```
ang1
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. 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.
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.

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

The **BLOC**k data input segment is used to generate:

1. 2-node line elements in 1, 2, or 3-D.
2. 4 to 9-node quadrilateral elements in 2 or 3-D.
3. 3 or 6-node triangles in 2 or 3-D. For the 3-node elements alternative diagonal directions may be specified as indicated below.
4. 8-node bricks in 3-D.
5. Nodes only in 1, 2 or 3-D patches.

The patch of nodes and triangular or quadrilateral elements defined by **BLOC**k is developed from a master element which is defined by an isoparametric 4 to 9 node mapping function in terms of the two natural coordinates, **r** (or *xi*) and **s** (or *eta*), respectively. The node numbers on the master element of each patch defined by **BLOC**k are specified according to the figure below. 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 **BLOC**k. 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

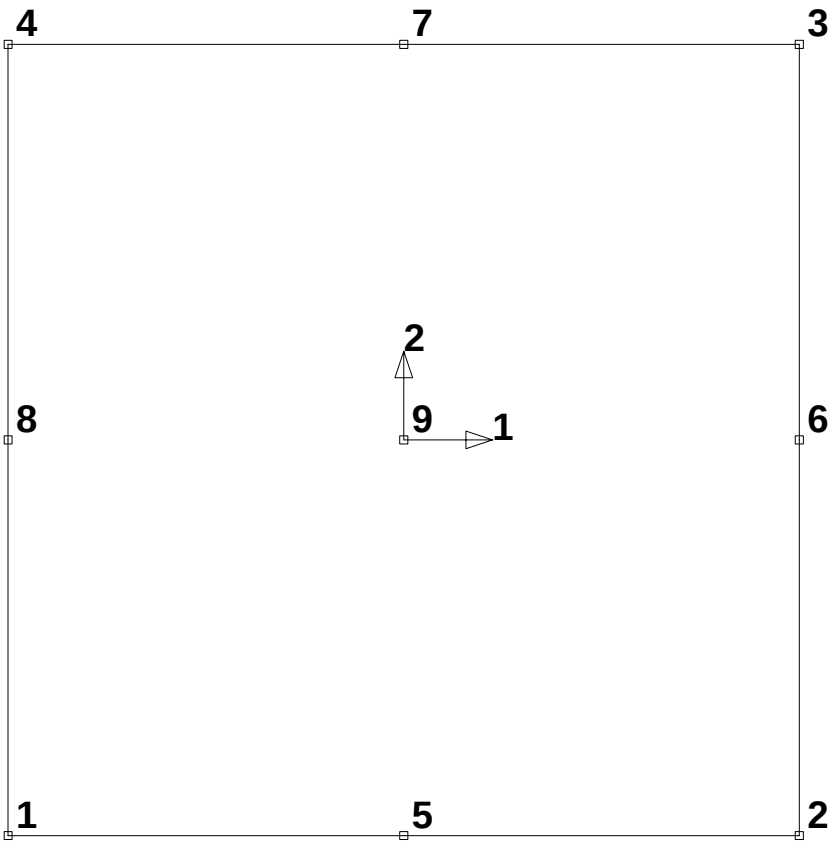
the corner nodes are required and mid-edge/side nodes are optional, as is the center node (node numbers are not described further here).

Patches of surfaces may be interconnected, in a restricted manner, by using the *r-skip* parameter judiciously. Alternatively, the **TIE** command may be used to merge adjacent patches.

The data parameters are defined as:

type - Master node coordinate type (**CART, POLA,** or **SPHE**).
r-inc - Number of nodal increments to be generated along r-direction of the patch.
s-inc - Number of nodal increments to be generated along s-direction of the patch.
t-inc - Number of nodal increments to be generated along t-direction of the patch (N.B. 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; if negative no elements are generated.
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).
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 bricks.

When using the **BLOC**k command one may enter zero for the total number of nodes and elements on the **FEAP** control record. **BLOC**k 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.



Node Specifications on 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* directions

The boundary restraint codes are interpretted as follows:

id(i,node) $= 0$	a force will be an applied load to dof.
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:

Degree-of-Freedoms

node	1	2	.	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.

```
cang
node,(x(i),i=1,ndm),angle

linear,angle-1,angle-2
x1,y1,x2,y2

quadratic,angle-1,angle-2,angle-3
x-1,y-1,x-2,y-2,x-3,y-3

cartesian

pola,x-0,y-0

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 to be supplied during the definition of the mesh consists of;

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	Angle for all nodes with value

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 1-2 instead of the global frame x1-x2. 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,angle
x-1,y-1,x-2,y-2
```

for quadratic segments the ends (nodes 1 and 2) together with an intermediate point are used. The quadratic segment is given as:

```
QUADratic,angle
x-1,y-1,x-2,y-2,x-3,y-3
```

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,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 **POLA**r option may be used to set the origin of a polar coordinate system. Coordinates entered after **POLA**r will be assumed to be radius and angle. The **CART**esian option resets the coordinate system to a cartesian frame.

```
cbou
node,(x(i),i=1,ndm),(ibc(j),j=1,ndf)

linear,(ibc(j),j=1,ndf)
x-1,y-1,x-2,y-2

quadratic,(ibc(j),j=1,ndf)
x-1,y-1,x-2,y-2,x-3,y-3

cartesian

polar,x-0,y-0

gap,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. After use files are deleted automatically.

