Mechanics of Structures (CE130N)
Lab 4

1 Objective

The objective of this lab is for you to complete the truss program which you started in Lab 3. In this lab you will add boundary condition data to your program and set up the active equations and solve them.

2 Boundary Conditions

The relevant technical details of the imposition of boundary conditions on a truss are provided in the posted notes. In particular see §1.4.2 of those notes. The primary data that needs to be specified for each node in the truss is which degrees of freedom (dofs) are specified, are part of $\boldsymbol{u}_d$, and which nodes are subject to applied forces, even if zero, and thus are part of $\boldsymbol{u}_f$. Additionally, one needs to specify the values of applied forces and imposed displacements. To achieve this you will create three additional entries in the truss structure from Lab3.

1. `truss.bc` is a $2 \times n$ matrix. Each column corresponds to one node. The first entry is for the x-dof and the second for the y-dof. If a dof has a prescribed displacement, then the matrix entry is set to 1 and if the dof has a prescribed force (even if zero) the matrix entry is 0. Because in typical applications few dofs have prescribed displacements, the matrix is usually initialized to be all zeros and then 1s are inserted where needed.

2. `truss.u` is a $2 \times n$ matrix. Each column corresponds to one node. The first entry is for the x-displacement and the second for the y-displacement. If a dof has a prescribed displacement, then the matrix entry is set to the value of the imposed displacement. Only the values associated with boundary code 1 in `truss.bc` are used; the rest are ignored.

3. `truss.f` is a $2 \times n$ matrix. Each column corresponds to one node. The first entry is for the x-force and the second for the y-force. If a dof has a prescribed force, then the matrix entry is set to the value of the imposed force. Only the values associated with boundary code 0 in `truss.bc` are used; the rest are ignored.

3 Download

Download the three lab files from bspace. `plotmesh_truss_L4.m` and `plot_defo_truss.m` are plotting routines which you will use in your program. There is no need to edit them. `truss_student_L4.m` is the file you will need to edit. In this file there are several locations identified with the text `COMPLETE AS APPROPRIATE`. At these points you will need to provide the requisite code.

4 Walk Through

4.1 Setting the boundary condition codes

You will first need to set the boundary condition codes for the nodes. These will be stored in a $2 \times n$ matrix `truss.bc`. The first row of this matrix will contain the x-direction boundary codes of the nodes and the second row the y-direction boundary codes. Each column will thus represent the x, y-direction boundary codes of each node and there will be one column per node. Set up this matrix for the two bar example from lecture where nodes 1 and 3 will have boundary codes $(1, 1)^T$ and node 2 will have boundary codes $(0, 0)^T$.

4.2 Set the imposed displacements

You will next need to define the driven displacement values at the node/dof combinations where you set a value of 1 in `truss.bc`. The given file already zeros out the `truss.u` array so there is nothing more to do for the example truss from lecture. If non-zero values were specified as support motion, then one would have to set the corresponding numbers in this array.

4.3 Set the imposed forces

You will next need to define the imposed force values at the node/dof combinations where you set a value of 0 in `truss.bc`. The given file already zeros out the `truss.f` array so you will only need to set the forces on nodes with applied loads. For the lecture example there was a horizontal force at node 2. For the purposes of this part of the lab assignment set column 2 of `truss.f` to $(1, 0)^T$; i.e. a unit horizontal force.

4.4 Checkpoint

At this moment, comment out all the lines below this point, run your program and then call `plot_truss_L4(truss)`. This should produce a plot showing the appropriate supports and loads. Change node 1 to have a horizontal roller support, re-run, and re-plot. Make sure the plot is correct. If every thing is working, set the boundary codes back to being a pin support at nodes 1 and 3 and uncomment the lines which you commented out. Note that sometimes you will need to increase the size of the plot on your screen to distinguish between pins and rollers.

4.5 Stiffness matrix

The code to generate the stiffness matrix $\boldsymbol{K}$ is set up and should correspond to what you did in Lab 3. There is no need to edit this part of the code.

4.6 Extract the sub-matrices for the driven and free degrees of freedom

The degrees of freedom associated with the driven degrees of freedom are flagged by a 1 in `truss.bc`. To find the corresponding equation numbers one first needs to reshape the matrix into a $2n \times 1$ array. This is accomplished by using MATLAB's `reshape` command. The relevant equations numbers are then extracted using the `find` command. This sets up the needed index arrays `idd` and `idf`. This is already done for you in the code. The sub-matrix $\boldsymbol{K}_{ff}$, for example, is the extracted using the statement `Kff = K(idf,idf)`. Complete the statements to extract $\boldsymbol{K}_{dd}, \boldsymbol{K}_{fd},$ and $\boldsymbol{K}_{df}$.

4.7 Extract the forces on the free degrees of freedom

To extract the relevant forces, we first reshape the forces into a $2n \times 1$ vector and then use the `idf` index array to find the relevant values and store them in `Ff`. This is done for you already.

4.8 Solve for the free displacements

We will now solve for $\boldsymbol{u}_f$. To do so we first reshape the displacement array into a $2n \times 1$ matrix. The solution $\boldsymbol{u}_f = \boldsymbol{K}_{ff}^{-1}[\boldsymbol{F}_f - \boldsymbol{K}_{fd}\boldsymbol{u}_d]$ is then stored in `u(idf)`. The reshape is done for you in the code already. Complete the line that computes `u(idf)`.

4.9 Compute the unknown reaction forces

Evaluate the unknown reactions forces and store them in `F(idd)`. The relevant relation is $\boldsymbol{F}_d = \boldsymbol{K}_{df}\boldsymbol{u}_f + \boldsymbol{K}_{dd}\boldsymbol{u}_d$.

4.10 Reshape and store result in `truss`

One now reshapes the $2n \times 1$ vectors `u` and `F` to store the solution into `truss.u` and `truss.f`, respectively. This is already done for you in the code.

4.11 Plotting

To plot the deformed truss, you can call the function `plot_defo_truss(truss,scale)`, where `scale` is a magnification factor for plotting the displacements (which are often too small to be seen without magnification/scaling).

4.12 Test 1

Run your program on the two bar truss from lecture with node 1 at $(0, 0)$, node 2 at $(1, 1)$, and node 3 at $(1, 0)$. A scale factor or 1 is plenty for this example since $AE = 10$ and the load is 1. Does the plot look correct? Print out `truss.u` to see the nodal displacements. Print out `truss.f`. Are the forces correct? Note you can easily check them by hand for this problem since the system is statically determinate.

4.13 Test 2

Use your code to input the truss shown below. Assume the distances are given in milli-meters and that node 2 is pinned, node 5 has a vertical roller, and node 4 has a horizontal roller. Let each bar be 30 mm in diameter and be made of steel $E = 200\ \mathrm{kN/mm^2}$. At node 1 assume there is an imposed load of $-10000\boldsymbol{e}_y$ N. Find the displacement of node 1, the reaction force at node 2, and plot the deformed truss using an appropriate scale factor to make the motion of the truss clear.