AME40541/60541: Finite Element Methods
Homework 3: Due Monday, February 22, 2021

Problem 1: (50 points) In this problem, you will implement the direct stiffness method in a series of steps. Before proceeding, carefully review the below code/comments as they provide crucial information regarding the specification of the truss topology, material properties, and boundary conditions, which will be needed for your implementation. For concreteness, the code below defines Truss 0 (Figure 1). Also carefully look through the starter code provided on the course website including the file `notation.m` as it defines the various terms you will encounter throughout the code.

```
% DEFINITIONS
% ————————————
% NDIM : Number of spatial dimensions
% NNODE : Number of nodes in mesh
% NELEM : Number of elements in mesh
% NDOF_PER_ELEM : Number of degrees of freedom per element
% NNODE_PER_ELEM : Number of nodes per element
% NDOF : Number of global degrees of freedom
% NDBC : Number of global degrees of freedom containing an essential BC

% XCG : Array (NDIM, NNODE) : The position of the nodes in the mesh.
%   The (i, j)—entry is the position of node j in the ith dimension. The
%   global node numbers are defined by the columns of this matrix, e.g.,
%   the node at XCG(:, j) is the jth node of the mesh.
xcg = [0.0, 1.0, 0.0, 1.0; ...
       0.0, 0.0, 1.0, 1.0];

% E2VCG : Array (NNODE_PER_ELEM, NELEM): The connectivity of the
%   mesh. The (:, e)—entries are the global node numbers of the nodes
%   that comprise element e. The local node numbers of each element are
%   defined by the columns of this matrix, e.g., E2VCG(i, e) is the
%   global node number of the ith local node of element e.
e2vcg = [1, 1, 2, 3, 1; ...
         2, 3, 4, 4, 4];

% EA : Array (NELEM,) : Young's modulus times cross—sectional area for
%   each element.
EA = [1.0; 2.0; 3.0; 4.0; 5.0];

% DBC_IDX : Array (NDBC,) : Indices into array defined over global dofs
%   (size = NDIM*NNODE) that indicates those with prescribed
%   primary variables (essential BCs).
dbc_idx = [1; 2; 4];

% DBC_VAL : Array (NDBC,) : Value of the prescribed primary variables such
%   that U(DBC_IDX) = DBC_VAL, where U is a (NDIM*NNODE,) vector
%   that contains the primary variable (all dofs of all nodes).
dbc_val = [0.0; 0.0; 0.0];

% FBC_VAL : Array (NDOF—NDBC,) : Value of the prescribed forces at all
%   global dofs without a prescribed displacement (NFBC = NDIM*NNODE—NDBC).
%   Let FBC_IDX = setdiff(1:NDIM*NNODE, DBC_IDX), then F(FBC_IDX) = FBC_VAL
fbc_val = [0.0; 0.0; 0.0; 0.1; 0.0];
```

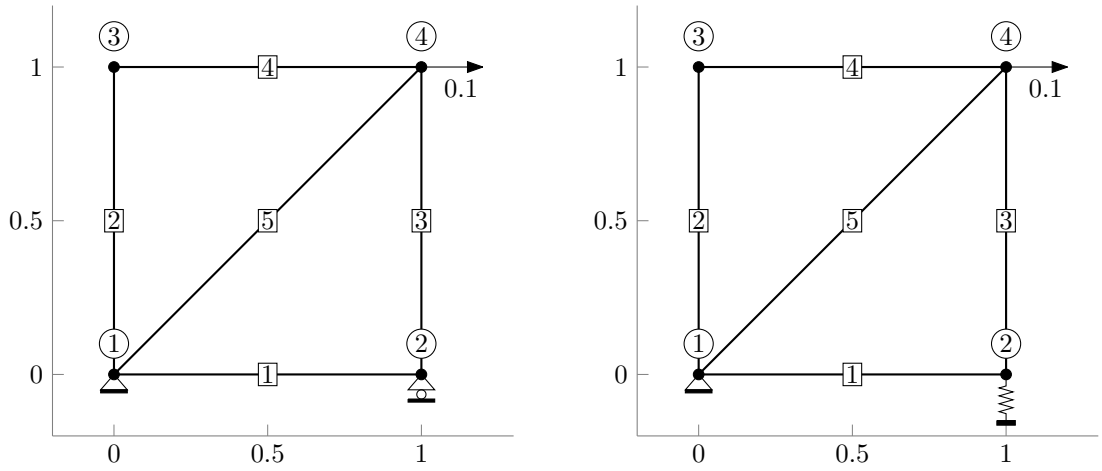

Figure 1: Truss 0 (*left*) and Truss 1 (*right*)

Problem 1.1 Implement a function `create_transf_data_truss.m` that creates a MATLAB structure defining the quantities needed to map to/from coordinate system aligned with element. Starter code is provided on the course website in the Homework 3 code distribution. Be sure to *test your code*.

Problem 1.2 Implement a function `intg_elem_stiff_truss.m` that evaluates the stiffness matrix for a truss element. Starter code is provided on the course website in the Homework 3 code distribution. Be sure to *test your code*.