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)
x-1,y-1,x-2,y-2
```

for quadratic segments the ends (nodes 1 and 2) together with an intermediate point are used. The quadratic segment is given as:

```
QUADratic,(ibc(i),i=1,ndf)
x-1,y-1,x-2,y-2,x-3,y-3
```

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,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 **pola**r option may be used to set the origin of a polar coordinate system. Coordinates entered after **pola**r will be assumed to be radius and angle. The **cart**esian option resets the coordinate system to a cartesian frame.

```
cdis

gap,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.

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 *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 a force value is imposed, whereas, if the boundary restraint value is non-zero a displacement value is imposed. (See **BOUN**dary, **CBOU**ndatry, 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.

```
cfor

gap,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.

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 *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 a force value is imposed, whereas, if the boundary restraint value is non-zero a displacement value is imposed. (See **BOUN**dary page for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value.

```
coor
node1,ngen1,(x(i,node1),i=1,ndm)
node2,ngen2,(x(i,node2),i=1,ndm)
<etc.,terminate with blank record>
```

The **COOR**dinate command is used to specify the values for nodal coordinates. For each node to be specified a record is entered with the following information:

node	- Number of node to be specified
ngen	- Increment to next node, if generation is used, otherwise 0.
x(1,node)	- Value of coordinate in 1-direction
x(2,node)	- Value of coordinate in 2-direction
	etc., to 'ndm' directions

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

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

The values generated or 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 **POLA**r or **SPHE**rical commands.

```
csur
linear,p-1,p-2
x-1,y-1,x-2,y-2

quadratic,p-1,p-2,p-3
x-1,y-1,x-2,y-2,x-3,y-3

disp,component

normal

tangential

polar,x-0,y-0

cartesian

gap,value

<terminate with a blank record>
```

A mesh may be generated in FEAP in which it is desired to specify distributed loading or displacements on parts of the body. For two dimensional problems it is possible to specify the surface to which the boundary condition is applied using the **CSUR**face command (The command is for Coordinate specified SURfaces.). The input values are saved in files and searched after the entire mesh is input (i.e., after the **END** mesh command. After use files are deleted.

The type of input to be generated is set using the **DISP**lacement, **NORM**al, or **TANG**ential options. These specify that inputs will be a specific displacement component, normal tractions (pressures), or tangential tractions (shears), respectively. The default is **NORM**al loadings. For displacement inputs the component to be generated is specified immediately after the **DISP**lacement command.

A two-dimensional surface may be specified as a **LINE**ar or a **QUAD**ratic line. For the linear surface the values at the ends are given together with the end coordinates. These are specified as:

```
LINEar,p-1,p-2
x-1,y-1,x-2,y-2
```

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,p-1,p-2,p-3
x-1,y-1,x-2,y-2,x-3,y-3
```

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,gap-value
```

The **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 **POLA**r option may be used to set the origin of a polar coordinate system. Coordinates entered after **POLA**r will be assumed to be radius and angle. The **CART**esian option resets the coordinate system to a cartesian frame. The default mode is **CART**esian.

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

The **DAMP**er command is used to specify the values for linear nodal dampers to earth. For each node on which non-zero values are to be specified a record is entered with the following information:

node	- Number of the node to be specified
ngen	- Increment to the next node, if generation is used, otherwise 0.
c(1,node)	- Value of the damper in 1-dof
c(2,node)	- Value of the damper in 2-dof
	etc., to *ndf* directions

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

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

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

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

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

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

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

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 a force value is imposed, whereas, if the boundary restraint value is non-zero a displacement value is imposed. (See `BOUN`dary page for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value.

```
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-idir	- Direction of coordinate (i.e., 1 = x, 2 = y, etc.)
x-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 x_1-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 x_1-x_2 (x-y). For three dimensional problem the 3-direction coincides with the x_3-direction (z).

WARNING: This command only affects the nodes that have their coordinates defined *before* issuing **EANG**le.

```
ebou
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-direction* coordinate (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-idir	- Direction of coordinate (i.e., 1 = x, 2 = y, etc.)
x-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.)

WARNING: This command only affects the nodes that have coordinates defined before issuing **EBOU**ndary.

```
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-direction* coordinate (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-idir	- Direction of coordinate (i.e., 1 = x, 2 = y, etc.)
x-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)	
d(2)	Value of displacement for dof's
...	
d(ndf)	

WARNING: This command only affects the nodes that have coordinates defined before issuing **EDIS**lacement.

```
efor
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-direction* coordinate (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-idir	- Direction of coordinate (i.e., 1 = x, 2 = y, etc.)
x-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)	
f(2)	Value of force for dof's
...	
f(ndf)	

WARNING: This command only affects the nodes that have coordinates defined before issuing **EFOR**ce.

```
elem
nelm1,ngen1,matl1,(ix(i,nelm1),i=1,nen)
nelm2,ngen2,matl2,(ix(i,nelm2),i=1,nen)
<etc.,terminate on blank record or when nelm2=nume1>
```

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 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*. Then 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 or when the element number reaches the value specified for *numel* on the control record.

ADVICE: When the number of elements on the control record is input as zero FEAP attempts to compute the number of elements in the mesh. The number computed is the largest number input by an **ELEM**ent input or during a **BLOC**k generation. Thus, during **ELEM**ent input it is necessary to input the last element in generation sequences.

