Chapter 1

Finite element model components

1.1 Nodes

1.1.1 Description

The nodes of a finite element mesh are the points where the degrees of freedom reside. Each node object has, at least, the following information:

- Coordinates wich define its position in space. Typically (x,y,z) coordinates.
- Definition of the degrees of freedom in the node (displacements, rotations,...)

The nodes can also serve to define loads or masses that act over the model at its position.

1.1.2 Node creation

To create a node you can use the following commands:

```
nodos.newNodeXY(x,y)
nodos.newNodeIDXY(tag,x,y)
nodos.newNodeXYZ(x,y,z)
nodos.newNodeIDXYZ(x,y,z)
```

where:

nodos: is a node container obtained from the modeler.

tag: is an integer that identifies the node in the model.

(x,y) or (x,y,z): are the cartesian coordinates that define node's position.

1.2 Constraints

1.2.1 MP constraints

Description

An MP_Constraint represents a multiple point constraint in the domain. A multiple point constraint imposes a relationship between the displacement for certain dof at two nodes in the model, typically called the *retained* node and the *constrained* node:

$$U_c = C_{cr} U_r \tag{1.1}$$

An MP_Constraint is responsible for providing information on the relationship between the dof, this is in the form of a constraint matrix, C_{cr}, and two ID objects, *retainedID* and *constrainedID* indicating the dof's at the nodes represented by C_{cr}. For example, for the following constraint imposing a relationship between the displacements at node 1, the constrained node, with the displacements at node 2, the retainednode in a problem where the x,y,z components are identified as the 0,1,2 degrees-of-freedom:

$$u_{1,x} = 2u_{2,x} + u_{2,z} \tag{1.2}$$

$$u_{1,y} = 3u_{2,z} \tag{1.3}$$

the constraint matrix is:

$$C_{cr} = \left[\begin{array}{cc} 2 & 1 \\ 0 & 3 \end{array} \right] \tag{1.4}$$

and the vectors defining the dof's at the nodes are:

$$constrainedID = [0,1] \tag{1.5}$$

$$retainedID = [0,2] \tag{1.6}$$

Chapter 2

Solver components

2.1 Analysis components

2.1.1 Constraints

LagrangeMP_FE

LagrangeMP_FE is a subclass of FE_Element used to enforce a multi point constraint, of the form $U_c = C_{cr} U_r$, where U_c are the constrained degrees-of-freedom at the constrained node, U_r are the retained degrees-of-freedom at the retained node and C_{cr} a matrix defining the relationship between these degrees-of-freedom.

To enforce the constraint the following are added to the tangent and the residual:

$$\left[\begin{array}{cc} 0 & \alpha C^t \\ \alpha C & 0 \end{array} \right], \left\{ \begin{array}{c} 0 \\ 0 \end{array} \right\}$$

at the locations corresponding to the constrained degree-of-freedoms specified by the MP_Constraint, i.e. $[U_c \ U_r]$, and the lagrange multiplier degrees-of-freedom introduced by the LagrangeConstraintHandler for this constraint, $C = [-I \ C_{cr}]$. Nothing is added to the residual.

To construct a LagrangeMP_FE element to enforce the constraint specified by the MP_Constraint *theMP* using a default value for α of *alpha*. The FE_Element class constructor is called with the integers 3 and the two times the size of the *retainedID* plus the size of the *constrainedID* at the MP_Constraint *theMP* plus . A Matrix and a Vector object are created for adding the contributions to the tangent and the residual. The residual is zeroed. If the MP_Constraint is not time varying, then the contribution to the tangent is determined. Links are set to the retained and constrained nodes. The DOF_Group tag ID is set using the tag of the constrained Nodes DOF_Group, the tag of the retained Node Dof_group and the tag of the LagrangeDOF_Group, *theGroup*. A warning message is printed and the program is terminated if either not enough memory is available for the Matrices and Vector or the constrained and retained Nodes of their DOF_Groups do not exist.

virtual void setID(void);
Causes the LagrangeMP_FE to determine the mapping between it's equation numbers and the degrees-of-freedom. This information is obtained by using the mapping information at the DOF_Group objects associated with the constrained and retained nodes and the LagrangeDOF_Group, *theGroup*. Returns 0 if successful. Prints a warning message and returns a negative number if an error occurs: -2 if the Node has no associated DOF_Group, -3 if the constrained DOF specified is invalid for this Node (sets corresponding ID component to -1 so nothing is added to the tangent) and -4 if the ID in the DOF_Group is too small for the Node (again setting corresponding ID component to -1).

*virtual const Matrix &getTangent(Integrator *theIntegrator);*
If the MP_Constraint is time-varying, from the MP_Constraint *theMP* it obtains the current C_{cr} matrix; it then adds the contribution to the tangent matrix. Returns this tangent Matrix.
*virtual const Vector &getResidual(Integrator *theIntegrator);*
Returns the residual, a *zero* Vector.

