Solution Process

Anthony Ricciardi

October 2021

# Cofe Class

The Cofe class is the primary user interface class for CoFE. The class constructor creates and solves the model; the user inputs either: (1) the path to a Nastran-format input file, or (2) a BdfEntries object. Class properties are Model objects and Solution objects.

## Model Class

Stores finite element model data and case control data. Contains methods for assembling and the system of equations and methods for results recovery. An array of Model classes is assembled if superelements are present.

## Solution Class

An abstract superclass – subclasses must be created for specific solution types (e.g., statics, modes). Provides solution methods. Contains results data.

# Input File Processing

The CoFE input file process creates CoFE Model objects from Nastran-formatted input files. The process uses a sequence of limited-scope classes.

## BdfLines Class

Reads Nastran-formatted input file lines. Handles INCLUDE statements. Removes comments. Partitions executive, case control, and bulk data sections. Partitions the bulk data section into part superelements.

## BdfFields Class

Constructed from a BdfLines object. Converts Nastran input lines to distinct entries and fields stored as char variables in cell/struct arrays. This is a general class with respect to the Case Control and Bulk Data sections; data are processed without regard to the specific case control commands and bulk data entries, so, no updates are required to handle new types of case control data or bulk data. All file management and executive control commands are ignored except for the for the first SOL entry, which is stored.

## BdfEntries Class

Created from BdfFields object and interfaces to Model objects. Can be exported as a Nastran-formatted file using the BdfEntries.echo() method. Specific CaseEntry and BulkEntry subclasses must be created for all input data types. The BdfEntries.entries2model() method creates a Model object from a BdfEntries object.

# Assembly Process

The initial Model properties mirror the data from the source BdfEntries object. The assembly process expands this data through the following methods.

## Model.preprocess() Method

Performs data sorting and checking. Full geometric processing of coordinate systems and nodes is carried out. Degree of freedom (DOF) sets are processed and checked. Certain data that require repeated access are processed and stored to speed up follow-on assembly.

* Parameter.preprocess() checks parameters and assigns default values if unspecified.
* CoordinateSystem.preprocess()sorts the coordinate systems data by ID number and checks ID uniqueness. Checks and assembles the full interdependent system of coordinate systems.
* Material.preprocess() sorts the materials data by ID number and checks ID uniqueness.
* Property.preprocess() sorts the properties data by ID number and checks ID uniqueness. Saves material data to property objects to save assembly time.
* Element.preprocess)sorts the element data by ID number and checks ID uniqueness.
* Point.preprocess)sorts the points (nodes and scalar points) data by ID number and checks ID uniqueness. Calculates and saves location of all nodes expressed in the basic coordinate system. Calculates and saves transformation matrices from the basic coordinate system to node deformation coordinate systems. Assigns global degrees of freedom for all nodes and scalar points.
* Point.getPerminantSinglePointConstraints() returns a logical array for the set permanently constrained DOF.
* Spcs.preprocess() returns a logical array for the sb [nGdof,1 logical] set – the set of constrained DOF specified for boundary conditions. The SID selected by SPC=SID in the first case control subcase is applied to all subcases. Different analyses must be run for different constraints.
* Mpcs.preprocess() returns a logical arrays for independent and dependent sets. Calculates constraint coefficient matrices (analogous to element matrices) for rigid elements (it’s more efficient to do this while defining sets than to postpone). Checks that MPCs don’t over constrain the model.
* DofSet.preprocess() provides a logical arrays for several DOF sets. Checks exclusivity of exclusive sets.
* SuperElement.preprocess() determines DOF indices of superelement connections.
* DofSet.partition() set logic and assignment related to superelements and Guyan/Dynamic model reduction.

Vector lists of ID numbers (e.g., a vector containing all element ID numbers) are concatenated and stored in the Model object so that they can be used later. This duplicates data, but is cost-effective because repeated concatenation is expensive.

## Model.assemble() Method

Calculates elastic element matrices. Assembles global matrices. Partitions global matrices. Optionally calculates reduced analysis-set matrices.

* Element.assemble() calculates element data (mass, stiffness, global-local transformations, recovery data) and assembles global mass and stiffness matrices.
* Mpcs.assemble() assembles the global multipoint constraint matrix from matrices calculated by Mpcs.preprocess().
* Load.assemble() assembles global load vectors and enforced-displacement vectors.
* Model.mpcPartition() partitions mass, stiffness, and load matrices for the independent set.
* Model.reducedModel.constructFromModel() calculates analysis set matrices and loads. Performs Guyan/dynamic reduction if requested.
* Superelement mass and stiffness matrices are superimposed on the residual structure.