```
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** macro command any additional data required to manipulate the mesh (e.g., **TIE**, **LINK**, **ELIN**k, **PART**ition, **OPTI**mize) should be given prior to initiation of a problem solution using **BATC**h and/or **INTE**ractive.

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

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

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

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

node1, node1+ngen1, node1+2×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 a force value is imposed, whereas, if the boundary restraint value is non-zero a displacement value is imposed. (See **BOUN**dary page for setting boundary conditions.). It is possible to change the type of boundary restraint during execution by resetting the boundary restraint value.

```
glob
plane stress
plane strain
axisymmetric
small
finite
temp,dof,value
```

The **GLOB**al command is used to set parameters which apply to all elements. Use of **PLAN**e **STRE**ss sets all 2-d elements to compute properties based on the plane stress assumption; use of **PLAN**e bf STRAin sets the properties for plane strain condition; and **AXIS**ymmetric sets the geometry to an axisymmetric condition.

The option **SMAL**l designates a small deformation solution option for all elements (this is the default mode); whereas, the option **FINI**te designates a finite deformation solution mode (at present only the two dimensional solid element supports this option - in displacement mode).

The **TEMP**erature **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 **THER**mal element type specified by the **MATE**rial set command).

Global parameters may be superceded by specifying a different condition during input of **MATE**rial commands.

```
incl,filename
```

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

```
coordinates
  include,COOR.DAT
                    !blank terminator
```

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

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

```
parameter
  n=2
                    !blank terminator
include,Example.prb
                    !blank terminator
parameter
  n=4
                    !blank terminator
include,Example.prb
                    !blank terminator
```

permits two executions for different values of the parameter n.

```
manu,level
```

The **MANU**al command will set the **level** of help commands shown when the command **HELP** is given in an interactive solution mode. The levels are: 0 = basic; 1 = intermediate; 2 = advanced; 3 = expert. The default level is 0.

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

The **MASS** command is used to specify the values for nodal point masses. For each node on which non-zero values are to be specified a record is entered with the following information:

node	- Number of the node to be specified
ngen	- Increment to the next node, if generation is used, otherwise 0.
m(1,node)	- Value of the mass in 1-dof
m(2,node)	- Value of the mass in 2-dof
	etc., to *ndf* directions

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

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

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

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

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

The parameter *type* denotes the element formulation to be employed. FEAP includes a library of elements for thermo-mechanical analyses. The included types are:

TRUSs	2-Node truss element (1, 2, or 3-D).
FRAMe	2-Node frame element (2 or 3-D).
PLATe	2-d Plate bending element.
SHELl	3-d Shell element.
SOLId	Continuum solid mechanics element (2 or 3-D).
THERmal	Continuum thermal element (2 or 3-D).

Users may also add their own elements and access by setting *type* to **USER** and the parameter *iel* to the number of the element module added (between 1 and 50).

The parameter *id* is the material identifier. Defined during element generation using **ELEM**ent or **BLOC**k commands. If *id* is less than or equal to zero it defaults to the value of the *ma* parameter. Material sets with the same *id* number are associated to each element which designate this *id* number, thus, an element can be associated with more than one material set.

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

The *MATE*rial command may also be used to provide a material identification label for the FEAP output file.

Example:

```
mate,1,Cam shaft material model:  Aluminum mechanical
solid,,1,1,2,3    !  properties for solid analysis
elastic,,200.0d09,0.3
                            !  terminate set 1
mate,,2,Cam shaft material model:  Aluminum thermal
thermal,,1,3,0,0  !  properties for thermal analysis
fourier,,50
                            !  terminate set 2
```

The *Cam shaft material model: Aluminum mechanical* will appear in the output file

before the first material parameter values printed from the element routine. Note, that two material sets have the same material identifier, consequently the element connection list belonging to this identifier will be processed twice - once for the mechanical and once for the thermal. For the mechanical element the local dofs 1, 2, and 3 will map to global dofs 1, 2, and 3; for the thermal element local dof 1 will map to global dof 3. The mechanical element will not form residual or tangent terms for the 3-dof; however, it is used to extract the temperature used to calculate the thermal strains. This temperature degree of freedom must be designated for the material set using a **TEMP**erature command (or globally, using the **GLOB**al,**TEMP**erature command).

The specific parameters to be input are described in the user manual for the elements included with FEAP. For **USER** elements the data is set by the programmer of each module.

```
nopr
```

The use of the **NOPR**int command will discontinue placing information in the FEAP output file of most subsequent mesh data (material data printed in each element will always be output). The use of **PRIN**t will cause the mesh information to again be reported in the output file. The default value is **PRIN**t at the start of each problem execution.

```
para
x = expression
```

The use of the **PARA**meter command may be used to assign values to letter parameters. A letter parameter is defined immediately following the **PARA**meter command (several may follow terminating with a blank record) according to the following:

$$x = expression$$

where x may be any of the single letters (a-z) and followed by the equal sign. The expression may be any set of numbers (floating point numbers must contain an E or a D exponent format so they will not be interpretted as letter constants!) or 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. An example is

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

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

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

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

is not legal because of the nested 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.

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

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

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

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

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

Several records may follow the **POLA**r command. Execution terminates with a blank record.

