbf{F}(t_{n+1}) \;-\; \mathbf{P}(\mathbf{u}(t_{n+1}), \dot{\mathbf{u}}_{n+1}(t)) \;=\; \mathbf{0} \tag{10.16}$$

   where

   $$\dot{\mathbf{u}}_{n+1} \;=\; \frac{1}{\Delta t}[\mathbf{u}_{n+1} \;-\; \mathbf{u}_n] \tag{10.17}$$

   The command:

   ```
   TRANsient,BACK
   ```

   is used to specify this solution option.

2. A generalized midpoint method which solves the problem

   $$\mathbf{R}(t_{n+\alpha}) \;=\; \mathbf{F}(t_{n+\alpha}) \;-\; \mathbf{P}(\mathbf{u}(t_{n+\alpha}), \dot{\mathbf{u}}_{n+\alpha}(t)) \;=\; \mathbf{0} \tag{10.18}$$

   where

   $$\dot{\mathbf{u}}_{n+\alpha} \;=\; \frac{1}{\Delta t}[\mathbf{u}_{n+1} \;-\; \mathbf{u}_n] \tag{10.19}$$

   This solution option is selected using the command

   ```
   TRANsient,GN11,alpha
   ```

   where $0 < \alpha \leq 1$ (default is 0.5); $\alpha = 1$ should produce answers identical to those from the backward Euler option. Alternatively, the `SS11` algorithm may be used.[1]

10.2.3 Second order transient solutions

The governing equation to be solved for second order transient solutions is expressed as:

$$\mathbf{R}(t) \;=\; \mathbf{F}(t) \;-\; \mathbf{P}(\mathbf{u}(t), \dot{\mathbf{u}}(t), \ddot{\mathbf{u}}(t)) \;=\; \mathbf{0} \tag{10.20}$$

where, for example, the $\mathbf{P}$ vector is given by:

$$\mathbf{P} \;=\; \mathbf{P}_\sigma \;+\; \mathbf{C}\,\dot{\mathbf{u}} \;+\; \mathbf{M}\,\ddot{\mathbf{u}} \tag{10.21}$$

with $\mathbf{C}$ the damping and $\mathbf{M}$ the mass matrix.

The solution options for second order problems are:

1. A Newmark method[14] which solves the problem

   $$\mathbf{R}(t_{n+1}) \;=\; \mathbf{F}(t_{n+1}) \;-\; \mathbf{P}(\mathbf{u}_{n+1}, \mathbf{v}_{n+1}, \mathbf{a}_{n+1}) \;=\; \mathbf{0} \tag{10.22}$$

   where

   $$\mathbf{v}_n \;=\; \dot{\mathbf{u}}_n \qquad ; \qquad \mathbf{a}_n \;=\; \ddot{\mathbf{u}}_n \tag{10.23}$$

   with updates computed as:

   $$\mathbf{u}_{n+1} \;=\; \mathbf{u}_n \;+\; \Delta t\,\mathbf{v}_n \;+\; \Delta t^2\,[\,(0.5-\beta)\,\mathbf{a}_n \;+\; \beta\,\mathbf{a}_{n+1},\;] \tag{10.24}$$

   and

   $$\mathbf{v}_{n+1} \;=\; \mathbf{v}_n \;+\; \Delta t\,[\,(1-\gamma)\,\mathbf{a}_n \;+\; \gamma\,\mathbf{a}_{n+1}\,] \tag{10.25}$$

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

   ```
   TRANsient,NEWMark
   ```

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

   ```
   TRANsient
   ```

   or

   ```
   TRANsient,GN22
   ```

   or

```
TRANsient,SS22
```

also selects the Newmark algorithm.[1] Default values for classical Newmark are $\beta = 0.25$ and $\gamma = 0.5$. Default values for the `GN22` or `SS22` algorthim are $\theta_1 = \theta_2 = 0.5$.

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. The `GN22` and `SS22` algorithms are self starting and do not require the above special starting condition.

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 `BATCh` solution option appears immediately after the `END` command. In an interactive mode a user receives a prompt to specify the data.

FEAPpv 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).

10.3 Transient Solution of Linear Problems

The solution of second order linear equations by the finite element method leads to the set of equations

$$\mathbf{M}\,\ddot{\mathbf{u}}(t) \;+\; \mathbf{C}\,\dot{\mathbf{u}}(t) \;+\; \mathbf{K}\,\mathbf{u}(t) \;=\; \mathbf{F}(t) \tag{10.26}$$

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. 10.26 may be constructed by partitioning the solution into the parts

$$\mathbf{u} \;=\; \begin{bmatrix} \mathbf{u}_u \\ \mathbf{u}_s \end{bmatrix} \tag{10.27}$$

where $(\cdot)_u$ denotes an unknown part and $(\cdot)_s$ a specified part. With this division, the equations are then written in the form:

$$\begin{bmatrix} \mathbf{M}_{uu} & \mathbf{M}_{us} \\ \mathbf{M}_{su} & \mathbf{M}_{ss} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{u}}_u \\ \ddot{\mathbf{u}}_s \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{uu} & \mathbf{C}_{us} \\ \mathbf{C}_{su} & \mathbf{C}_{ss} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_u \\ \dot{\mathbf{u}}_s \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{us} \\ \mathbf{K}_{su} & \mathbf{K}_{ss} \end{bmatrix} \begin{bmatrix} \mathbf{u}_u \\ \mathbf{u}_s \end{bmatrix} \;=\; \begin{bmatrix} \mathbf{F}_u \\ \mathbf{F}_s \end{bmatrix} \tag{10.28}$$

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. *FEAPpv* uses only an integration in time directly using a numerical step-by-step procedure (e.g., the Newmark method).

10.4 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
FORCe
 ...
END
FORCE
 ...
END
 ...
```

in which a set of new forces appears for each time step performed. The use of the `MESH` command within a solution strategy permits the alteration of any nodal or element data. It is not permitted to change the size of the problem by adding new nodes or elements (elements may be `ACTI`vated or `DEAC`tivated based on region descriptions); however, nodal forces, displacements, boundary restraint codes, etc. may be changed. Material paramters 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

$$\mathbf{F}(t) = p_j(t)\,\mathbf{F}_j \tag{10.29}$$

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 loadings to be input. If `Jj` is zero only the `Ji` function is to be input. A functional type of proportional loading is

$$p(t) = a_0 + a_1 t + a_2[\sin a_3(t - t_{min})]^k \quad ; \quad t_{min} \le t \le t_{max} \tag{10.30}$$

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:

$$t_{min} = 0 \ ; \ t_{max} = 10^6 \ ; \ a_1 = 1 \ ; \ k = a_0 = a_2 = a_3 = 0 \tag{10.31}$$

A piecewise linear set of values may be input using the Type 2 proportional loading function which is specified by a PROPortional 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/factor pairs on each record. Input terminates with a blank record.

10.5 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 ARCLength 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.

10.6 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 and also to enforce incompressibility constraints in some of the continuum elements. The use of large penalty parameter values in some material models and some finite deformation analyses makes 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 pp 358 of *The Finite Element Method, Vol 1*, 4th ed., by Zienkiewicz & Taylor. The command language program to perform an augmented solution is given by:

```
LOOP,augment,n-augm
  LOOP,newton,n-iter
    TANGent,,1
  NEXT,iter
  AUGMent
NEXT,augment
```

The number of augmented iterations (n-augm) should be kept quite small as convergence of the iteration process is only checked by the TANGent command. If convergence is achieved in this loop execution passes to the AUGMent command and another augmentation is performed until the n-augm augmentation iterations are performed.

10.7 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

Type	n_1	n_2	x	y	z
disp	Node	dof	x	y	z
velo	Node	dof	x	y	z
acce	Node	dof	x	y	z
reac	Node	dof	x	y	z
stre	Elmt	component	-	-	-
arcl	Node	dof	x	y	z
cont	Node	dof	x	y	z
ener	Comp	print	-	-	-

Table 10.1: Tplot types and parameters

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 Table 10.1.

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.

10.8 Viewing Solution Data: SHOW Command

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

DICTionary opation 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.

10.9 Reexecuting 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.

10.10 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.

10.11 Output of Element Arrays

When solving problems difficulties often occur for which additional information is needed about an element. *FEAPpv* 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 EPRInt. For example, after a TANGent command the use of EPRInt 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 immediatly after the MASS command.

Chapter 11

PLOT OUTPUTS

FEAPpv provides for the construction of plots to represent features of the problem and its solution. This section contains specific instructions for constructing graphical views for a solution.

11.1 Screen Plots

FEAPpv presents graphics to the screen when starting the program. Options include an X-window mode and a PC-window mode. 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
```

followed by the sequence of plot commands to be executed (for clarity, a prompt indicating the solution mode is shown before each command; however, only the command is entered by the user). When in the interactive mode the `PLOT` is not issued as part of the command. Thus, the command

```
Plot> MESH
```

will display the mesh. 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 its material number. Additional options may exist for these and subsequent commands as described in Appendix C; however, below some of the options to construct basic plots are described.

