NAME

felt − finite element problem description file format

DESCRIPTION

The *felt*(4fe) file format is used by the programs of the finite element package, *felt*(1fe), *velvet*(1fe), and *burlap*(1fe) to describe a finite element problem. The file is human readable and consists of a friendly, intuitive syntax rather than a table of numbers. Syntactic and semantic errors are detected and reported assuring that only valid problem descriptions are used. In general white space is unimportant, arbitrary numeric expressions may be used, and case of keywords is unimportant. As per standard convention, boldface items represent keywords, italicized items represent the syntax of the grammar, and items in brackets are optional. The file syntax is shown below.

[*problem-description*]
[*analysis-parameters*]
[*object-definitions*]
[*appearance-description*]
end

Problem description

The *problem-description* section specifies the problem title and the number of nodes and elements in the problem. If this section is not specified then the problem will be unnamed and is assumed to contain zero nodes and zero elements. This section may be absent for example in defaults files which define objects but do not specify an actual problem instance. The format for a *problem-description* is given below.

problem description
[**title** = *string*]
[**nodes** = *integer*]
[**elements** = *integer*]
[**analysis** = **static** | **transient** | **modal** | **static-thermal**| **transient-thermal**| **spectral**]

If the **title** is missing then the problem is unnamed. If **nodes** or **elements** is missing then none of that type of object are expected. The assignments can occur in any order and can even be repeated with the last assignment being used.

Analysis parameters

The *analysis-parameters* section defines any parameters for a specific type of analysis. Currently, this section is used only if the analysis mode is **transient**. The format of the *analysis-parameters* section is given below.

analysis parameters
[**alpha** = *expression*]
[**beta** = *expression*]
[**gamma** = *expression*]
[**step** = *expression*]
[**start** = *expression*]
[**stop** = *expression*]
[**Rm** = *expression*]
[**Rk** = *expression*]
[**nodes** = **[** *node-list* **]**]
[**dofs** = **[** *dof-list* **]**]
[**mass-mode** = **lumped** | **consistent**]

The **alpha**, **beta**, and**gamma** parameters are used in numerical integration schemes (transient and transient-thermal analysis). **start**, **stop**, and **step** define the range of time or frequency interest for transient or spectral analyses. In transient analyses, **start** is meaningless and **duration** and **dt** can be used as aliases for **stop** and **step**, respectively. **Rk** and **Rm** are global Rayleigh (stiffness and mass) damping proportionality constants. The *node-list* is a comma or white space separated list of node numbers that are of interest in the analysis. Similarly, the *dof-list* is a list of the degrees of freedom (**Tx**, **Ty**, **Tz**, **Rx**, **Ry**, and **Rz**) that are of interest.

An *object-definition* section defines objects of a specified type. Objects include nodes, elements, materials, constraints, forces, and distributed loads. Each of these types of objects is discussed below. Multiple *object-definition* sections are allowed and the sections may occur in any order.

Nodes

Nodes are points in cartesian space to which elements are attached. A node must have a constraint and may have an optional force. A node is identified by a natural number. The syntax is as follows:

nodes
node-definitions

where a *node-definition* takes the following form:

node-number
[**x** = *expression*]
[**y** = *expression*]
[**z** = *expression*]
[**constraint** = *constraint-name*]
[**force** = *force-name*]
[**mass** = *expression*]

The *node-number* starts the definition. Each node must have a unique number. If a cartesian coordinate is not given then the coordinate of the previous node is used. Similarly, if no constraint is given then the constraint applied to the previous node is used. As above, the assignments can appear in any order and any number of times. As indicated above, some objects are identified by their name and some by their number. Elements and nodes have numbers while materials, forces, loads, and constraints have names.

Elements

Elements are linear, planar, or solid objects which are attached to nodes. Each element must have a material and may have optional loads. Furthermore, each element has a type, or definition. Like nodes, elements are identified by a unique natural number. Elements of specific type are defined with the following syntax:

element-type **elements**
element-definition

where an *element-type* is one of the following:

spring
truss
beam
beam3d
CSTPlaneStrain
CSTPlaneStress
iso2d_PlaneStrain
iso2d_PlaneStress
quad_PlaneStrain
quad_PlaneStress
timoshenko
htk
brick
ctg
rod

and an *element-definition* has the following form:

element-number
nodes = [*node-list* **]**
[**material** = *material-name*]
[**load** = *load-name-list*]

The *element-number* starts the definition. Each element must have a unique number. If no material is given

then the material applied to the previous element is used. The *load-name-list* is a list of up to three loads to apply to the element. The *node-list* is a comma or white space separated list of node numbers. Each type of element requires a certain of nodes and in some cases a special "null node" which is numbered zero may be used to indicate a gap or filler in the list.