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

```
prin
```

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

```
regi,nreg
```

The **REGI**on command sets the current region number to *nreg*. The default value is 0. Regions may be used to separate parts of the mesh for which use of a **TIE** command is to connect. Alternatively, regions may be used during execution to **ACTI**vate or **DEAC**tivate parts of the mesh during execution.

```
rese
```

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

```
shif
```
x_0 , y_0 , z_0

The **SHIF**t command is used to specify the values for the origin of *polar* and *spherical* coordinate transformations (used by commands **POLA**r, **SPHE**rical, or **BLOC**k). The input of x_0, y_0, and, for three dimensional problems, z_0 are in cartesian values based on the reference mesh coordinate distances.

```
sphe
node1,node2,inc
<terminate with blank record>
```

The **SPHE**rical command may be used to convert any coordinates which have been specified in spherical form, to cartesian coordinates. The conversion is performed using the following relations:

radius	= x(1,node)	- input value
theta	= x(2,node)	- input value in degrees
phi	= x(3,node)	- input value in degrees
x(1,node)	$= x_0 +$ radius $\times$ cos(theta) $\times$ sin(phi)	
x(2,node)	$= y_0 +$ radius $\times$ sin(theta) $\times$ sin(phi)	
x(3,node)	$= z_0 +$ radius $\times$ cos(phi)	

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

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

Several records may follow the **SPHE**rical command. Execution terminates with a blank record.

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

The **STIF**fness command is used to specify the values for linear nodal stiffness (i.e., spring) to earth. For each node for which non-zero values are to be specified a record is entered with the following information:

node	- Number of the node to be specified
ngen	- Increment to the next node, if generation is used, otherwise 0.
k(1,node)	- Value of the stiffness in 1-dof
k(2,node)	- Value of the stiffness in 2-dof
	etc., to *ndf* directions

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

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

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

```
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 FEAP 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 **BATC**h or **INTE**ractive 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.

```
tran
T11, T12, T13
T21, T22, T23
T31, T32, T33
X0, Y0, Z0
```

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

$$\boldsymbol{X} = \boldsymbol{T}\,\boldsymbol{X}_{input} + \boldsymbol{X}_0$$

Thus the $\boldsymbol{X}$ correspond to the nodal values after applying the transformation and become the values used for the analysis.

```
acce,,n1,n2,n3
acce,coor,idir,xi
acce,list,n1
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` with value equal to `xi`.

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,list,n1
```

all nodal quantities contained in `list` number `n1` are output (see command **LIST** for specification of the list).

Example: `acce,list,3`

Prints the nodal accelerations contained in list number 3.

4.) 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.

`chec`

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

If the jacobian is negative at all nodes the nodal sequencing has been in put in reverse order and should be resequenced. The 4-node solid elements contained in *FEAP* will attempt the resequencing automatically; however, the error is not corrected in the data input file so that it is necessary to use the check command each time the problem is executed.

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

The command **DISP**lacement 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
```

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

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,list,n1
```

all nodal quantities contained in `list` number `n1` are output (see command **LIST** for specification of the list).

Example: `disp,list,3`

Prints the nodal solutions contained in list number 3.

4.) 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,,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.

```
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.

```
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
```

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,
form,acel
form,expl
```

The **FORM** command computes the residual for the current time and iteration of a solution. *FEAP* is a general nonlinear program and computes a residual for each solution by subtracting from any applied loads: (1) The force computed for the stresses in each element, often called the *stress divergence* or *internal force* term; (2) If the problem is dynamic the inertia forces.

At the end of each computation *FEAP* reports the value of the current residual in terms of its Euclidean norm, which is the square root of the sum of squares of each component of force.

If the **ACCEleration** option is present an acceleration is computed by solving the equation:

$$\boldsymbol{M}\ \boldsymbol{a}\ =\ \boldsymbol{R}$$

where $\boldsymbol{M}$ is a consistent mass or a lumped mass computed by the **MASS** command which must be available before the specification of the form command. This option may be used to compute consistent accelerations for starting a transient analysis using the Newmark type integration algorithms when initial forces or initial displacements are specified.

If the **EXPL**icit option is present *FEAP* computes a solution to the equations of motion (momentum equations) using an explicit solution option. Prior to using the **FORM,-EXPL**icit command it is necessary to form a diagonal mass using the **MASS,LUMP**ed command and 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.

```
help
```

The use of the **HELP** command will produce a list of the currently implemented commands for 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.

```
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.

```
init,disp
init,rate
init,spin,w1,w2,w3
```

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

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

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

When using the **SPIN** option, `w1,w2,w3` are the angular velocities of the body rotating about the origin. This initializes all active nodes.

```
loop,<xxxx>,n1
```

The **LOOP** command is used with the **NEXT** command to repeat the set of commands between them `n1` times. The four characters **xxxx** may be used to describe the action taken by the loop, e.g., loop,time,5 can indicate that the looping is for time steps and is to be performed for 5 steps. Alternatively, the **xxxx** may be left blank (i.e., 4 blank fields must be included before the next comma and the value for **n1**.