To place the nodes on the screen while in interactive mode only the command

```
Plot> NODE
```

is given. To place the number for node 5 only, the command

```
Plot> NODE,5
```

is used. 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.

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 original 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`. 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 only be displayed using the `BOUN,dir` form.

Once a solution is performed using the command language features described in Chapter 10 it is possible to display several features of the solution. Vectors of the nodal generalized displacements may be shown using the command `DISPlacement`. Contours of the displacements are 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.

Contours of element variables, such as stresses, may be constructed using the command `STREss,comp` where `comp` is the component to be plotted. For *FEAPpv* solid elements the stress components are ordered as shown in Table 11.1.

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

COMP	Description
1	11-Stress
2	22-Stress
3	33-Stress
4	12-Stress
5	23-Stress
6	31-Stress

Table 11.1: 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 11.2: Component number for solid element principal stress value

`ESTRess,comp` command. Projected principal values of stresses may also be displayed using the command `PSTRess,comp` where the components are ordered as shown in Table 11.2

The directions of the principal axes at nodes may be shown using the command `PRAXis`.

It is also possible to show all of the above plots on a deformed position of the mesh by using the command

```
DEFOrm,factd,freeze,facte
```

before giving any of the above commands. The parameters `factd` and `facte` are scale factors for displacements and eigen-vectors, respectively. A non-zero value for the parameter `freeze` will keep the values of original scaling distances and, thus, permits a deformed mesh over an undeformed mesh with the same scaling. Similarly, the command `UNDE,,freeze` returns plots to an undeformed configuration without rescaling the plot.

To plot an eigen-vector it is necessary to provide the `facte` scaling using the `DEFOrm` command before issuing the eigen-vector plot command `EIGVector,num` where `num` is the number of the vector to plot. The ordering for `num` is the same as that for the eigen-values computed by the `SUBS`pace solution command.

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`.

11.2 PostScript Plots

FEAPpv 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 *FEAPpv* logo is normally not placed in the file.

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 C for more information on specifying the perspective view data.

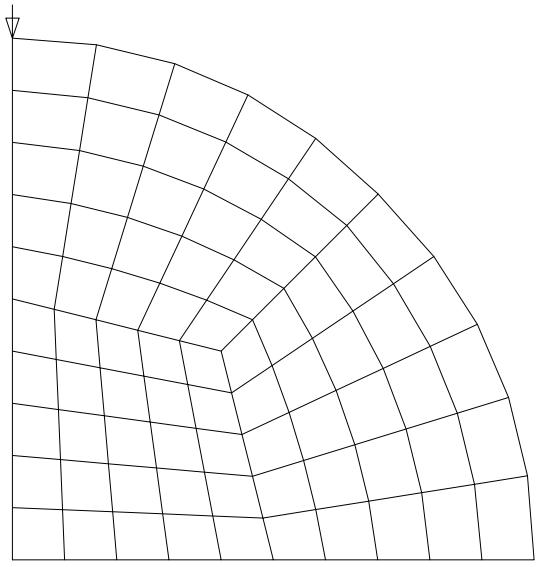

Figure 11.1: Mesh for Circular Disk. 75 Elements

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 11.1 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 11.2. To get contours for the velocity or acceleration the `CONT`our command is replaced by `VELO`city or `ACCE`leration, respectively.

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>
```

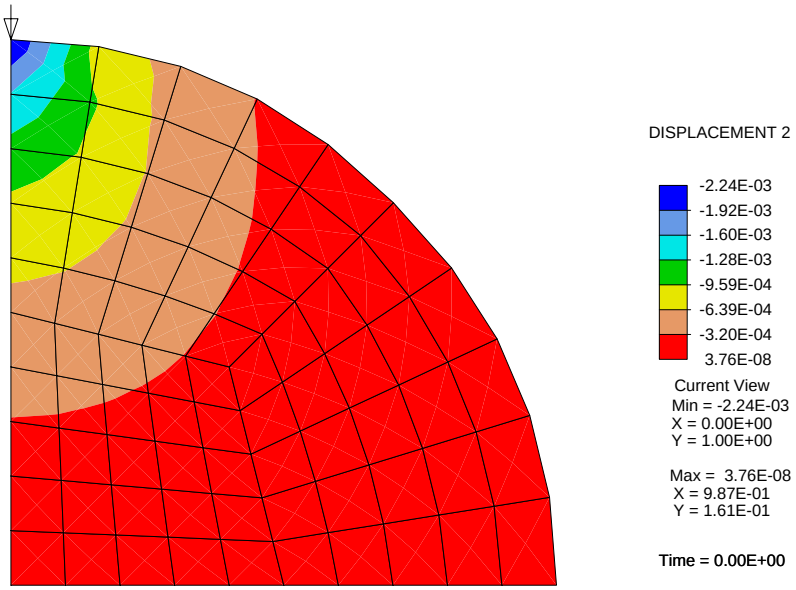

Figure 11.2: Contours of Vertical Displacement for Circular Disk

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

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

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

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

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

[7] 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.

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

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

[10] 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.

[11] 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.

[12] 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.

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

[14] N. Newmark. A method of computation for structural dynamics. *Journal of the Engineering Mechanics Division*, 85:67–94, 1959.

Appendix A

Mesh Manual Pages

FEAPpv has several options which may be used to input data to analyize a wide range of finite element problems in 1 to 3 dimensions. The following pages summarze the commands which are available to input specific parts of the mesh data. 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.

```
feappv [ title of problem for printouts, etc.]
     numnp,numel,nummat,ndm,ndf,nen,npd,nud,nad
```

Each problem to be solved by *FEAPpv* initiates with a single record which contains the characters `FEAPpv` (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 `FEAPpv` record may be preceded by `PARA`meter specifications (see parameter input manual page).

Immediately following the `FEAPpv` 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.
npd	– Maximum number of parameters for element properties. (default 200).
nud	– Maximum number of parameters for user element properties. (default 50).
nad	– Increases size of element arrays to *ndf*×*nen*+*nad*.

For many problems it is not necessary to specify values for *numnp*, *numel*, or *nummat*. *FEAPpv* 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 `OPTI`mize 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,SPAR`se, 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 `ITER`ation solution command.

If necessary, the main subprogram, `program feappv`, can be recompiled with a larger value set for the `parameter mmax` controling the size of blank common.

```
angl
   node1,ngen1,angl(node1)
   node2,ngen2,angl(node2)
   <etc,,terminate with blank record>
```

The `ANGL`e command is used to specify angles (degrees) for sloping nodal boundary conditions as shown in Fig. A.1. For each node `I` to be specified a record is entered with the following information:

node	– the number of the I-node to be specified
ngen	– the increment to the next node, if generation is used, otherwise 0.
angl(node)	– value of angle new 1-coordinate makes with x(1,node).

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

node1, node1+ngen1, node1+2×ngen1, ... , node2

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

The degrees-of-freedom associated with the sloping boundary may differ from element to element as described in the element manuals. The default will be the first two degrees-of-freedom (2 and 3-D problems) which are affected by the sloping condition. Both force and displacement values will be assumed to be given in the rotated frame. To activate the rotated boundary condition use the `BOUN`dary-, `FORC`e-, `DISP`lacement- etc. command.

Angle conditions may also be specified using the `EANG`le and `CANG`le commands.

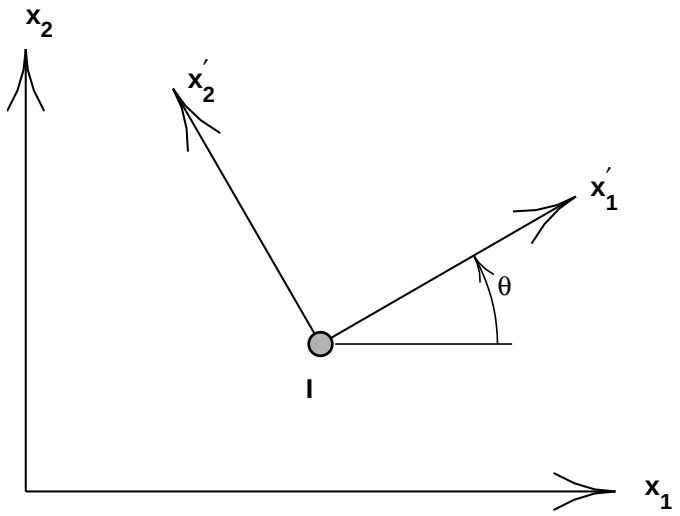

Figure A.1: Coordinate rotation for nodes

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.

BLENd FEAPpv MESH INPUT COMMAND MANUAL

