Write-up

Yu Gu

November 1, 2018

The basic problem of this project is to analyze the stability of a 2D truss and compute the beam forces. I design a python class Truss to deal with this task. Besides *__init__()* method to call the function of the whole process and *__repr__()* method to print out the result, this Truss class contains 5 methods as the decomposition of this task.

1. *solve(joints_file, beams_file, output_file)*:
   This method is to implement all steps to do truss analysis. The input arguments are the directory and filename of data and output figure. The output_file is optional and default = False.

2. *load_file(joints_file, beams_file)*:
   This method is to load the data of joints and beams from input file. The input arguments are the directory and filename of joints and beams data.

   - Use *np.loadtxt(filename, skiprows=1)* to load the data and skip the first row which is the name of the variables.
   - Generate a dictionary *xy* by extracting the coordinate of joints.
   - Store number of joints and beams in the class.

3. *PlotGeometry(output_file)*:
   This method is to plot the geometry of the truss. The input argument is the directory and filename of output figure. The idea is: while output_file is not False. For each beam, extract the coordinates of two joints, then plot beams as blue line and plot joints as red point.

4. *sparse_matrix()*:
   This method is to create sparse matrix of the linear system of equations. The idea of designing the linear system Ax=b is:

   We have 2 types of equations: (1) equilibrium of each joint in 2 directions; (2) geometric constraints for beam force of each beam. Hence, there should be (2*n_joints+n_beams) equations.

   For each equation, the variables are 2 types of unknown forces: (1) beams forces of each beam in 2 directions; (2) reaction forces of each fixed joint in 2 directions. Then, there should be (2*n_beams+2*n_support) variables.

   As for the known external force, we put it as b. Thus, the matrix A are of size(2*n_joints+n_beams, 2*n_beams+2*n_support). Comparing with setting directional beam forces in x and putting geometric constrains as coefficients into equilibrium equation, the matrix in our system has larger size, but the entries of A in the first two columns are just 1 or -1, this is much easier to generate the matrix.

   - The first 2*n_joints rows are the equilibrium equations:
     Row i and i+n_joints are equations for the ith joint in x and y directions.
   - The last n_beams rows are the geometric constraint equations:
     Row i is equation for the (i-2*n_joints)th beam.
   - The first 2*n_beams columns are coefficients of beam forces:
     Column j and j+n_beams are coefficients of jth beam in x and y directions.

- The last 2*n_support are coefficients of reaction forces:
  Column j and j+n_support are coefficients of (j-2*n_beams)th fixed joint in x and y directions.

Because each joint only belongs to a few beams, and each geometric constraint also stands for only one beam. So there are many 0 appears in A. Thus, we try to use sparse matrix to describe A. First, we generate the sparse matrix in COO format.

- Get entries for the submatrix A[0:2*n_joints, 0:2*n_beams]
  For each beam, the Bx, By coefficient of the first joint is 1, and -1 for the second joint.
  For each joint, related beams can be divided into 2 types by the position of the joint.
- Get entries for the submartix [2*n_joints:2*n_joints+b_beams, 0:2*n_beams]
  For each beam, the Bx, By coefficient of the first joint is -dy, and dx for the second joint.
- Get entries for the submatrix [0:2*n_joints, 2*n_beams:2*n_beams+2*n_support]
  For each fixed joint, the Rx, Ry coeffcient is 1.
- Finally, convert sparse matrix to CSR format for later computing.

5. *compute_force()*: This method is to compute the beam force by solving the linear system of equations. We will raise error messages in 2 cases:

   - The matrix is not square, which means this truss geometry not suitable for static equilibrium analysis.
   - The matrix is singular, which means the truss is unstable.

   After solving x by *sparse.linalg.spslove(A, b)*, we need to compute the directional beam forces. The sign of the beam forces are determined by the sign of dot product between (dx, dy) and (Bx, By).

By doing all these 5 methods above, we can get the result of truss analysis: whether the truss is stable, and the equilibrium beam forces while stable. Figure below shows the geometric of sample truss 2.

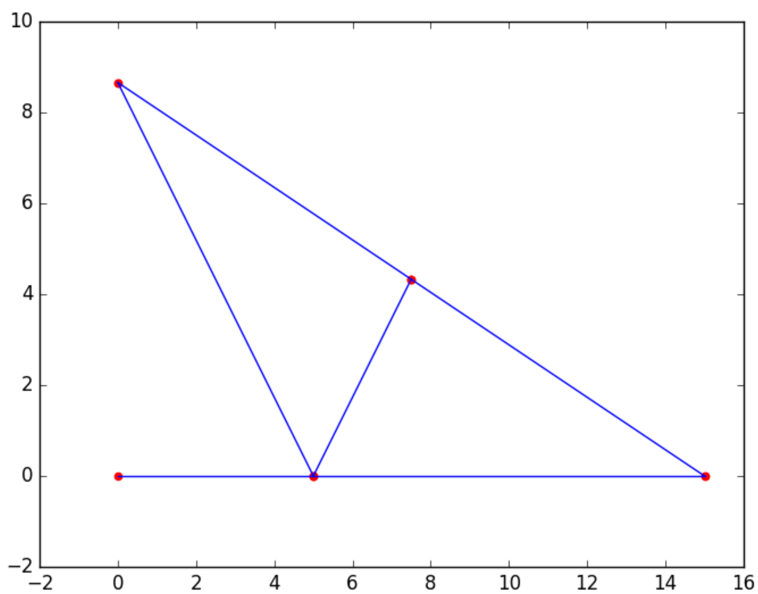

Figure 1: Geometry of Truss 2