During interactive executions, **LOOP - NEXT** commands are not executed until the **NEXT** command is issued. In this way a set of statements may be grouped together and **BATC**h executed.

```
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
```

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.

```
next,<xxxx>
```

The **NEXT** command must be used in conjunction with a **LOOP** command. The **LOOP-NEXT** pair are 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. If desired, the **xxxx** may be used 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 issued. In this way a set of statements may be grouped and executed together.

```
nopr
```

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

```
para
letter=<value>
list
```

The use of the **PARA**meter command permits the input of data parameters during execution. These are normally used during the data input phase to vary the input values. For example, parameters may be set and used during proportional loading table inputs. Use of **LIST** will display the parameters and values for all letters set previously to non-zero values. Only 1-character parameters are permitted and should be lower case letters only.

```
  plot,quantity,[n1,n2,n3]
or
  plot
  quantity,[n1,n2,n3]
  ....
  end
```

In *FEAP*, screen and hard copy PostScript plots may be made for several quantities of interest. A **PLOT** may be specified to initiate interactive graphics outputs. After entering graphics mode a prompt will be displayed. At this time, **quantity** and the **n1**, **n2**, and **n3** values may be specified. Alternatively, a **PLOT,quantity,n1,n2,n3** command also may be issued while in interactive execution mode (this is the only option for batch executions). See the PLOT Manual for admissible values of **quantity** and parameters.

```
prin
prin,off
prin,on
prin,<xxxx>
```

The use of the **PRIN**t restores printing turned off by the **NOPR**int command. In interactive mode the use of **PRIN**t,**OFF** eliminates all printing between commands except as explicitly requested; **PRIN**t,**ON** reverses the action of **PRIN**t,**OFF**. The default value is bf PRINt and **PRIN**t,**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 **DEBU**g option is also available.

```
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 bf N1 to bf 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, exp, tmin, tmax, a(i),i=1,4

The proportional loading may be specified as:

Type 1 is defined by:

$$Prop(t) = a(1) + a(2) * (t - tmin) + a(3) * (\ sin(a(4) * (t - tmin)) ** \exp$$

for all time values between "tmin" and "tmax". The value of "exp" must be a positive integer all other parameters are real.

If a blank record is input the value of tmin is set to zero; tmax 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,,2 (or 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 'fpro' the sum of the records is used.

Type 2 is a table input. Up to 10 tables may be input. The input is as follows:

```
prop,,3,3  !  input proportional loading 3 only
2,nn  (default nn is 1)
t-1,   prop-1,   t-2,   prop-2,    ...  , t-nn,  prop-nn
t-nn+1,prop-nn+1,t-nn+2,prop-nn+2, ...  , t-2*nn,prop-2*nn
 (up to 100 data pairs may be input).
```

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, $prop_i$ and $prop_{i+1}$. This option is particularly useful for specifying cyclic loadings.

Example:

```
prop,,3,3
2,1
0.,0.
1.,1.
3.,-1.
5.,1.
7.,-1.
8.,0.
0.,0.
```

gives a cyclic loading with linear behavior between the times 0. and 8. and is zero thereafter.

```
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,list,n1
reac,file
```

Nodal reactions may be computed for all nodes in the problem and reported for nodes **n1** to **n2** at increments of **n3** (default increment = 1). If **n2** is not specified then only the values for node **n1** are output. When both **n1** and **n2** are not specified only total sum information is reported.

If the command is specified as:

```
reac,coor,idir,xi
```

prints all nodal reactions for the coordinate direction **idir** with value equal to **xi**.

Example:

```
reac,coor,1,3.5
```

will print all the nodal reactions which have x-1 = 3.5.

This is useful in finding the nodal values along a particular constant coordinate line.

All reactions may be output using the **REAC,ALL** command.

In addition to computing the reaction at each degree of freedom an equilibrium check is performed by summing the values for each degree of freedom over all nodes in the analysis. The sum of the absolute value of the reaction at each degree of freedom is also reported to indicate the accuracy to which equilibrium is attained. It should be noted that problems with rotational degrees of freedom or in curvilinear coordinates may not satisfy an equilibrium check of this type. For example, the sum for the radial direction in an axisymmetric analysis will not be zero due to the influence of the *hoop stresses*.

In addition to sums over all the nodes a sum is computed for only the nodes output. This permits the check of equilibrium on specified series of nodes, or the computation of the applied load on a set of nodes in which motions or restraints are specified.

If the command is specified as:

```
reac,list,n1
```

all nodal reactions contained in **list** number **n1** are output (see command **LIST** for specification of the list).

Example:

```
reac,list,3
```

will print all the nodal reactions which are in list number 3.

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

```
show
show,dict
show,elem
```

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** option (where *name* is the array name displayed using the **SHOW,DICT** command) outputs the current values of the array.

The **SHOW,ELEM**ent provides a one-line description of the currently loaded user elements.

```
solv,<line,v1>
```

The command **SOLV**e is used to specify when the equations generated by a **FORM** are to be solved. In *FEAP*, a direct solution of the equations is performed using a profile storage with a variable band (active column) method of solution or by an iterative method.

In the solution of some nonlinear problems it is possiible 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.