```
blen (Surface in 2 or 3-D)
   surf,r-inc,s-inc,[node1,elmt1,mat],b-type
     (snode(i),i=1,4)
 blen (3-D Solid)
   soli,r-inc,s-inc,t-inc,[node1,elmt1,mat],b-type
     (snode(i),i=1,8)
```

FEAPpv 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, volume 1 pp 181 ff. Each super node is defined by an input of the following information:

The `BLEN`d data input segment may be used to generate:

1. 4-node quadrilateral elements in 2 or 3-D.
2. 8-node bricks in 3-D.

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. *FEAPpv* searches the list of edges defined by the the `SIDE` command. If a match is found it is used as the patch edge. If no match is found *FEAPpv* will define a straight edge with linear equal increment interpolation used to define the spacing of nodes in the finite element mesh. Care must be used in defining any specified sides in order to avoid errors from this automatic generation.

For three dimensional solid patches the same technique is used; however, now it is necessary to define eight vertex super-nodes to define the blend patch. The eight nodes are numbered by any right-hand rule sequence. The r-direction and s-direction are defined in the same way as for the surface patch. The third t-direction ξ_3 is along the direction defined by the first to fifth vertex super-node.

The r-, s-, and t-increments are used in the same manner as for the `BLOCk` command. Care must be used in defining the increments along any direction which involves a multi-segment interpolation to ensure that the total number of intervals from the side definition for the mult-segment agrees with the number of increments specified with the `BLEN`d command.

Examples for two and three dimensional blends are illustrated in the *FEAPpv* 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.

BLOCk FEAPpv MESH INPUT COMMAND MANUAL

```
bloc (Line in 1,2,or3-D)
   type,r-inc,,node1,[elmt1,mat,r-skip],b-type
     1,x1,y1,z1 (only ndm coordinates required)
     2,x2,y2,z2
   etc.,blank record after all nodes are input
   bloc (Surface in 2 or 3-D)
   type,r-inc,s-inc,node1,[elmt1,mat,r-skip],b-type
     1,x1,y1,z1 (only ndm coordinates required)
     2,x2,y2,z2
   etc.,blank record after all nodes are input
   bloc (3-D Solid)
   type,r-inc,s-inc,t-inc,node1,[elmt1,mat],b-type
     1,x1,y1,z1
     2,x2,y2,z2
   etc.,blank record after all nodes are input
```

The BLOCk data input segment is used to generate:

1. 2-node line elements in 1, 2, or 3-D.
2. 4 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 BLOCk is developed from a master element which is defined by an isoparametric 4 to 9 node mapping function in terms of the two natural coordinates, r (or ξ_1) and s (or ξ_2), respectively. The node numbers on the master element of each patch defined by BLOCk are specified according to Figure A.2. The four corner nodes of the master element must be

specified, the mid-point and central nodes are optional. The spacing between the r-increments and s-increments may be varied by an off-center placement of mid-side and central nodes. Thus, it is possible to concentrate nodes and elements into one corner of the patch generated by `BLOCk`. The mid-nodes must lie within the central-half of the r-direction and the s-direction to keep the isoparametric mapping single valued for all (r,s) points. For a line patch, the nodes and 2 node elements are defined from a 1-2 master linear line patch or a 1-3-2 master quadratic line patch. The *s-inc* parameter must be 0 for this option. For a 3-D solid the patch is described by an 8 to 27-node master solid element where the corner nodes are required and mid-edge/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.

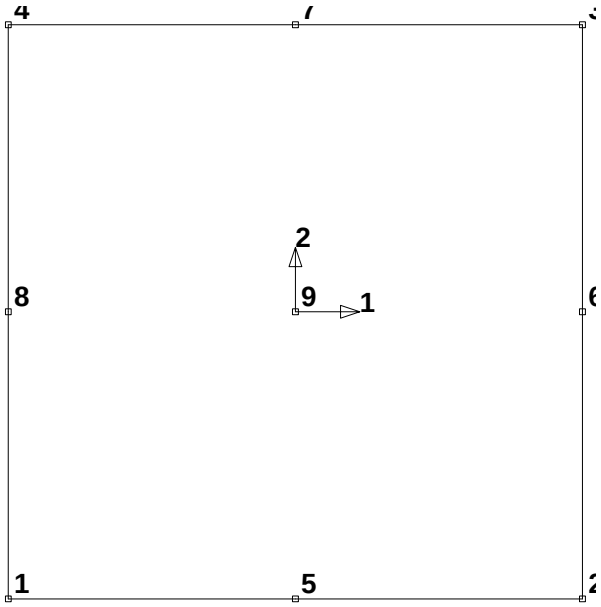

Figure A.2: Node Specification on 2D Master Block.

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 `FEAPpv` 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 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

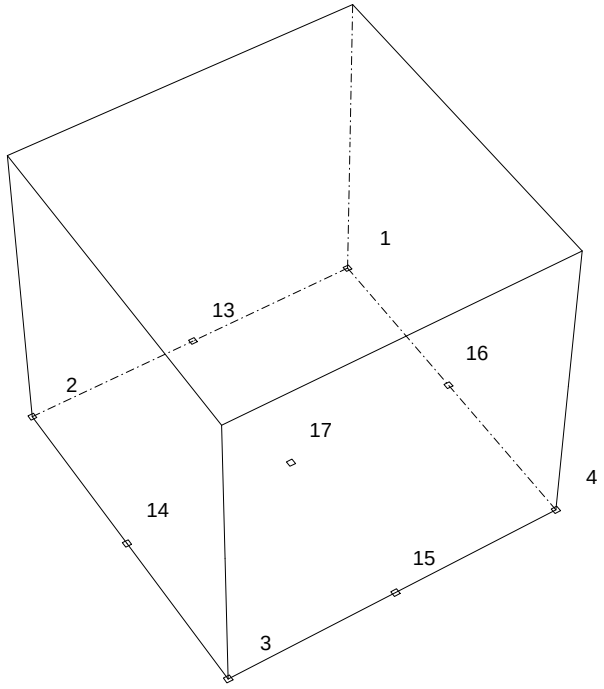

Figure A.3: Node Specification on 3D Master Block.

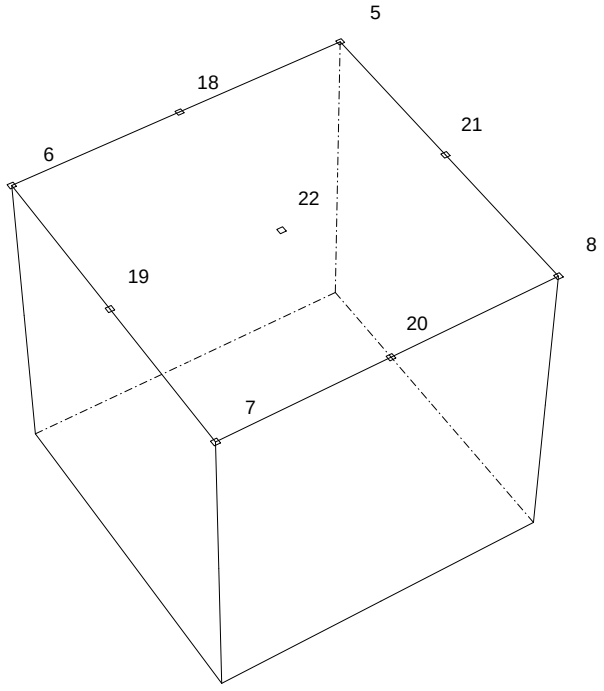

Figure A.4: Node Specification on 3D Master Block.

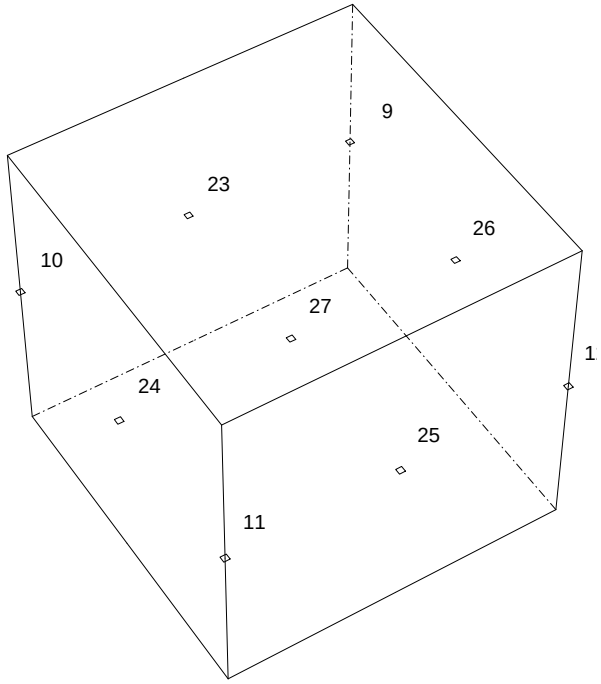

Figure A.5: Node Specification on 3D Master Block.

