Files

#include <~/material/section/SectionForceDeformation.h>

Class Declaration

class SectionForceDeformation : public Material

Class Hierarchy

TaggedObject
MovableObject
Material
SectionForceDeformation

Description

SectionForceDeformation provides the interface which all SectionForceDeformation models must implement.

Class Interface

// Constructor
SectionForceDeformation (int tag, int classTag);
// Destructor
virtual~SectionForceDeformation ();

// Public Methods
virtual int setTrialDeformation (const Vector &def) = 0;
virtual const Vector &getDeformation (void) = 0;
virtual const Vector &getResistingForce (void) = 0;
virtual const Vector &getPrevResistingForce (void) = 0;
virtual const Matrix &getTangentStiff (void) = 0;
virtual const Matrix &getPrevTangentStiff (void) = 0;
virtual const Matrix &getTangentFlex (void);
virtual const Matrix &getPrevTangentFlex (void);
virtual int commitState (void) = 0;
virtual int revertToLastCommit (void) = 0;
virtual int revertToStart (void) = 0;
*virtual SectionForceDeformation *getCopy (void) = 0;*
virtual const ID &getType (void) = 0;
virtual int getOrder (void) = 0;

// Public Methods for Output

virtual int sendSelf (int commitTag, Channel &theChannel) = 0;

virtual int recvSelf (int commitTag, Channel &theChannel, FEM_ObjectBroker &theBroker) = 0;

virtual void Print (ostream &s, int flag = 0) = 0;

Constructor

SectionForceDeformation (int tag, int classTag);

To construct a SectionForceDeformation whose unique integer among SectionForceDeformations in the domain is given by *tag*, and whose class identifier is given by *classTag*. These integers are passed to the Material class constructor.

Destructor

virtual ~SectionForceDeformation ();

Does nothing.

Public Methods

virtual int setTrialDeformation (const Vector &def) = 0;

Sets the value of the trial section deformation vector to be *def*. Returns 0 if successful, a negative number if not.

virtual const Vector &getDeformation (void) = 0;

Returns the section deformations at the current trial state.

virtual const Vector &getResistingForce (void) = 0;

Returns the section resisting forces at the current trial state.

virtual const Vector &getPrevResistingForce (void) = 0;

Returns the section resisting forces at the previous trial state.

virtual const Matrix &getTangentStiff (void) = 0;

Returns the section tangent stiffness matrix at the current trial state.

virtual const Matrix &getPrevTangentStiff (void) = 0;

Returns the section tangent stiffness matrix at the previous trial state.

virtual const Matrix &getTangentFlex (void);

Gets the current section tangent stiffness matrix and returns its inverse, the current section tangent flexibility matrix, via an explicit matrix inversion. NOTE: The explicit matrix inversion provides default behavior and may be overridden in subclasses to suit specific SectionForceDeformation implementations.

virtual const Matrix &getPrevTangentFlex (void);
Returns the section tangent flexibility matrix at the previous trial state. NOTE: This function provides default behavior and may be overridden in subclasses to suit specific SectionForceDeformation implementations.

virtual int commitState (void) = 0;
Commits the section state. Returns 0 is successful and a negative number if not.

virtual int revertToLastCommit (void) = 0;
Reverts the section to its last committed state. Returns 0 if successful and a negative number if not.

virtual int revertToStart (void) = 0;
Reverts the section to its initial state. Returns 0 if successful and a negative number if not.

*virtual SectionForceDeformation *getCopy (void) = 0;*
Returns a pointer to a new SectionForceDeformation, which is an exact copy of this instance. It is up to the caller to ensure that the destructor is invoked.

virtual const ID &getType (void) = 0;
Returns the section ID code that indicates the ordering and type of response quantities returned by the section. Lets the calling object (e.g. an Element) know how to interpret the quantites returned by this SectionForceDeformation model.

virtual int getOrder (void) = 0;
Returns the number of response quantities given by the section.

int sendSelf(int commitTag, Channel &theChannel);
Creates a Vector of size 5 into which it places *tag*, E, A, I, and $massDens$. In-

vokes *sendVector()* on *theChannel* using the ElasticSection2DObjects *dbTag*, the integer *commitTag* and the Vector as arguments. Returns 0 if successful, a warning message and a negative number are returned if the Channel object fails to send the Vector.

int recvSelf(int commitTag, Channel &theChannel, FEM_ObjectBroker &theBroker);
Creates a Vector of size 5. Invokes *recvVector()* on *theChannel* using the ElasticSection2DObjects *dbTag*, the integer *commitTag* and the Vector as arguments. Using the data in the Vector to set its *tag*, E, A, I, and $massDens$. Returns 0 if successful, a warning message is printed, *tag* and E are set to 0.0, and a negative number is returned if the Channel object fails to receive the Vector.

void Print(ostream &s, int flag =0);
Prints to the stream s the objects *tag* and E values.