Chapter 3

Materials

3.1 Stardard uniaxial materials

3.1.1 defElasticMaterial

Construct an elastic uniaxial material

```
defElasticMaterial(mdlr,name,E)
```

`mdlr`	modeler name
`name`	name identifying the material
`E`	tangent in the stress-strain diagram (see figure 3.1)

Example

*

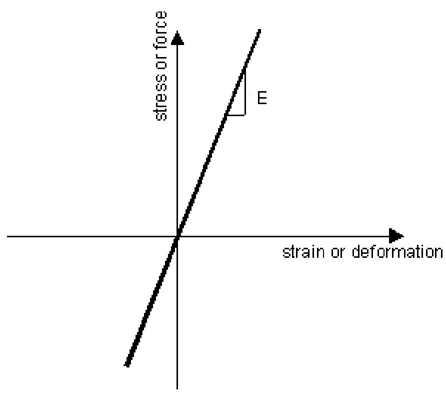

Figure 3.1: Elastic uniaxial material. Stress-strain diagram

3.1.2 defElasticPPMaterial

Construct an elastic perfectly-plastic uniaxial material

```
defElasticPPMaterial(mdlr,name,E,fyp,fyn)
```

`mdlr`	modeler name
`name`	name identifying the material
`E`	tangent in the elastic zone of the stress-strain diagram (see figure 3.2)
`fyp`	stress at which material reaches plastic state in tension (see figure 3.2)
`fyn`	stress at which material reaches plastic state in compression (see figure 3.2)

Example

*

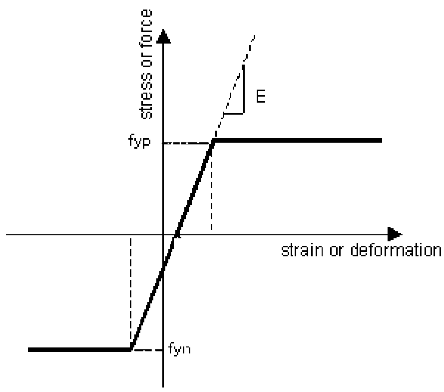

Figure 3.2: Elastic perfectly-plastic uniaxial material. Stress-strain diagram

3.1.3 defElastNoTracMaterial

Construct a uniaxial elastic-no tension material

```
defElastNoTracMaterial(mdlr,name,E)
```

`mdlr`	modeler name
`name`	name identifying the material
`E`	tangent in the elastic zone of the stress-strain diagram (see figure 3.3)

Example

*

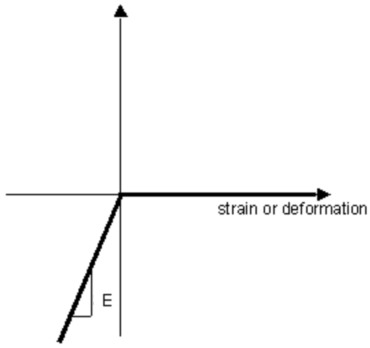

Figure 3.3: Elastic-no tension material. Stress-strain diagram

3.2 Steel and reinforcing steel materials

3.2.1 defCableMaterial

Construct a uniaxial bilinear prestressed material. The stress strain ranges from slack (large strain at zero stress) to taught (linear with modulus E).

`defCableMaterial(mdlr,name,E,prestress,rho)`

`mdlr`	modeler name
`name`	name identifying the material
`E`	Young modulus
`prestress`	prestress
`rho`	effective self weight (gravity component of weight per volume transverse to the cable)

Example

*

3.2.2 defSteel01

Construct a uniaxial bilinear steel material object with kinematic hardening

`defSteel01(mdlr,name,E,fy,b)`

`mdlr`	modeler name
`name`	name identifying the material
`E`	initial elastic tangent (see figure 3.4)
`fy`	yield strength (see figure 3.4)
`b`	strain-hardening ratio: ratio between post-yield tangent and initial elastic tangent (see figure 3.4)

Example

*

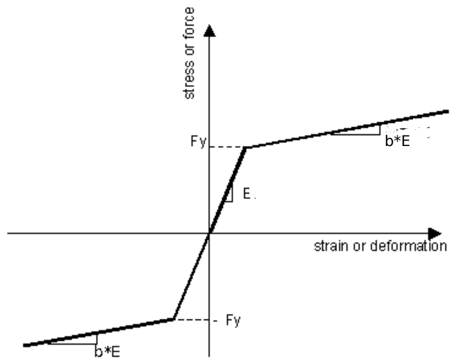

Figure 3.4: Steel001: uniaxial bilinear steel material with kinematic hardening. Stress-strain diagram

3.2.3 defSteel02

Construct a uniaxial Giuffre-Menegotto-Pinto steel material object with isotropic strain hardening

```
defSteel02(mdlr,name,E,fy,b,initialStress)
```

`mdlr`	modeler name
`name`	name identifying the material
`E`	initial elastic tangent (see figure 3.5)
`fy`	yield strength (see figure 3.5)
`b`	strain-hardening ratio: ratio between post-yield tangent and initial elastic tangent)
`initialStress`	initial stress