```
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 interpretted as follows:

id(i,node) $= 0$	a force will be an applied load to dof (default).
id(i,node) $\neq 0$	a displacement will be imposed to dof.

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

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each boundary restraint will be as follows:

id(i,node1) =	0 or positive	→ id(i,node1+ngen1) = 0
id(i,node1) =	negative	→ id(i,node1+ngen1) = −1

With this convention the value of a zero *id(i,node2)* will be set negative whenever the value of *id(i,node1)* starts negative. Accordingly, it is necessary to assign a positive value for the restraint code to terminate a generation sequence (e.g., when it is no longer desired to set a dof to be restrained). Alternatively, an i-dof may be eliminated for all nodes by using the generation sequence:

node	ngen	dofs				
		1	...	i	...	ndf
1	1	0	...	-1	...	0
numnp	0	0	...	+1	...	0

Subsequent records may then be used to assign values to other degree-of-freedoms.

Boundary condition restraints may also be specified using the `EBOU`nd or `CBOU`nd commands.

Example: BOUNdary

Consider a problem which has 3 degrees of freedom at each node. The sequence of records:

```
BOUNdary conditions
  1 4 1 -1 0
 13 0 0  1 1
```

will define boundary conditions for nodes 1, 5, 9 and 13 and the restraint codes will have the following values

node	DOFS		
	1	2	3
1	1	-1	0
5	0	-1	0
9	0	-1	0
13	0	1	1

Any degree of freedom with a non-zero bounday 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 *FEAPpv* 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 *cartesian* reference coordinates for a node. 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, `EANGL`e processed second and the `CANGL`e data processed last. The value defined last is used for any analysis. The data input by `CANG` replaces previously assigned values. 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:

```
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:

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

For three dimensional problems it is possible to specify the segment to which the boundary conditions are applied. The segment is specified as a *surface*. The data is specified as:

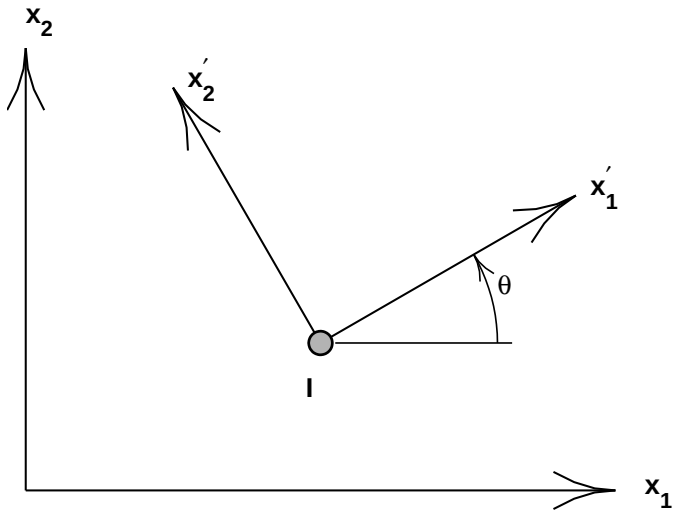

Figure A.6: Coordinate rotation for node I

```
SURFace
  1,x1,y1,z1,angle1
  2,x2,y2,z2,angle2
  3,x3,y3,z3,angle3
  4,x4,y4,z4,angle4
```

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: CANGle

In a two-dimesional 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
```

Note that 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. Users should check (using graphics plot mode) that the correct node(s) are selected.

CBOUndary FEAPpv MESH INPUT COMMAND MANUAL

```
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 or a *quadratic* line. The input values are saved in files and searched after the entire mesh is specified. The data is order dependent with data defined by `BOUNe` processed first, `EBOUle` processed second and the `CBOUle` 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:

```
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:

```
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:

```
SURFace,(ibc(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` to show the locations of conditions).

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 angles must be input in degrees. The *cartesian* option resets the coordinate system to a cartesian frame.

CDISplacement FEAPpv MESH INPUT COMMAND MANUAL

```
cdis,[set,add]
   gap,value
   node,(x(i),i=1,ndm),(d(j),j=1,ndf)
   <etc.,terminate with a blank record>
```

The specified displacement boundary conditions may be set using the reference coordinates for a *node*. The input values are saved in files and searched after the entire mesh is specified. After use files are deleted. The data is order dependent with data defined by `DISP` processed first, `EDIS` processed second and the `CDIS` data processed third and data specified by the `CSURf` 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 the last releases of FEAPpv.

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 `FORCe` processed first, `EFORce` processed second and the `CFORce` data processed last. The value defined last is used for any analysis.

The `CFOR` 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.

```
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 appled to nodal forces and displacments 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 `FPROp` processed first, `EPROp` processed second and the `CPROp` 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
   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):

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

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.

CSURface FEAPpv MESH INPUT COMMAND MANUAL

```
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 *FEAPpv* 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* loadings. 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.

DISPlacements FEAPpv MESH INPUT COMMAND MANUAL

```
disp
   node1,ngen1,(d(i,node1),i=1,ndf)
   node2,ngen2,(d(i,node2),i=1,ndf)
   <etc.,terminate with blank record>
```

The DISPlacement command is used to specify the values for nodal boundary displacements. For each node to be specified a record is entered with the following information:

node	– Number of node to be specified
ngen	– Increment to next node, if generation is used, otherwise 0.
d(1,node)	– Value of displacement for 1-dof
d(2,node)	– Value of displacement for 2-dof
	etc., to *ndf* directions

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

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each displacement will be a linear interpolation between the *node1* and *node2* values for each degree-of-freedom.

While it is possible to specify both the force and the displacement applied to a node, only one can be active during a solution step. The determination of the active value is determined from the boundary restraint condition value. If the boundary restraint value is zero and you use one of the force-commands a force value is imposed, whereas, if the boundary restraint value is non-zero and you use one of the displacement-commands a displacement value is imposed. (See BOUNdary, CBOUndary, or EBOUndary pages for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value. It is not possible, to specify a displacement by using the combination of a force-command with a non-zero boundary restraint value, as it was in the last releases of *FEAPpv*. For further information see the CDISplacement 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>
```

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.). 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.

EBOUndary FEAPpv MESH INPUT COMMAND MANUAL

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

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.). 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).
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 $x_3 = z = 0$ 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 0.0 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.

EDISplacement FEAPpv MESH INPUT COMMAND MANUAL

```
edis
   i-coor,xi-value,(d(j),j=1,ndf)
   <etc.,terminate with a blank record>
```

The values of boundary displacment 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.). 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 *FEAPpv*. 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.

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

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.). The data to be supplied during the definition of the mesh consists of:

i-coor	– Direction of coordinate (i.e., 1 = x, 2 = y, etc.)
xi-value	– Value of i-direction coordinate to be used during search (a tolerance of 1/1000 of mesh size is used during search, any coordinate within the gap is assumed to have the specified value).
f(1)	– Value of force for dof's
f(2)	
...	
f(ndf)	

While it is possible to specify both the force and the displacement applied to a node, only one can be active during a solution step. The determination of the active value is determined from the boundary restraint condition value. If the boundary restraint value is zero and you use one of the force-commands a force value is imposed, whereas, if the boundary restraint value is non-zero and you use one of the displacement-commands a displacement value is imposed. (See `BOUN`dary, `CBOU`ndary, or `EBOU`ndary pages for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value.

The `EFOR` command may be used with two options. Using the `EFOR,SET` option replaces previously defined forces at a node by the pattern specified. Using the `EFOR,ADD` option accumulates the forces with previously defined values. The default mode is `ADD`.

Force conditions may also be specified using the `FORC`e and `CFOR`ce commands. The data is order dependent with data defined by `FORC`e processed first, `EFOR`ce processed second and the `CFOR`ce data processed last. The value defined last is used for any analysis.

Example: EFORce

All the nodes located on the $x_3 = z = 10$ plane are to have a common specified horizontal force value. (Note that this is not a common case as end nodes on equally spaced intervals would have different values from other nodes.) This may be specified using the command set:

```
EFORce
  3 10.0 -12.5
```

where -12.5 is the value of each force

```
elem                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 **ELEM**ent 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 **BLOC**k and **BLEN**d inputs to generate elements in a mesh. For each element to be specified by an **ELEM**ent 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 *FEAPpv* 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.

END FEAPpv 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`, `ELINk`, `PARTition` `ORDEr`, `RIGId` and `JOINt` should be given prior to initiation of a problem solution using `BATCh` and/or `INTEractive`.