Materials

Elements are made of a type of material. Each material has a name and certain physical properties not all of which may be used by any one element. The syntax for defining materials is as follows:

material properties
material-definitions

where *material-definition* has the following form:

material-name
[**color** = *string*] # color for *velvet*
[**E** = *expression*] # Young's modulus
[**Ix** = *expression*] # moment of inertia about x-x axis
[**Iy** = *expression*] # moment of inertia about y-y axis
[**Iz** = *expression*] # moment of inertia about z-z axis
[**A** = *expression*] # cross-sectional area
[**J** = *expression*] # polar moment of inertia
[**G** = *expression*] # bulk (shear) modulus
[**t** = *expression*] # thickness
[**rho** = *expression*] # density
[**nu** = *expression*] # Poisson's ratio
[**kappa** = *expression*] # shear force correction
[**Rk** = *expression*] # Rayleigh damping coefficient (K)
[**Rm** = *expression*] # Rayleigh damping coefficient (M)
[**Kx** = *expression*] # thermal conductivity in the x-direction
[**Ky** = *expression*] # thermal conductivity in the y-direction
[**Ky** = *expression*] # thermal conductivity in the z-direction
[**c** = *expression*] # heat capacitance

The *material-name* starts the definition. If an attribute of a material is not specified then that attribute is zero. The assignments may occur in any order. The *color* specifies the color to use in drawing the material within *velvet*, and is ignored by other applications.

Constraints

Constraints are applied to nodes to indicate about which axes a node can move. The syntax for defining a constraint is as follows:

constraints
constraint-definitions

where *constraint-definition* has the following form:

constraint-name
[**color** = *string*] # color for *velvet*
[**tx** = **c** | **u** | *expression*] # boundary translation along x axis
[**ty** = **c** | **u** | *expression*] # boundary translation along y axis
[**tz** = **c** | **u** | *expression*] # boundary translation along z axis
[**rx** = **c** | **u** | *expression* | **h**] # boundary rotation about x axis
[**ry** = **c** | **u** | *expression* | **h**] # boundary rotation about y axis
[**rz** = **c** | **u** | *expression* | **h**] # boundary rotation about z axis
[**itx** = *expression*] # initial displacement along x axis
[**ity** = *expression*] # initial displacement along y axis
[**itz** = *expression*] # initial displacement along z axis
[**irx** = *expression*] # initial rotation about x axis

[**iry** = *expression*] # initial rotation about y axis
[**irz** = *expression*] # initial rotation about z axis
[**vx** = *expression*] # initial velocity along x axis
[**vy** = *expression*] # initial velocity along y axis
[**vz** = *expression*] # initial velocity along z axis
[**ax** = *expression*] # initial accel. along x axis
[**ay** = *expression*] # initial accel. along y axis
[**az** = *expression*] # initial accel. along z axis

The *constraint-name* starts the definition. A value of **c** for a boundary condition indicates that the axis is constrained; a value of **u** indicates that the axis is unconstrained. An expression indicates a displacement (non-zero) boundary condition and may contain the **t** variable for time varying boundary conditions in transient analysis problems. The initial dislacement, velocity and acceleration specifications are only used in transient problems. A value of **h** for a rotational boundary condition indicates a hinge. By default, all axes are unconstrained. The *color* specifies the color to use in drawing the constraint within *velvet*, and is ignored by other applications.

Forces

Forces, or point loads, may be applied to nodes. The syntax for a force definition is as follows:

forces
force-definitions

where a *force-definition* has the following form:

force-name
[**color** = *string*] # color for *velvet*
[**Fx** = *expression*] # force along x axis
[**Fy** = *expression*] # force along y axis
[**Fz** = *expression*] # force along z axis
[**Mx** = *expression*] # moment about x axis
[**My** = *expression*] # moment about y axis
[**Mz** = *expression*] # moment about z axis
[**Sfx** = *expression*] # frequency-domain spectra of force along x axis
[**Sfy** = *expression*] # frequency-domain spectra of force along y axis
[**Sfz** = *expression*] # frequency-domain spectra of force along z axis
[**Smx** = *expression*] # frequency-domain spectra of moment about x axis
[**Smy** = *expression*] # frequency-domain spectra of moment about y axis
[**Smz** = *expression*] # frequency-domain spectra of moment about z axis

The *force-name* starts the definition. If the force or moment is not specified then it is assumed to be zero. The *expressions* for forces may be time-varying. Time-varying expressions include the single variable **t** to represent the current time in the solution of a dynamic problem or consist of a list of discrete (time, value) pairs. Frequency varying expressions for spectra can also use **w** to represent the independent variable (radial frequency). The *color* specifies the color to use in drawing the force within *velvet*, and is ignored by other applications.

Loads

Distributed loads, or loads for short, are applied to elements. The syntax for a defining a distributed load is as follows:

distributed loads
load-definitions

where a *load-definition* has the following form:

load-name
[**color** = *string*] # color for *velvet*
[**direction** = *dir*] # direction

[**values** = *pair-list*] # local nodes and magnitudes

The *load-name* starts the definition. The *dir* is one of **LocalX**, **LocalY**, **LocalZ** (local coordinate system), **GlobalX**, **GlobalY**, **GlobalZ** (global coordinate system), **parallel**, or **perpendicular**. The *pair-list* is a sequence of *pairs*. A *pair* is a node number and an expression enclosed in parentheses. The node number refers to the position within the element rather than referring to an actual node. The *color* specifies the color to use in drawing the load within *velvet*, and is ignored by other applications.

