HW4-README

Paula Charles

October 2021

I constructed the **class Truss** to create a 2D truss and to analyze it and find out its stability and forces. It takes as inputs a joints file and a beams file that are all we need to create the truss system.

There is an optional 3rd parameter that provides the location of an output plot. If this location is precised, the __init__() method creates a plot that provides a 2D view of the truss and save it at that location. To do that, the __init__() method calls another method: **PlotGeometry()** which takes the location of the output plot as input and returns a file that is a plot of the 2D truss. This file is saved at the location asked.

Overall, the **__init__()** method creates self.beams which is an array containing the beams and the joints they bring together and self.joints which is a dictionary containing the joints and all their characteristics.

The matrix created to analyze the system has 2 rows per joint (one for the x equation, one for the y equation) and one column per beam and 2 additional columns per reaction force due to fixed support (one column for the x component, one for the y). We only need one column per beam force because we already know the direction of the beam force, so we only need to find out its norm. This is done in the **get_matrix()** method. The resulting matrix is a sparse matrix: a CSR matrix.
The method raises a Runtime error if the matrix is not square: the model will be overdetermined or underdetermined.
This method does not return anything and does not take anything as input but uses variables to create a new attribute: self.matrix.

The method **get_vector()** creates a vector with the external forces to complete the equations (given that they will be on the RHS of the equation, we have put a minus sign before them). It enables us to have a full set of equation, that we can now solve. This method does not take inputs and creates self.vector.

The method **solve_matrix()** uses the matrix and vector created and solves the linear equations system. It uses SciPy to do so. If the system is singular, it will raise a Runtime error to signal that it can't solve the linear system. It returns a vector of values that have been found for the unknowns in our equations.

Finally, the **__repr__** method creates a nice print. It calls the solve_matrix() to get the values of the beam forces and displays them in a pleasing way.