EPROp FEAPpv MESH INPUT COMMAND MANUAL

```
epro
   i-coor,xi-value,(pnum(i),i=1,ndf)
   <etc.,terminate with a blank record>
```

The proportional loading number to be appled to nodal forces and displacments 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.). The data to be supplied during the definition of the mesh consists of:

i-coor	– Direction of coordinate (i.e., 1 = x, 2 = y, etc.)
xi-value	– Value of i-direction coordinate to be used during search (a tolerance of 1/1000 of mesh size is used during search, any coordinate within the gap is assumed to have the specified value).
pnum(1,node)	– Proportional load number of dof-1
pnum(2,node)	– Proportional load number of dof-2
	etc., to *ndf* directions

For nodes with sloping conditions, the degrees-of-freedom are expressed with respect to the rotated frame 1-2 instead of the global frame x1-x2 (x-y). For three dimensional problem the 3-direction coincides with the x3-direction (z).

Proportional load numbers may also be specified using the `FPROp` and `CPROp` commands. The data is order dependent with data defined by `FPROp` processed first, `EPROp` processed second and the `CPROp` data processed last. The value defined last is used for any analysis.

```
ereg
   elem1,ngen1,reg1
   elem2,ngen2,reg2
   <etc.,terminate with blank record>
```

The EREGion command is used to specify the region number for elements. For each element to be specified a record is entered with the following information:

elem – Number of the element to be specified
ngen – Increment to the next element, if generation is used, otherwise 0.
reg – Region number to be assigned

When generation is performed, the element number sequence will be

$$elem1,\ elem1+ngen1,\ elem1+2\times ngen1,\ \ldots.\ ,\ elem2$$

The generated element are assigned to *reg1*.

Region numbers may also be assigned to element groups using the REGIon mesh command.

FORCe FEAPpv MESH INPUT COMMAND MANUAL

```
forc
   node1,ngen1,(f(i,node1),i=1,ndf)
   node2,ngen2,(f(i,node2),i=1,ndf)
   <etc.,terminate with blank record>
```

The FORCe command is used to specify the values for nodal boundary forces. For each node to be specified a record is entered with the following information:

node	– Number of node to be specified
ngen	– Increment to next node, if generation is used, otherwise 0.
f(1,node)	– Value of force for 1-dof
f(2,node)	– Value of force for 2-dof
	etc., to *ndf* directions

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

node1, node1+ngen1, node1+2×ngen1, ... , node2

The values for each force will be a linear interpolation between the *node1* and *node2* values for each degree-of-freedom.

While it is possible to specify both the force and the displacement applied to a node, only one can be active during a solution step. The determination of the active value is determined from the boundary restraint condition value. If the boundary restraint value is zero and you use one of the force-commands a force value is imposed, whereas, if the boundary restraint value is non-zero and you use one of the displacement-commands a displacement value is imposed. (See BOUNdary, CBOUndary, or EBOUndary pages for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value.

Force conditions may also be specified using the EFORce and CFORce commands. The data is order dependent with data defined by FORCe processed first, EFORce processed second and the CFORce data processed last. The value defined last is used for any analysis.

Example: FORCe

A concentrated force is to be applied to nodes 10 and 15. The force at node 10 has values of 100.0 in the horizontal direction and 0 in the vertical direction; whereas the force at node 15 has a magnitude of 200 and makes an angle of 60^o with the horizonatal 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 *FEAPpv* to compute the horizontal and vertical components.

FPROportional factors FEAPpv MESH INPUT COMMAND MANUAL

```
fpro
   node1,ng1,(pnum(i,node1),i=1,ndf)
   node2,ng2,(pnum(i,node2),i=1,ndf)
   <etc.,terminate with blank record>
```

The FPROportional factors command is used to specify the proportional load numbers for forced nodal conditions. For each node a record is entered with the following information:

node	– Number of node
ng	– Generator increment to next node
pnum(1,node)	– Proportional load number of dof-1
pnum(2,node)	– Proportional load number of dof-2
	etc., to *ndf* directions

The proportional load numbers are interpretted 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
   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.

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 vecto* 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 superceded by specifying a different condition during input of MATErial commands.

INCLude FEAPpv MESH INPUT COMMAND MANUAL

```
incl,filename
```

The INCLude command may be used to access data contained in a file called *filename*. This permits the data to be separated into groups which may be combined to form the problem data. Thus, if all the coordinate numerical data is in a file called COOR.DAT it may be combined into the mesh by using the command sequence:

```
COORdinates
  INCLude,COOR.DAT
         !blank terminator}
```

This is particularly useful when data is generated by another program.

Another use is for cases in which multiple executions are to be performed using a different value for some parameter. Placing the problem data in a file named Example.prb (without the definition for the parameter) and using the sequence:

```
PARAmeter
  n=2
         !blank terminator
INCLude,Example.prb
         !blank terminator
PARAmeter
  n=4
         !blank terminator
INCLude,Example.prb
         !blank terminator}
```

permits two executions for different values of the parameter n.

`loop,<xxxx>,n1`

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 input statements. The `LOOP` appears first, followed by one or more input statements 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,level\_3
  NEXT,level\_2
NEXT,level\_1
```

is permitted. If desired, the `xxxx` may be used (as above) to describe the type of next which is being closed, i.e., `NEXT,block` would indicate the end of a sequence of block generations.

MANUal FEAPpv 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.

```
mate,ma,<output label>
   type,iel,<id,(idf(i),i=1,ndf)>
   <parameters element type>
```

The MATErial set command is used to specify the parameters for each unique material set number *ma* in the analysis, as well as to specify the element type associated with the material set parameters.

The parameter *type* denotes the element formulation to be employed. *FEAPpv* 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 ELEMent or BLOCk commands. If *id* is less than or equal to zero it defaults to the value of the *ma* parameter. Material sets with the same *id* number are associated to each element which designate this *id* number, thus, an element can be associated with more than one material set.

The *idf* parameters are used to assign active degrees of freedom. Default: idf(i) = i, i=1,ndf.

The *MATE*rial command may also be used to provide a material identification label for the *FEAPpv* 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 *FEAPpv*. For `USER` elements the data is set by the programmer of each module.

NEXT FEAPpv MESH 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 input of a set of statements. 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,level\_3
  NEXT,level\_2
NEXT,level\_1
```

is permitted. If desired, the `xxxx` may be used (as above) to describe the type of next which is being closed, i.e., `NEXT,block` would indicate the end of a set of block inputs.

NOPArse FEAPpv MESH INPUT COMMAND MANUAL

```
nopa
```

The `NOPA`rse command may be used to enforce no parsing of the input data. *FEAPpv* 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.

NOPRint FEAPpv MESH INPUT COMMAND MANUAL

```
nopr
```

The use of the `NOPR`int command will discontinue placing information in the *FEAPpv* 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 PARAmeter command may be used to assign values to letter parameters. A letter parameter is defined immediately following the PARAmeter command (several may follow terminating with a blank record) according to the following:

```
x = expression
```

where x may be any of the single letters (*a*-*z*), any group of two letters (*aa*-*zz*), or any letter and a numeral (*a0*-*z9*) followed by the equal sign. The expression may be any set of numbers (floating point numbers should contain an E or a D exponent format so they will not be interpretted as integer constants!) or one or two letter constants together with any of the arithmetic operations +, -, *, /, or ^. The expression is processed left to right and can contain one set of parentheses to force groupings. Examples are:

```
a  = 3.
bb = 14/3.45
f  = a + 3.23/bb
c  = f + 1.03e-04*a/bb
d1 = (f + 1.03e-04)*a/bb
               ! blank terminator
```

In interactive mode of execution, the current set of paramater values may be output by entering *list* while in PARAameter input mode. After listing, input of additional parameters may be continued. It is possible to use expressions containing the parameters while in any input mode.

An input file may contain multiple PARAmeter commands. The values for parameters may be reset as needed. If an expression requires more than one set of parentheses a parameter may be used to temporarily hold the value for one set of parentheses and then reset. For example,

```
a = cos( (2*n-1)*p/l )
```

is not legal because of the nested parenthese, 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 FEAPpv MESH INPUT COMMAND MANUAL

```
pars
```

The `PARS`e command may be used to enforce parsing of the input data. *FEAPpv* 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.

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

The `POLAr` command may be used to convert any coordinates which have been specified in polar (or cylindrical) form, to cartesian coordinates. The conversion is performed using the following relations:

radius	= x(1,node)	– input value
theta	= x(2,node)	– input value in degrees
x(1,node)	= x_0 + radius × cos(theta)	
x(2,node)	= y_0 + radius × sin(theta)	
x(3,node)	= z_0 + x(3,node)	– 3-D only

The values for x_0, y_0, and z_0 are specified using the `SHIFt` command (default values are zero). A sequence of nodes may be converted by specifying non-zero values for *node1*, *node2*, and *inc*. The sequence generated will be:

node1, node1+inc, node1+2×inc, ... , node2

Several records may follow the `POLAr` command. Execution terminates with a blank record.

