README

Homework 4

Ines Dormoy
06679317

What I did

The purpose of the code is to determine if a truss is balanced or not. For that, we use the joints method to solve the equation for every joint.
The truss class first loads the data files and creates two data structures. We have a dictionary for the joints, and another one for the beams. The data structures are as follows:

- joints_dict={'1': 'coords': (x1, x2), 'F': (Fx, Fy), 'R': 0, (support reaction?), 'beams': ['1', '2'] (list of beams connected to the joint)}
- beams_dict={'1' (beam number): ('1', '2' (joints number))}

After loading the data, we calculate the coefficients of the A matrix to solve Ax = b. b contains the F forces. x contains the unknowns of the problem (B1, B1,... and R1, R1...), Ri is in the unknowns of the problem only if there is a reaction force for this joint. A contains the associated coefficients in function of the physics of the problem. There are lots of zeros in the A matrix, as lot of forces only apply for one or two specific points. Therefore, A is sparse and it's not interesting to store it as a dense matrix. We choose to save it as a CSR matrix, represented by three arrays. Those arrays are data, rows, and columns. Data stores the actual non zero data of the A matrix. Rowns and columns contain the rows and columns indicating where the values are stored. After creating A as a sparse matrix, we solve the system. We then output the x result in a nice way for the user.

The class is Truss, here are the methods

- __init__(self, joints_file, beams_file, output_file=None) :
  input: joints_file, beams_file - names of the joints and beams files structure chosen: 2 dictionaries joints_dict={'1': {'coords': (x1, x2), 'F': (Fx, Fy), 'R': 0, (support reaction?) 'beams': ['1', '2'] (list of beams connected to the joint)} beams_dict='1' (beam number): ('1', '2' (joints number))}
- load_files(self):
  loads the files *joints.dat* and *beams.dat*
  output: joints_dict and beams_dict described in $\_\_init\_\_$ and $reac = [0, 0, 1]$ means J1, J2 don't have reaction, J3 has a reaction.
- calc_coeff(self, pt1, pt2, case):
  computes the coefficient in the matrix A of the linear system input: pt1 (str), pt2 (str), case ('x' or 'y') pt1 and pt2 are the numbers of the joints pt1 is the point of the joint which equation is being considered case 'x' if we compute the coefficient on x axis equation, otherwise y
  output: coefficient of A (float)

- create_system(self):
  creates the A and b matrices to solve the Ax = b system A is a sparse matrix created with the data, rows and cols lists output: A and B matrices
- PlotGeometry(self):
  show the geometry of the truss
- solve_system(self, A, B):
  solves the system Ax=B, returns x
- __repr__(self):
  solves the system and returns the result