Documentation - 'Truss' Class

November 1, 2019

This 'Truss' class is broken down into four methods: '$\_\_init\_\_$', 'PlotGeometry', '$\_\_calc_forces$', and '$\_\_repr\_\_$', which will be described subsequently.

1. **'**$\_\_init\_\_$**'** - This method takes in the names for the joint and beam files as '$file\_joint$' and '$file\_beams$', and loads the data into the desired Numpy format. It also separates the '$zero\_disp$' column into another Numpy array and converts it to integers. Finally, it calls the method '$\_\_calc\_forces$' to calculate the static equilibrium forces on the truss from the data.

2. **'**$PlotGeometry$**'** - This method creates a plot of the truss geometry and saves it as a .png file to the destination specified by '$file\_plot$'. This function is only called if there is a destination specified in the command line when main.py is called.

3. **'**$\_\_calc\_forces$**'** - This method computs the beam forces for the trust of interest. First, it find the number of joints and beams to initialize arrays, and then calculates the unit vectors for each beam. The next section finds all of the nonzero coefficients of the unknowns (B,...,R,...) in each equation (x and y for each beam), and adds them to lists 'cols', 'rows', and 'data', which are formed into a COO sparse matrix. In this section, the right hand side of the system of equations, containing the external forces on the joints, is also populated. The COO sparse matrix is then converted into the CSR format and is used by scipy.sparse.linalg.spsolve to find the values of the unknowns (stored as '$self\_\_.solved\_vec$. Here, if errors are thrown, the appropriate exception is called.

4. **'**$\_\_repr\_\_$**'** - This method formats and returns the desired output string, containing the computed beam forces (from the previous method), as the representation of an instance of the class 'Truss'.