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

PRINt FEAPpv MESH INPUT COMMAND MANUAL

```
prin
```

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

REGIon FEAPpv MESH INPUT COMMAND MANUAL

```
regi,nreg
```

The `REGIon` command sets the current region number to *nreg*. The default value is 0. Regions may be used to separate parts of the mesh for which use of a `TIE` command is to connect. Alternatively, regions may be used during execution to `ACTIvate` or `DEACtivate` parts of the mesh during execution.

RESEt FEAPpv MESH INPUT COMMAND MANUAL

```
rese
```

Use of the RESEt 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 RESEt command, boundary conditions for specified *displacement* conditions may be reimposed using BOUNdary, EBOUdary, or CBOUndary commands.

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

The SHIFt command is used to specify the values for the origin of polar and spherical coordinate transformations (used by commands POLAr, SPHErical, or BLOCk). The input of *x0*, *y0*, and, for three dimensional problems, *z0* are in cartesian values based on the reference mesh coordinate distances.

SIDE FEAPpv MESH INPUT COMMAND MANUAL

```
side
   type1,(is(i,side1),i=1,nn)
   type2,(is(i,side2),i=1,nn)
   <etc.,terminate with blank record>
```

Currently, *FEAPpv* 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, volume 1 pp 181 ff. 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. 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 necessay to define the super-node locations using the mesh command `SNODe`. Finally, the mesh command `BLENd` must be specified for each mesh patch to be created.

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

The SNODe command is used to specify the values for nodal coordinates of *super nodes*. Currently, *FEAPpv* 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, volume 1 pp 181 ff. 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. Location of all super nodes may be graphically displayed using the PLOT,SNODe command.

In addition to the supernodes it may be necessay to define the sides of blend patches using the mesh command SIDE. Also, the mesh command BLENd must be given for each mesh patch to be created.

TEMPerature FEAPpv MESH INPUT COMMAND MANUAL

```
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):

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

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

TITLe FEAPpv MESH INPUT COMMAND MANUAL

```
titl,<on>
 titl,off
```

The `TITLe`,*off* command is used to suppress the print of headers on output pages produced by *FEAPpv*. 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 commmand 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 spurrious 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.

Appendix B

Mesh Manipulation Manual Pages

After the mesh is initially defined *FEAPpv* 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 summarze the commands which are available to manipulate the mesh data.

LINK FEAPpv 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 *FEAPpv* 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 interpretted 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* degree of freedoms

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 FEAPpv 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.

REMArk FEAPpv MESH MANIPULATION COMMAND MANUAL

```
remark,text
```

The REMArk command permits the writing of remarks to the output file. Remarks may only be written while in a mesh manipulation state or before the 'feappv' start record. They are inserted immediately into the output file.

```
tie
tie,line,n1
tie,node,n1,n2
tie,regi,n1,n2
tie,mate,n1,n2
tie,,dir,x-dir
```

A mesh may be generated by *FEAPpv* in which there is more than one node with the same coordinates. 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:

line	– [Currently not documented]
node	– Search node list between nodes *n1* and *n2*
regi	– Search regions *n1* and *n2* (*n1* can equal *n2*)
mate	– Search material identifiers *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 `BATCh` or `INTEractive` command for defining a solution algorithm, use of a `TIE` statement will cause the program to search for all coordinates that are to be connected together. 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`,,*i,value* (with i = 1,..,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 *region* or *material* parameters will result in searches based on these identifiers.

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.

Appendix C

Solution Command Manual Pages

FEAPpv 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 *FEAPpv*. These include optioins needed to solve most problems; however, provisions are also available for users to include their own solution routines through use of `UMACRn` subprograms. Methods to write and interface user routines to the program are described in the *FEAP* Programmers Manual.

BATCh/INTEractive FEAPpv COMMAND INPUT COMMAND MANUAL

```
batc
inte
xxxx,yyyy,v1,v2,v3
```

The solution algorithm used by *FEAPpv* to solve problems is defined by a *command language program*. The command language program may be executed in either a *batch* or an *interactive* mode using the initial command `BATCh` or `INTEractive`, respectively. By properly specifying the commands following either of these modes, a very wide range of applications may be addressed – including both linear and non-linear, as well as, steady state and transient applications.

The name for the command `xxxx` is selected from the list contained in the following pages of this appendix. The description for the options for `yyyy` and `v1`, `v2`, `v3` also may be obtained from the manual entry for each command.

```
acce,,n1,n2,n3
acce,coor,idir,xi
acce,all
```

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,coor,idir,xi
   ```

   prints all nodal quantities for the coordinate direction `idir`

   Example:

   ```
   acce,coor,1,3.5
   ```

   prints all the nodal accelerations 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:

   ```
   acce,all
   ```

   all nodal accelerations are output.

In order to output a acceleration vector it is first necessary to specify commands language instructions to compute the desired values, e.g., for accelerations perform a dynamic analysis.

`arcl,<xxxx,kfl,lfl>`

The `ARCL`ength command computes a solution using an arclength continuation method.

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 and retained to perform an arclength solution. To turn arclength off after it has been activated issue `ARCL,OFF`.

For the calculation of load deflection curves specify `PRO`Portional load using default parameters; the actual load level is computed by `ARCL`.

AUGMent FEAPpv 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 (e.g., incompressibility) during a solution.

```
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.

```
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. *FEAPpv* 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.

CHECk mesh FEAPpv COMMAND INPUT COMMAND MANUAL

```
chec
```

The `CHECk` command requests a check of the mesh consistency. It is necessary for elements to have checking capability for the `isw = 2` option in order for `CHECk` to report results. Typical tests include jacobian tests at nodes, tests on node sequencing, etc.

If the jacobian is negative at all nodes the nodal sequencing has been in put in reverse order and should be resequenced. The 4-node solid elements contained in *FEAPpv* 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.

DAMPing matrix FEAPpv COMMAND INPUT COMMAND MANUAL

```
damp
```

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 standard version of *FEAPpv* does not use the damping matrix. In the past user modified versions have included damping matrices it to compute the complex modes and frequencies of non-proportionally damped systems.

DATA FEAPpv 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.

```
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
```

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 requires the tangent matrix to fit within the blank common array dimensioned in the main program (see *FEAP* Programmer Manual for procedures to reset the size of the blank common array). 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.

An option exists to solve the equations for some problems using an iterative method (see, the `ITER`ative command language manual page).

DISPlacements FEAPpv COMMAND INPUT COMMAND MANUAL

```
disp,,<n1,n2,n3>
disp,coor,idir,xi
disp,all
disp,eigv,<n1,n2,n3>
```

The command `DISPlacement` may be used to print the current values of the solution 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,coor,idir,xi
   ```

   all nodal quantities for the coordinate direction `idir` with value equal to `xi` are output.

   Example:

   ```
   disp,coor,1,3.5
   ```

   prints all the nodal solution vector 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:

   ```
   disp,all
   ```

   all nodal solutions 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.

DT FEAPpv 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.

```
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 paratmeter. 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,all,n1,nn
```

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,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.

END FEAPpv 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.

EPRInt FEAPpv 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.

EXIT FEAPpv COMMAND INPUT COMMAND MANUAL

```
exit
```

The last interactive command must be `EXIT` or `QUIT` (they may also be abreviated 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.

FORM FEAPpv COMMAND INPUT COMMAND MANUAL

```
form
form,acce
form,expl
```

The `FORM` command computes the residual for the current time and iteration of a solution. *FEAPpv* 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 *FEAPpv* 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:

$$\boldsymbol{M}\,\boldsymbol{a} \;=\; \boldsymbol{R} \tag{C.1}$$

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 *FEAPpv* 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.

```
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 performfed 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 stiffnes 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 FEAPpv 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 FEAPpv 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 reexecute 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	Reexecute 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 FEAPpv COMMAND INPUT COMMAND MANUAL

```
iden,,<n1,n2>
```

The IDENtitiy 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.

```
init,disp
init,rate
```

Non-zero initial displacements or rates (e.g., velocities) for a dynamic solution may be specified using the `INITial` command. The values for any non-zero vector are specified after the `END` command for batch executions and may be generated in a manner similar to nodal generations in the mesh input. For interactive execution prompts are given for the corresponding data. Accordingly, the vectors are input as:

```
n1,ng1,v1-1, . . . ,v1-ndf
n2,ng2,v2-1, . . . ,v2-ndf
etc.
```

where, `n1` and `n2` define two nodes; `ng1` defines an increment to node `n1` to be used in generation; `v1-1`, `v2-1` define values for the first degree of freedom at nodes `n1`, `n2`, respectively; etc. for the remaining degree of freedoms. Generated values are linearly interpolated using the `v1` and `v2` values; etc. for the remaining degree of freedoms. Note that `ng2` is used for the next pair of generation records. If a value of `ng1` or `ng2` is zero or blank, no generation is performed between `n1` and `n2`.