The transition from elastic to plastic branches (see figure 3.5) is controlled by parameters R0, R1, R2. The default values R0=15, R1=0.925 and R2=0.15

Example

*

3.3 Concrete materials

3.3.1 defConcrete01

Construct a uniaxial Kent-Scott-Park concrete material object with degraded linear unloading/reloading stiffness according to the work of Karsan-Jirsa and no tensile strength.

```
defConcrete01(mdlr,name,epsc0,fpc,fpcu,epscu)
```

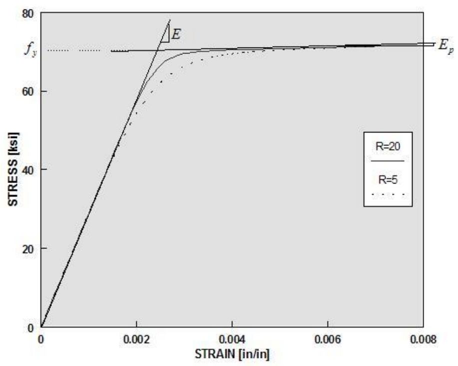

Figure 3.5: Steel002: uniaxial bilinear steel material with isotropic strain hardening. Stress-strain diagram

`mdlr`	modeler name
`name`	name identifying the material
`fpc`	concrete compressive strength at 28 days (compression is negative) (1)
`epsc0`	concrete strain at maximum strength (see figure 3.6) (2)
`fpcu`	concrete crushing strength (see figure 3.6)
`epscu`	concrete strain at crushing strength (see figure 3.6)

(1): Compressive concrete parameters should be input as negative values (if input as positive, they will be converted to negative internally)

(2): The initial slope for this model is $2 * fpc/epsc0$ (see figure 3.6)

Example

*

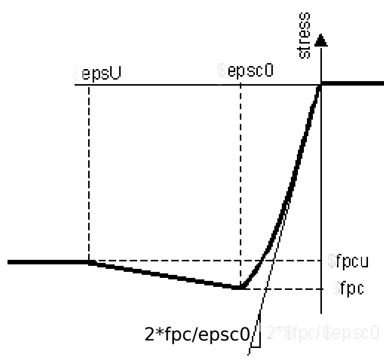

Figure 3.6: Concrete01: uniaxial Kent-Scott-Park concrete material. Stress-strain diagram

3.4 Sections

A section represents a force-deformation (or resultant stress-strain) relationship at beam-column and plate sample points.

3.4.1 defElasticSection2d

Construct an elastic section appropiate for 2D beam analysis.

```
defElasticSection2d(mdlr,name,A,E,I)
```

`mdlr`	modeler name
`name`	name identifying the section
`A`	cross-sectional area of the section
`E`	Young's modulus of material
`I`	second moment of area about the local z-axis

Example

*

3.4.2 defElasticShearSection2d

Construct an elastic section appropiate for 2D beam analysis, including shear deformations.

```
defElasticShearSection2d(mdlr,name,A,E,G,I,alpha)
```

`mdlr`	modeler name
`name`	name identifying the section
`A`	cross-sectional area of the section
`E`	Young's modulus of material
`G`	shear modulus
`I`	second moment of area about the local z-axis
`alpha`	shear shape factor

Example

*

3.4.3 defElasticSectionFromMechProp2d

Construct an elastic section appropiate for 2D beam analysis, taking mechanical properties of the section form a MechProp2d object.

```
defElasticSectionFromMechProp2d(mdlr,name,mechProp2d)
```

`mdlr`	modeler name
`name`	name identifying the section
`mechProp2d`	object that contains mechanical properties of the section

Example

*

3.4.4 defElasticSection3d

Construct an elastic section appropiate for 3D beam analysis.

`defElasticSection3d(mdlr,name,A,E,G,Iz,Iy,J)`

`mdlr`	modeler name
`name`	name identifying the section
`A`	cross-sectional area of the section
`E`	Young's modulus of material
`Iz`	second moment of area about the local z-axis
`Iy`	second moment of area about the local y-axis
`J`	torsional moment of inertia of the section

Example

*

3.4.5 defElasticShearSection3d

Construct an elastic section appropiate for 3D beam analysis, including shear deformations.

`defElasticShearSection3d(mdlr,name,A,E,G,Iz,Iy,J,alpha)`

`mdlr`	modeler name
`name`	name identifying the section
`A`	cross-sectional area of the section
`E`	Young's modulus of material
`G`	shear modulus
`Iz`	second moment of area about the local z-axis
`Iy`	second moment of area about the local y-axis
`J`	torsional moment of inertia of the section
`alpha`	shear shape factor

Example

*

3.4.6 defElasticSectionFromMechProp3d

Construct an elastic section appropiate for 3D beam analysis, taking mechanical properties of the section form a MechProp3d object.

`defElasticSectionFromMechProp3d(mdlr,name,mechProp3d)`

`mdlr`	modeler name
`name`	name identifying the section
`mechProp3d`	object that contains mechanical properties of the section

Example

*