The truss class is designed to take in two files, a (i) joint file that lists the number of joints, their x/y coordinates, any external forces, and an indicator for whether a hinge is fixed or not, as well as (ii) a beams file that lists the number of beams and the joints they connect. It then uses the method of joints to attempt to solve for the individual beam forces under the assumption that the truss is in static equilibrium, that is the sum of the forces acting at each joint is 0.

The methods of the class are as follows:

- *init* – Initializes an instance of the class, with filenames designating the locations of the joints, beams, and an optional plot file.
- *read_files* – Reads in the joints and beams files and stores them in dictionaries. Three dictionaries are outputted, one for the joints information, one for the beams, and one which combines the two in a convenient way for later use.
- *PlotGeometry* – Outputs a simple $(x, y)$ plot with connected line segments of the given truss geometry, saved to the location designated by the optional plot file location. The inputs are the joints and beams dictionaries outputted from read_files.
- *system_setup* – Uses the three aforementioned dictionaries and creates the three vectors (one for the nonzero elements in row order, one for the column indices, and one for the row storage) needed to store the sparse matrix in CRS style. The matrix is conceptually one which has two equations for each beam, and progresses row-wise in beam order. That is, the first two rows of the matrix correspond the x/y equations for beam 1, the second two for beam 2, etc. It depends on the following auxiliary method to actually compute the coefficients, as it turns out just the organizing of the matrix in CRS format is complicated:
- *get_coeff* – Computes the elements of the aforementioned vectors, by splitting into cases based on the component the equation is representing (eqn_ind parameter), and the relationship between the coordinates of the joint this equation corresponds to and the other joints connected to it by other beams (encapsulated in the parameters joint and coord).
- *matrixSolve* – This method converts inputted vectors associated with the CRS form into csr_matrix format and uses the scipy sparse solver to solve for the force vector. Errors are raised for singular matrices or over/underdetermined systems.
- *str* – String representation method which nicely prints the beams in beam order and the associated forces to console. It will also call the PlotGeometry method to save a plot of the truss geometry to the designated file location, if provided.