ITERative FEAPpv COMMAND INPUT COMMAND MANUAL

```
iter
```

The ITERative command sets the mode of solution to iterative for the linear algebraic equations generated by a TANGent. Currently, iterative options exist only for symmetric, positive definite tangent arrays, consequently the use of the UTANgent command 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 *FEAP* programmer manual for procedures to reset the size of the blank common array).

The symmetric equations are solved by a preconditioned conjugate gradient method where the preconditioner is taken as the diagonal of the tangent matrix.

The iterative solution option currently available is 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 DIREct command language manual page).

`loop,<xxxx>,n1`

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. 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 matching `NEXT` command is input. In this way a set of statements may be grouped and executed together.

MANUal FEAPpv 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
mass,lump
```

The command `MASS` is used to compute a consistent or a diagonal *mass* matrix. Each element computes a contribution to both the consistent mass diagonal mass. The global mass to assemble is controlled by the parameter on the `MASS` command, with `LUMP` producing a diagonal mass and any option the consistent mass.

A 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.

MESH FEAPpv COMMAND INPUT COMMAND MANUAL

```
mesh
```

The use of the `MESH` command permits the redefinition 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.

NEWForce FEAPpv COMMAND INPUT COMMAND MANUAL

```
newf
newf,zero
```

The use of the NEWForce command will set a fixed pattern of nodal forces and displacements to the values of the current pattern in boundary force and displacements plus the previous "fixed" pattern. That is:

1. For degree-of-freedoms where forces (loads) are specified:

$$\texttt{f0(i,1)} = \texttt{f(i,1)} * \texttt{prop(t)} + \texttt{f0(i,1)} \tag{C.2}$$

2. For degree-of-freedoms where displacements are specified:

$$\texttt{f0(i,2)} = \texttt{u(i)} \tag{C.3}$$

   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 NEWForce had been used in generating the results.

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

NEXT FEAPpv 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 FEAPpv COMMAND INPUT COMMAND MANUAL

```
nopr
```

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

```
plot,quantity,[n1,n2,n3]
plot
```

In *FEAPpv*, 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 Appendix for admissible values of `quantity` and parameters.

```
prin
prin,on
prin,off
prin,comm
prin,data
prin,less
prin,<xxxx>
```

The use of the PRINt restores printing turned off by the NOPRint command or resets the level of printing to the screen. In interactive mode the use of PRINt,OFF eliminates all printing to the screen and the output file. PRINt,ON restores all printing.

Use of PRINt,LESS reduces the amount of command information displayed. Use of PRINt,COMMand restores command prints if they have been disabled by a PRINt,OFF or NOPRint,COMM.

The PRINt,DATA option restores printing of mesh data to the output file.

The default value is PRINt,ON.

The specification of:

```
xxxx = TANGent
xxxx = UTANgent
xxxx = CMASs
xxxx = LMASs
xxxx = RESIdual
```

will output the diagonal entries for the specified array. This may be useful in debugging elements, etc. The DEBUg option is also available.

PROPortional load FEAPpv COMMAND INPUT COMMAND MANUAL

```
prop,,<n1>
prop,,<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 loadings `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:

$$Prop(t) = a_1 + a_2\,(t - t_{min}) + a_3\,(\sin(a_4\,(t - t_{min}))^k \tag{C.4}$$

   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 loadings.

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.

QUIT FEAPpv 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).

```
reac,,<n1,n2,n3>
reac,coor,idir,xi
reac,all
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.

The `FILE` option outputs reactions to the restart save file with the extender `.ren` (starting from `re0`). These maybe used as input in Mesh (see Mesh `REAC`tion command).

READ FEAPpv COMMAND INPUT COMMAND MANUAL

```
read,xxxx
```

The READ command may be used to input the values of displacements and nodal stresses previously com- puted and saved using the WRITe command - it is primarily used for plots related to deformations or nodal stresses. It is not intended for a restart option (see RESTart) but may be used to restore displacement states of linear and non-linear elastic elements (or other elements with no data base requirements) for which reactions, stresses, etc. may then be computed.

The values of xxxx are used to specify the file name (4-characters only), manipulate the file, and read displacements and nodal stresses. The values permitted are:

```
xxxx = wind: Rewind current file.
xxxx = back: Backspace current file.
xxxx = clos: Close current file.
xxxx = disp: Read displacement state from current file.
xxxx = stre: Read nodal stress state from current file.
xxxx = Anything else will set current filename.
```

Only four characters are permitted and only one file may be opened at any time. Files may be opened and closed several times during any run to permit the use of more than one file name.

A READ input is created using the WRITe command which has identical options for xxxx except for the backspace option.

RESTart FEAPpv 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 fileset - the previous restart file is retained on the original fileset.

SAVE FEAPpv 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 exectution) 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.

SHOW FEAPpv COMMAND INPUT COMMAND MANUAL

```
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 arrays allocated together with their first word address, lengths, precisions, and remaining memory. All arrays are allocated out of a blank common whose length is assigned in the main routine `program feap`. The length also appears at the initiation of execution.

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 FEAPpv 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 *FEAPpv*, 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 depending on which solution scheme has been specified by the user.

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 FEAPpv 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 pro- jection 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 par- ticular 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.

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

The SUBSpace command requests the solution for n1 eigenpairs of a problem about the current state. An additional n2 vectors are used to expand the subspace and improve convergence (by default, n2 is set to the minimum of n1 plus 8 or 2 times n1 or the maximum number of eigenvalues in the problem). The SUBSpace command must be preceded by the specification of the tangent stiffness array using a TANGent command, and a mass array (either a lumped mass by MASS,LUMP or a consistent mass by MASS). Note that the smallest n1 eigenvalues and eigenvectors are computed with reference to the current shift specified on the TANGent command. If n2 is larger than the number of non-zero mass diagonals it is truncated to the actual number that exist. Whenever n1 is close to the number of non-zero mass diagonals one should compute the entire set since convergence will be attained in one iteration (this applies primarily to small problems).

Use of the PRINt option produces an output of all subspace matrices in addition to the estimates on the reciprocals of the *shifted* eigenvalues. For large problems considerable output results from a use of this option, and thus it is recommended for small problems only.

All eigenvalues are computed until two subsequent iterations produce values which are accurate to stol, (default stol = max(tol, 1.d-12)).

```
tang,,<n1,v2>
tang,line,<n1,v2,v3>
tang,eigv,,n1
```

The `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 FEAPpv 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 FEAPpv 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 maxiumum energy value). Normally, *FEAPpv* 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 ENERgy tolerance form can be set once whereas the EMAXimum form must be reset after each time command.

```
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
 arcl,node,dof
 stre,elmt,comp
 ener
 show
```

The `TPLOt` command can be used to specify components of displacement, velocity, acceleration, reaction, 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 20. An exception is for stress components where a maximum of 200 is permitted. The `inc` option is used to specify the number of time steps between saving of information. Each time the command tplot is given components are added to the list. The option show may be used to echo the current list to the screen during interactive executions.

Options which include both and `x,y,z` may be used in one of two ways. Giving the command as:

```
xxxx,node,dof
```

requires specific numbers to be provided for the `node` and `dof` parameters. The value of `node` is 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 `DISPlacement` option will save the node and degree of freedom value, together with the time in a file `Pxxx.dis`, where `xxx` is the name assigned for the input data