```
stre,,<n1,n2,n3>
stre,all
stre.<node,n1,n2,n3>
stre,coor,idir,xi
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 **SUBS**pace command requests the solution for **n1** eigenpairs of a problem about the current state. An additional **n2** vectors are used to expand the subspace and improve convergence (by default, **n2** is set to the minimum of **n1** plus 8 or 2 times **n1** or the maximum number of eigenvalues in the problem). The **SUBS**pace command must be preceded by the specification of the tangent stiffness array using a **TANG**ent command, and a mass array (either a lumped mass by **MASS,LUMP** or a consistent mass by **MASS**). Note that the smallest **n1** eigenvalues and eigenvectors are computed with reference to the current **shift** specified on the **TANG**ent command. If **n2** is larger than the number of non-zero mass diagonals it is truncated to the actual number that exist. Whenever **n1** is close to the number of non-zero mass diagonals one should compute the entire set since convergence will be attained in one iteration (this applies primarily to small problems).

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

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

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

The **TANG**ent command computes a symmetric tangent stiffness matrix about the current value of the solution state vector. For linear applications the current stiffness matrix is just the normal *stiffness* matrix.

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

```
TANGent,,1
```

is equivalent to the set of commands

```
TANGent
FORM
SOLVe
```

If the value of **v2** is non-zero a *shift* is applied to the stiffness matrix in which the element mass matrix is multiplied by **v2** and subtracted from the stiffness matrix. This option may be used with the **SUBS**pace command to compute the closest eigenvalues to the shift, **v2**. Alternatively, the shift may be used to represent a forced vibration solution in which all loads are assumed to be harmonic at a value of the square-root of **v2** (rad/time-unit).

After the tangent matrix is computed, a triangular decomposition is available for subsequent solutions using **FORM**, **SOLV**e, **BFGS**, etc.

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

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

The **EIGV**alue option is used in transient algorithms to compute eigenvalues of the (static) stiffness matrix. If **IDEN**tity has been issued, then the shift given by non-zero **n1** is with respect to the identity otherwise the element mass matrix is used. Note, **SUBS**pace is used to compute the actual eigen-pairs.

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

```
time,,<tmax>
time,<set,t>
```

The use of the **TIME** command will increment the current time by **DT**, the current time increment. In addition, a new value of the proportional loading will be computed, if necessary. The value of the current time and proportional loading are reported in the output (or to the screen). The time command also will perform the first update for an active time integration algorithm of the equations of motion (e.g., the Newmark-beta method) , as well as, update the history data base for any elements with non-linear constitutive equations (e.g., those which require variables other than the displacement state to compute a solution). Accordingly, it is imperative to include a time command for this class of problems. Example: Time dependent solution with loop control

```
dt,,1.
loop,,10
  time
   ..
  etc.
   ..
next
```

Performs 10 time steps of a solution.

As an option, it is possible to specify the maximum time that integration is to be performed. Accordingly, when a variable time step is employed the **TMAX** parameter value may be used as a convenient stop marker. This also is essential if an automatic time stepping algorithm is implemented. Example: Time dependent solution with loop control, terminate at specified time.

```
dt,,1.
loop,,10
  time,,5.0
   ..
  etc.
   ..
next
```

Performs 10 time steps of a solution; however, if the time reaches the value of 5.0 before the 10 steps terminate the execution. This may happen if the **DT** value is automatically adjusted by another step in the solution process.

The current time may be set to a specified value, **T**, using the command **TIME,-SET,T** (where **T** is the value desired). No other action is taken. This may be helpful in certain steady state problems where solutions are desired for certain specified times.

```
tol,,v1
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:

a.) 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).

b.) 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, *FEAP* performs nonlinear iterations until the value of the energy is less than the **TOL**erance value times the value of the energy from the first iteration. However, for some transient problems the value of the initial energy is approaching zero (e.g., for highly damped solutions which are converging to some steady state limit). In this case, it is useful to specify the energy for convergence relative to early time steps in the solution. Convergence will be assumed if either the normal convergence criteria or the one relative to the specified maximum energy is satisfied.

Finally, the tol command permits resetting the maximum energy value used for convergence. The command is issued as **TOL,EMAX**imum,**v1** where **v1** is the value of the maximum energy quantity. Since the **TIME** command sets the maximum energy to zero, the value of **EMAX**imum must be reset after each time step. Thus, a set of commands:

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

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

```
tplo
disp,node,dof
velo,node,dof
acce,node,dof
reac,node,dof
arcl,node,dof
stre,elmt,comp
ener
show
```

The **TPLO**t command can be used to specify components of displacement, velocity, acceleration, reaction, 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. stress, etc.) is limited to 20. 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.

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 stres option will save the element and component value, together with the time in a file Pxxx.str, 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. 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:

```
tplot
stre,3,24
stre,25,24
stre,25,26
disp,11,2
show
```

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. Finally, the list will be echoed bye 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. An Euler-backward difference method for first order ordinary differential equations such as heat transfer, etc.
3. Hilber-Hughes-Taylor alpha method for second order systems.
4. An explicit implementation of Newmark.
5. An energy conserving generalized mid-point method for second order systems.
6. A generalized mid-point method for static problems.
7. A generalized mid-point method for first order systems.

The **OFF** option turns off any active time integration algorithm returning *FEAP* to its default quasi-static solution mode.

The method used depends on the specified **NAME** in the command.

1. Newmark Method ('name' is 'newm' or blank)
   The values of the Newmark parameters are specified as follows:

```
v1  =  beta  - the Newmark parameter which primarily
               controls stability (default is 0.25).
v2  =  gamma - the Newmark parameter which primarily
               controls numerical damping ( default
               is 0.50)  Note:  gamma must be greater
               than or equal to 0.50.
```

   This option does not permit an "explicit" solution using beta = 0.0, only implicit solutions are considered. Accordingly, it is recommended that values of beta be set to 0.25 (the default value) unless there is a compelling reason not to use this value. With gamma set to 0.50 and beta set to 0.25 the method becomes the "average" acceleration or trapezoidal method.
2. Backward Euler ('name' is 'back')
   The backward Euler method requires no parameters for 'v1', etc., and may be used to solve any first order ode set. In this method only one rate vector exists, namely the rate of the solution vector.
3. HHT Alpha Method ('name' is 'alph')
   The HHT method requires the specification of three parameters. The first two are identical to the Newmark beta and gamma parameters, the third is the HHT alpha parameter (definition is different than original papers) in momentum equation:

$$\boldsymbol{M}\boldsymbol{a}(t_{n+1}) + \boldsymbol{N}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha})$$

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

```
v1  =  beta   (default = 0.25)
v2  =  gamma  (default = 0.5)
v3  =  alpha  (default = 0.5)
       Alpha should be between 0.5 and 1.  N.B. 1 = Newmark.
```

4. Explicit Newmark Method ('name' is 'expl')
   This option permits the explicit form of the Newmark method to be implemented. The input parameter is only
   ```
   v2  =  gamma  (default = 0.5)
   ```

5. Conserving Alpha Method ('name' is 'cons')
   The conserving method requires the specification of three parameters. The first two parameters are associated with the update formulas, the third with the momentum equa- tion,
   $$\boldsymbol{M}(\boldsymbol{v}(t_{n+1}) - \boldsymbol{v}(t_n))/\Delta t + \boldsymbol{N}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha})$$
   where $\boldsymbol{N}$ is the nonlinear internal force term.
   ```
   v1  =  beta   (default = 0.5)
   v2  =  gamma  (default = 1.0)
   v3  =  alpha  (default = 0.5)
          Alpha should be between 0.5 and 1.
   ```

6. Static Alpha Method ('name' is 'stat')
   The method requires the specification of the alpha parameter for the momentum equation
   $$\boldsymbol{N}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha})$$
   where $\boldsymbol{N}$ is the nonlinear internal force term.
   ```
   v3  =  alpha  (default = 0.5)
          Alpha should be between 0.5 and 1.
   ```

7. Alpha Method for First Order Systems('name' is 'gen1')
   The method requires the specification of the alpha parameter for the equation,
   $$\boldsymbol{M}\boldsymbol{v}(t_{n+\alpha}) + \boldsymbol{N}(\boldsymbol{x}(t_{n+\alpha})) = \boldsymbol{F}(t_{n+\alpha})$$
   where $\boldsymbol{N}$ is the nonlinear internal force term.
   ```
   v3  =  alpha  (default = 0.5)
          Alpha should be between 0.5 and 1.
   ```

   It is possible to specify nonzero values for the initial velocity in second order system integrators using the command **INIT**ial (for initial values). If the initial state is not in equilibrium an initial acceleration may be obtained by using a **FORM,ACCE** command before initiating any transient state. It is necessary for the parameters to first be set using a **TRAN**sient command. It is also possible to compute self equilibrating static states with non-zero displacements and then switch to a dynamic solution. Alternatively, a restart mode (**REST**art) may be used to start from a previously computed non-zero state.

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

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

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

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

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

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

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

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

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

all nodal quantities contained in `list` number `n1` are output (see command **LIST** for specification of the list).

Example: `velo,list,3`

Prints the nodal velocity contained in list number 3.

4.) 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.

`acce,n1,n2,n3`

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

a. 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.

b. 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,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 problem dimensions.

```
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.

`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 **ISOM**etric manual page). (N.B. Problems of centering the isometric view may occur).

```
cent,x,y
```

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

`cont,n1,n2,n3`

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

a. 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.

b. 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.

```
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.

```
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.

`dplo,<n1,n2>`

The **DPLO**t command may be used (in interactive mode only) to specify any line for a plot of an **n1** displacement component. After entering the command the LEFT mouse button is used to select the ends of a line through or in the mesh which defines the location for the displacement plot. For a PC (DOS mode) the i-j-k-l are used to move the cursor and the spacebar is used to select the ends of the line.

After entering the two ends for the line (labeled A and B), an X-Y plot for the **n1** displacement component is superposed on the screen. The X-axis of the plot corresponds to the selected A-B mesh line. The Y-axis of the plot displays the magnitude of projected displacement component along the line. The magnitude of displacement plotted is proportional to the largest and smallest values which occur anywhere in the mesh. A contour plot of the displacement component may be used to identify locations for the maximum and minimum (use the **CONT**our command). If **n2** is non-zero it is used as the plot number (up to 12 plots may be placed on the same figure). If **n2** is zero, the previous plot number is incremented and assigned as the current plot number.

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

```
eigv,n1,n2,n3
```

Plot information related to eigenvector **n1** (an eigensolution must be performed (see **SUBS**pace command in Appendix B). before attempting an eigenvector plot. The plot mode must also be set as **DEFO**rmed.

If **n3** is zero a deformed plot for the superposed eigenvector will be given.

If **n3** is nonzero contours for the **n3** degree of freedom for eigenvector **n1** will be constructed according to the value specified in **n2**.