Appearance Description

The *appearance-description* section is used by *velvet* to describe the appearance of a problem. This section is currently not used by *felt*. The appearance includes the state of the drawing area and any tool figures. This section consists of two subsections, the *canvas-configuration* section and the *figure-list* section. The *canvas-configuration* section has the following syntax.

canvas configuration
canvas-parameters

where a *canvas-parameter* has the following form:

[**node-numbers** = *boolean*] # node numbering
[**element-numbers** = *boolean*] # element numbering
[**snap** = *boolean*] # snap grid status
[**grid** = *boolean*] # visible grid status
[**snap-size** = *expression*] # snap grid size
[**grid-size** = *expression*] # visible grid size
[**node-color** = *color-name*] # node color
[**element-color** = *color-name*] # element color
[**label-font** = *font-name*] # labeling font
[**tool-color** = *color-name*] # tool figure color
[**tool-font** = *font-name*] # text figure font
[**x-min** = *expression*] # x-axis minimum
[**x-max** = *expression*] # x-axis maximum
[**y-min** = *expression*] # y-axis minimum
[**y-max** = *expression*] # y-axis maximum
[**x-pos** = *expression*] # x position of drawing area
[**y-pos** = *expression*] # y position of drawing area
[**width** = *expression*] # width of viewport window
[**height** = *expression*] # height of viewport window
[**scale** = *expression*] # scale of drawing area

A *boolean* is either **true** or **false**. A *color-name* is the name of a valid X11 color. Similarly, a *font-name* is the name of a valid X11 font. The last five parameters are probably not very meaningful to the user. The *figure-list* section has the following syntax.

figure list
figure-definitions

where a *figure-definition* has the following form:

figure-type
[**x** = *expression*] # x coordinate
[**y** = *expression*] # y coordinate
[**width** = *expression*] # width
[**height** = *expression*] # height
[**start** = *expression*] # starting angle
[**length** = *expression*] # arc length
[**text** = *name*] # text string
[**color** = *name*] # color
[**font** = *name*] # text font

[**points = [** *point-list* **]**] # line points

The *figure-type* starts the definition and is one of **rectangle**, **polyline**, **text**, or **arc**. Note that not all properties have meaning for all figures. Any unneeded property is ignored. If a color or font property is not given then the previous property is used. The *point-list* is a list of (x-coordinate, y-coordinate) pairs.

Expressions

An *expression* can be either constant or time-varying. As discussed above, time-varying expressions contain the variable **t** or consist of a list of discrete (time, value) pairs. If a time-varying expression is given where a constant expression is expected, the expression is evaluated at time zero. An *expression* has one of the following forms, where all operators have the precedences and associativities given to them in the C programming language.

expression **?** *expression* **:** *expression*	# in-line conditional
expression **||** *expression*	# logical or
expression **&&** *expression*	# logical and
expression **|** *expression*	# integer inclusive or
expression **^** *expression*	# integer exclusive or
expression **&** *expression*	# integer and
expression **==** *expression*	# equality
expression **!=** *expression*	# inequality
expression **<** *expression*	# less than
expression **>** *expression*	# greater than
expression **<=** *expression*	# less than or equal
expression **>=** *expression*	# greater than or equal
expression **<<** *expression*	# integer shift left
expression **>>** *expression*	# integer shift right
expression **+** *expression*	# addition
expression **-** *expression*	# subtraction
expression ***** *expression*	# multiplication
expression **/** *expression*	# division
expression **%** *expression*	# integer remainder
- *expression*	# arithmetic negation
! *expression*	# logical negation
~ *expression*	# integer bitwise negation
(*expression* **)**	# enforce precedence
sin (*expression* **)**	# sine
cos (*expression* **)**	# cosine
tan (*expression* **)**	# tangent
pow (*expression* **,** *expression* **)**	# power (exponentiation)
exp (*expression* **)**	# exponential
log (*expression* **)**	# natural logarithm
log10 (*expression* **)**	# base-10 logarithm
sqrt (*expression* **)**	# square root
hypot (*expression* **,** *expression* **)**	# Euclidean distance
floor (*expression* **)**	# floor
ceil (*expression* **)**	# ceiling
fmod (*expression* **,** *expression* **)**	# floating point remainder
fabs (*expression* **)**	# absolute value
number	# literal value
t	# current time

Finally, a discretely valued expression has the following syntax, where the optional **+** indicates that the list represents one cycle of an infinite waveform.

(*expression* **','** *expression* **)** ... [**+**]

AUTHOR

The *felt* file format was developed by Jason I. Gobat (jgobat@mit.edu) and Darren C. Atkinson (atkinson@ucsd.edu).

SEE ALSO

corduroy(1fe), felt(1fe), velvet(1fe), xfelt(1fe), corduroy(4fe).