file (with the I stripped). 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.

```
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. A Generalized Newmark method `GN11` for first order ordinary differential equations such as heat transfer, etc.
3. A Generalized Newmark method `GN22` for second order ordinary differential equations such as vibration, etc.
4. An explicit implementation of Newmark.
5. An `SS11` method for first order ordinary differential equations such as heat transfer, etc.
6. An `SS22` method for

The `OFF` option turns off any active time integration algorithm returning *FEAPpv* 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. Generalized Newmark (GN11) where (`name` is `gn11`)

   The GN11 method requires one parameter for `v1`, be set. Permissible values are between 0 (explicit) and 1 (backward Euler). (Default is 1).

3. Generalized Newmark (GN22) where (`name` is `gn22`)

   The GN22 method requires two parameters for `v1` and `v2`, be set. Permissible values are between 0 (explicit) and 1 (backward Euler). (Default for both is 0.5 - which is equivalent to Newmark with $\beta = 0.25$ and $\gamma = 0.5$).

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

   $$\text{v2} = \quad \text{gamma (default} = 0.5)$$

5. Single Step (SS11) where (`name` is `ss11`)

   The SS11 method requires one parameter for `v1`, be set. Permissible values are between 0 (explicit) and 1 (backward Euler). (Default is 1).

6. Single Step (SS22) where (`name` is `ss22`)

   The SS22 method requires two parameters for `v1` and `v2`, be set. Permissible values are between 0 (explicit) and 1 (backward Euler). (Default for both is 0.5 - which is equivalent to Newmark with $\beta = 0.25$ and $\gamma = 0.5$).

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 UTANgent command computes an unsymmetric tangent stiffness matrix about the current value of the solution state vector. For linear applications the current stiffness matrix is just the normal stiffness matrix.

If the value of n1 is non-zero, a force vector for the current residual is also computed (this is identical to the FORM command computation) - thus leading to greater efficiency when both the tangent stiffness and a residual force vector are needed.

If the value of v2 is non-zero a *shift* is applied to the stiffness matrix in which the element mass matrix is multiplied by v2 and subtracted from the stiffness matrix. This option may not be used with the SUBSpace algorithm, which is restricted to symmetric tangents only (see TANGent. The shift may be used to represent a forced vibration solution in which all loads are assumed to be harmonic at a value of the square-root of v2 (rad/time-unit).

After the tangent matrix is computed, a triangular decomposition is available for subsequent solutions using FORM and SOLVe, etc.

In the solution of non-linear problems, using a full or modified Newton method, convergence from any starting point is not guaranteed. Two options exist within available commands to improve chances for convergence. One is to use a line search to prevent solutions from diverging rapidly. Specification of the command UTAN,LINE plus options invokes the line search option (it may also be used in conjunction with SOLVe,LINE in modified Newton schemes). The parameter v3 is typically chosen between 0.5 and 0.8 (default is 0.8).

The second option to improve convergence of non-linear problems is to reduce the size of the load step increments. The command BACK may be used to *back-up* to the beginning of the last time step (all data in the solution vectors is reset and the history data base for inelastic elements is restored to the initial state when the current time is started). Repeated use of the BACK command may be used. However, it applies only to the current time interval. The loads may then be adjusted and a new solution with smaller step sizes started.

The UTANgent operation is normally the most time consuming step in problem solutions - for large problems several seconds are required - be patient!

VELOcity FEAPpv COMMAND INPUT COMMAND MANUAL

```
velo,,<n1,n2,n3>
velo,coor,idir,xi
velo,all
```

The command `VELOcity` 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,COOR,idir,xi
   ```

   prints all nodal quantities for the coordinate direction `idir` with value equal to `xi`.

   Example:

   ```
   VELO,COOR,1,3.5}
   ```

   prints all the nodal velocity 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:

   ```
   VELO,ALL
   ```

   all nodal velocities are output.

In order to output a velocity vector it is first necessary to specify commands language instructions to compute the desired values, e.g., for velocities perform a dynamic analysis.

WRITe FEAPpv COMMAND INPUT COMMAND MANUAL

```
writ,xxxx
```

The `WRITe` command may be used to save the current values of displacements and nodal stresses for subsequent use. This option is particularly useful for saving states which are to be plotted later. It is not intended as a restart option (see `RESTart` for restarting a previously saved problem state).

The values of `xxxx` are used to specify the file name (4-characters only), manipulate the file, and write out states. The values permitted are:

xxxx =	wind	rewind current output file.
xxxx =	clos	close current output file.
xxxx =	disp	write current displacement state onto the current file.
xxxx =	stre	write current nodal stress state onto the current file.
xxxx =	????	anything else is used to set current filename. Only four characters are permitted and only one file may be opened at any time. Files may be opened and closed several times during any run to permit use of more than one file name.

A `WRITe` output is reinput using the `READ` command which has nearly identical options for `xxxx`.

Appendix D

Plot Manual Pages

FEAPpv 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.

```
acce,n1,n2,n3
```

Plot contours for acceleration degree of freedom `n1` (default is 1). Two options are available to construct contouring:

1. If `n2` is zero or negative, areas between contours will be shaded in colors or grayscale. For this case, only the minimum and maximum contour values are specified - by default (enter return) the program constructs 7 evenly spaced intervals for the shading.

   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 `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.

In interactive mode, after the `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.

AXIS FEAPpv 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.

BOUNdary conditions FEAPpv 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.

CARTesian FEAPpv 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) or an isometric view (see `ISO`metric manual page). (N.B. Problems of centering the isometric view may occur).

CENTer FEAPpv PLOT INPUT COMMAND MANUAL

```
cent,x,y
```

The `CENTer` command is used to place the center at a specific location on the screen. The input values of `x` and `y` locate the center of the plot in terms of normalized screen coordinates. The plot region covers approximately the area bounded by $0 <$ x , 1.4 and $0 <$ y < 1.0.

COLOr FEAPpv PLOT INPUT COMMAND MANUAL

```
colo,n1,n2
```

Sets PostScript outputs to color or grayscale. If `n1` < 0, grayscale plots are produced (by default PostScript plots are in grayscale). 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 FEAPpv 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 contouring:

1. If `n2` is zero or negative, areas between contours will be shaded in colors or grayscale. For this case, only the minimum and maximum contour values are specified - by default (enter return) the program constructs 7 evenly spaced intervals for the shading.

   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 `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.

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.

DEFAult FEAPpv PLOT INPUT COMMAND MANUAL

```
defa,on
defa,off
```

Normally, *FEAPpv* 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 FEAPpv 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 superpostion of undeformed or previous solutions on the current plot for comparison purposes.

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

DISPlacements FEAPpv PLOT INPUT COMMAND MANUAL

```
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 FEAPpv PLOT INPUT COMMAND MANUAL

```
dofs,n1,n2,n3
```

This command allows one to reorder or turn off the degree of freedoms for plotting purposes. `n1` becomes the first degree of freedom, `n2` becomes the second degree of freedom, and `n3` becomes the third degree of freedom. Entering a zero for any degree of freedom turns off that degree of freedom when plotting deformed shapes. By default, plotting is done with all degrees of freedom turned on and in their logical order (dofs,1,2,3).

```
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.

ELEMents FEAPpv PLOT INPUT COMMAND MANUAL

```
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 STREss, 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 CONTour 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 CONTour). Default: n3 = 0.

```
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,,n2
```

For the current material setting (see `MATE`rial, where the default is all material), each element face is filled in the color or gray scale. given by `n2`. If `n2` is zero the program selects appropriate colors for each material set.

```
fram,n1
```

This command defines a region in the screen plot window accordint to the followint options:

n1	Region used
0	Entire window used
1	Upper left quadrant
2	Upper right quadrant
3	Lower left quadrant
4	Lower right quadrant
	(Default: n1 = 0)

By using different frames, a large amount of information may be placed on a single screen. Each part of `FRAM`e may be cleared independently for some devices using a `WIPE,n1` command.

```
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.

```
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
```

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 loadings are given.

```
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
```

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,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 postion; if `n1` is zero numbers are placed near the postion of all nodes.

```
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.

```
pers,n1
   Requires:
      inew:
      vx,vy,vz
     ex,ey,ez
```

All subsequent plots are to be drawn in a three dimesional 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.

POSTscript FEAPpv PLOT INPUT COMMAND MANUAL

```
post,n1,n2
```

The `POST`Script command will enable the output of a postscript file for later use in producing hard copy plots. The sequence is initiated by the first `POST`Script command (a non-zero `n1` is in landscape mode, a zero value is in portrait mode). The name of the file containing the output is `feapX.eps` (where X is between `a` and `z`) and appears on the text screen. Subsequent commands will produce plots which appear on the screen and will also send information to the output file. A second `POST`Script command closes the output file and subsequent commands will give plots only on the screen.

Up to 26 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 FEAPpv execution session. Note that PostScript files can be quite large and disk quotas can easily be exceeded.

If `n2` is non-zero the *FEAPpv* logo is printed in the PostScript file by commands that cause it to be printed on the screen (e.g., `WIPE`). Default: n2 = 0.

```
prom,on
prom,off
```

Normally, *FEAPpv* will issue prompts for parameters needed to construct plots. Usually, default values may be accepted by pressing the return (or enter) key. The `PROMpt` command may be used to eliminate the need to press the key to accept the default values. The command has one parameter which is either `ON` or **OFF**. Omitting the parameter turns off the prompts.

```
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.

```
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.

SCALe FEAPpv 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.

```
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.

```
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 superpostion of deformed solutions on the current plot for comparison purposes.

A deformed option is specified using the `DEFO`rm command.

`velo,n1,n2,n3`

Plot contours for velocity degree of freedom `n1` (default is 1). Two options are available to construct contouring:

1. If `n2` is zero or negative, areas between contours will be shaded in colors or grayscale. For this case, only the minimum and maximum contour values are specified - by default (enter return) the program constructs 7 evenly spaced intervals for the shading.

   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 `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.

In interactive mode, after the `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.

```
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.