For **n2** positive, **n2** contour values will be constructed. The values for each contour must be specified after the command language program for batch execution or at the prompt for interactive execution. Eight values per record are input. The number for the first contour is specified on the record (or prompt) immediately following the values.

For **n2** non-positive, a fill-type plot will be constructed. The maximum and minimum value of the quantity to be plotted must be given. The program will compute equally spaced intervals between these values for the plot. Alternatively a blank record may be input and the program will select values to be plotted based on maximum and minimum values of the component.

```
elem,n1
```

Plot numbers in or near the elements appearing in the visible plot region. If **n1** is non-zero plot number for specified element number only. After a **PICK**, **CLIP** or **ZOOM** some numbers may appear for elements surrounding the plot region even though no lines for element edges are shown.

```
estr,n1,n2,n3
```

This command functions exactly like **STRE**ss, except that the quantities plotted are done without inter-element smoothing.

The command plots contours of stresses (or other element variables), where **n1** is the component to be plotted and **n2** is the number of contours (same as for **CONT**our including shading options). The definitions of **n1** for 2 and 3 dimensional elasticity problems are:

```
n1   Component
 1   11-stress
 2   22-stress
 3   33-stress
 4   12-stress
 5   23-stress
 6   31-stress
 7-  User generated values
```

The n3 parameter is used for filled (solid color) stress plots as follows:

```
n3   Action
 0   superpose mesh on plot
 1   suppress showing mesh
-1   suppress showing mesh and uses previously set contour values
```

For contour line plots (**n2** > 0), a zero **n3** value will suppress numbers near each contour line (same as **CONT**our). Default: n3 $= 0$.

```
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,,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 **n2** is non-zero the all boundary facets are plotted. If **n3** is positive the color is set to n3.

```
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.

```
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.

```
material,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
```

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 number with the value of **n1** is placed near the node postion; if `n1` is zero numbers are placed near the postion of all nodes.

```
outl,n1
```

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 if **n1** is non-zero.

```
pers
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) or an isometric view (see **ISOM**etric 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

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.

```
pick
```

The **PICK** command may be used to select a portion of the plot region as a new plot region. The command works only with X11 and DOS modes and uses the mouse (or in DOS the i-j-k-l keys) for selection. Prompts are given to select two points from the screen with the left mouse button (or in DOS the spacebar); these two points are used to center a new square plot region. The command may be used repeatedly to identify successively smaller parts of the mesh as the plot region. The command **ZOOM** may be used to restore the full mesh to the plotting region. The command works in any plot mode (e.g., Cartesian, perspective) for 2 or 3 dimensional problems.

```
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 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 FEAP execution session. Note that PostScript files can be quite large and disk quotas can easily be exceeded.

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

```
parx,n1,n2,n3
```

This command plots principal stress axes for 2 and 3 dimensional problems. The parameters are interpreted as follows:

```
n1 = 0  all principal directions shown (or blank)
   = 1  maximum principal direction only (2 and 3-D)
   = 2  middle principal directions only (3-D)
        minimum principal direction only (2-D)
   = 3  minimum principal direction only (3-D only)

n2 = 0 (or blank) principal directions associated with
        both negative and positive principal values shown
   < 0  only principal directions associated with negative
        principal values shown
   > 0  only principal directions associated with positive
        principal values shown

n3      corresponds to different plotting colors <range 1-7>
```

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

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

```
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)
```

The **n3** parameter is used for filled (solid color) stress plots as follows:

```
n3   Action
 0   superpose mesh on plot
 1   suppress showing mesh
-1   suppress showing mesh and uses previously set contour values
```

For contour line plots (**n2** > 0), a zero **n3** value will suppress numbers near each contour line (same as **CONT**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.

```
show
```

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

```
splo,<n1,n2>
```

The **SPLO**t command may be used (in interactive mode only) to specify any line for a plot of an **n1** stress component. After entering the command the LEFT mouse button is used to select the ends of a line through or in the mesh which defines the location for the stress plot. For a PC (DOS mode) the i-j-k-l are used to move the cursor and the spacebar is used to select the ends of the line.

After entering the two ends for the line (labeled A and B), an X-Y plot for the **n1** stress component is superposed on the screen. The X-axis of the plot corresponds to the selected A-B mesh line. The Y-axis of the plot displays the magnitude of projected stress component along the line. The magnitude of stress plotted is proportional to the largest and smallest values which occur anywhere in the mesh. A contour plot of the stress component may be used to identify locations for the maximum and minimum (use the **STRE**ss command). If **n2** is non-zero it is used as the plot number (up to 12 plots may be placed on the same figure). If **n2** is zero, the previous plot number is incremented and assigned as the current plot number.

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

```
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-  User generated values
```

The **n3** parameter is used for filled (solid color) stress plots as follows:

```
n3   Action
 0   superpose mesh on plot
 1   suppress showing mesh
-1   suppress showing mesh and uses previously set contour values
```

For contour line plots (**n2** > 0), a zero **n3** value will suppress numbers near each contour line (same as **CONT**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:

a. 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.

b. 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.

```
wipe,n1
```

The frame is changed to **n1** and the region is cleared. The permissible values for **n1** are:

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

Some graphics terminals do not support the feature of erasing only part of the screen; in these cases the entire screen may be erased instead of only the part specified.

`zoom`

Restore current plot view to entire mesh. The limits for parts of the plot region to be displayed may be selected using the **PICK** view command.