Problem 1.3 Implement a function `eval_unassembled_stiff_truss.m` that evaluates and stores the element stiffness matrix for each member in the truss. Starter code is provided on the course website in the Homework 3 code distribution. Be sure to *test your code*.

Problem 1.4 Implement a function `create_ldof2gdof_cg.m` that creates a matrix that maps local degrees of freedom for each element to global degrees of freedom (ignoring boundary conditions). Starter code is provided on the course website in the Homework 3 code distribution. Be sure to *test your code*.

Problem 1.5 Implement a function `assemble_nobc_mat_dense.m` that assembles the element stiffness matrices into the global stiffness matrix without applying Dirichlet boundary conditions. Starter code is provided on the course website in the Homework 3 code distribution. Be sure to *test your code*.

Problem 1.6 Implement a function `apply_bc_solve_dsm.m` that applies boundary conditions via static condensation to the global stiffness matrix and solves for the unknown displacements and reaction forces. Starter code is provided on the course website in the Homework 3 code distribution. Be sure to *test your code*.

Problem 1.7 Implement a function `solve_dsm_truss.m` that uses the direct stiffness method to solve for the nodal displacements and reaction forces of a truss structure using the functions created in Problems 1.1-1.6. Starter code is provided on the course website in the Homework 3 code distribution. Be sure to *test your code*.

Problem 2: (10 points) Use the functions written in Problem 1 to solve for the nodal displacements and reaction forces of Truss 0 (Figure 1). The Young's modulus times the cross-sectional area of each element are: $EA_e = e$ for $e = 1, \ldots, 5$. Report the displacements and forces at each node and plot the deformed truss using the function `visualize_truss` provided on the course website. The setup function `setup_truss0` is provided for you in the starter code.

Problem 3: (10 points) Use the functions written in Problem 1 to solve for the nodal displacements and reaction forces of Truss 2 (Figure 2). Report the displacements and forces at each node and plot the deformed truss using the function `visualize_truss` provided on the course website. The Young's modulus times the cross-sectional area of each element are: $EA_e = e$ for $e = 1, \ldots, 8$. This requires implementing a new function to replace `setup_truss0` that defines the topology, material properties, and boundary conditions of this truss and then passing the resulting variables to your function `solve_dsm_truss`.

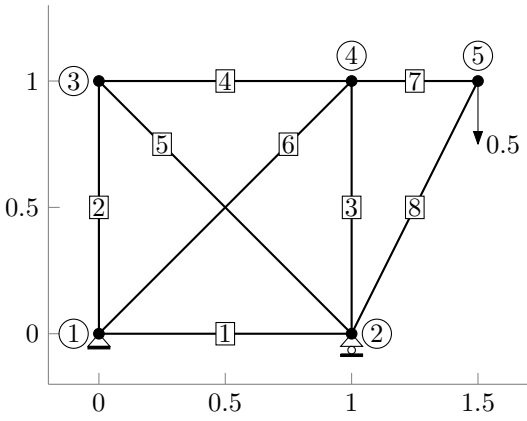

Figure 2: Truss 2

Problem 4: (10 points) Use the functions written in Problem 1 to solve for the nodal displacements and reaction forces of the Warren truss (Figure 3). Report the displacements of the node at the top right of the truss (node with the horizontal external force) and forces on the node at the bottom left of the truss (pinned node). The nodal coordinates, connectivity, boundary conditions, and load are defined in the function `setup_warren_truss` that can be found in the Homework 3 code distribution on the course website.

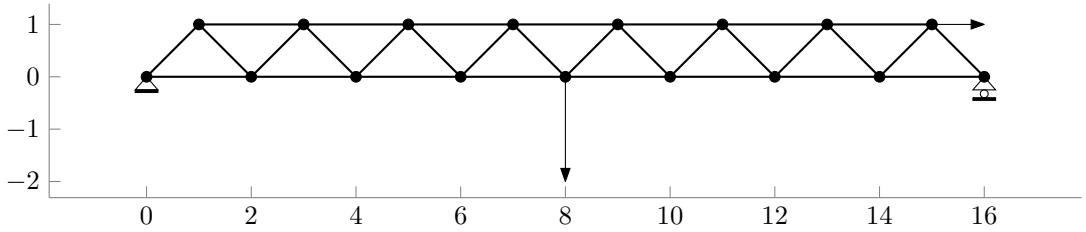

Figure 3: Warren truss

Problem 5: (20 points) (AME60541 only) Add support in your direct stiffness method code for elastic boundary conditions. Implement a function `solve_dsm_truss_ebcs.m` to replace `solve_dsm_truss.m` that uses the direct stiffness method to solve for the nodal displacements and reaction forces of a truss structure with elastic boundary conditions. Carefully review `notation.m` for my recommendation on specifying the elastic boundary conditions. Starter code is provided on the course website in the Homework 3 code distribution. Be sure to *test your code*. Use your code to solve for the nodal displacements and reaction forces (including the force the spring exerts on node 2) of Truss 1. The Young's modulus times the cross-sectional area of each element are: $EA_e = e$ for $e = 1, \ldots, 5$ and the stiffness of the spring is $k = 1$.