Mechanics of Structures (CE130N)
Lab 4

1 Objective

The objective of this lab is to implement the techniques introduced to analyze statically determinate and indeterminate truss structures. When the size of the structure is small, i.e., the number of unknown quantities is small, one can solve the problems by hand, but as the size of the problem increases, the complexity of hand solutions grows exponentially. The systematic method introduced allows one to easily analyze large structures.

In this lab you will be asked to write some functions and procedures which will allow you to complete a program which analyzes general truss structures. The steps you will follow can be roughly organized as,

- Understand the data structure which describes the problem, including the truss structure and boundary conditions,
- Complete a function which computes the compatibility matrix of the structure $\mathbf{A}$, which relates the nodal displacements $\mathbf{u}$ with the element deformations $\mathbf{\Delta L}$,
- Complete functions which compute the matrices relating, resultant forces and bar stresses, bar stresses and bar strains, and bar strain and element deformation, which are necessary in forming the stiffness matrix,

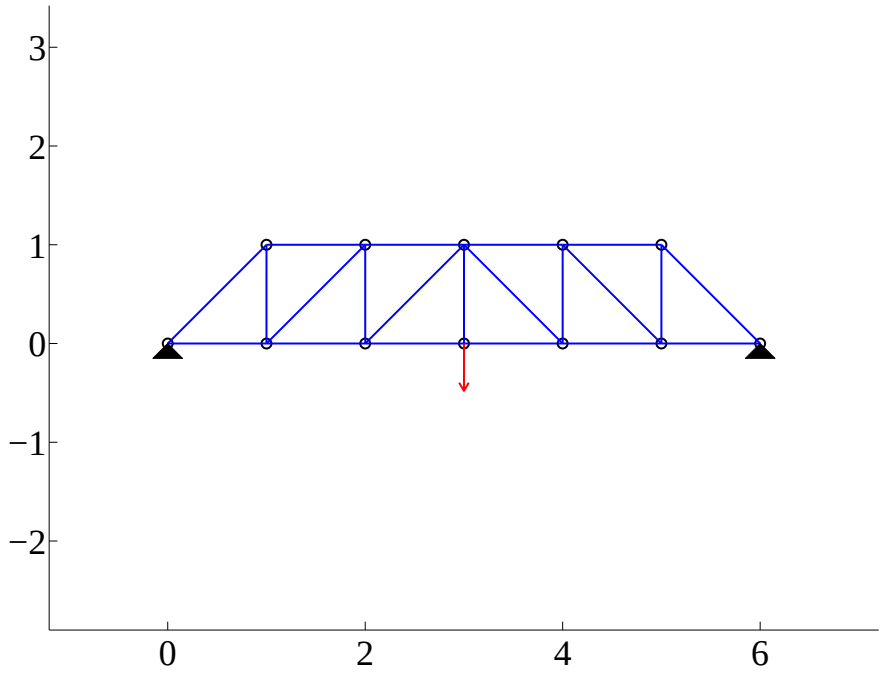

Figure 1: Howe truss

2 Truss structures

2.1 Governing equation

Let us define the following quantities,

$ndim$: Number of dimensions, for 2D(2), for 3D(3)
nnp : Number of node points
nel : Number of elements (bars)

The governing equations for a truss structure are the following,

- Equilibrium

$$\mathbf{F} = \mathbf{A}^T\mathbf{R} \tag{1}$$

Here $\mathbf{F} \in \mathbb{R}^{ndim\cdot nnp\times 1}$ is the vector of nodal forces, $\mathbf{R} \in \mathbb{R}^{nel\times 1}$ is the vector of bar forces, and $\mathbf{A} \in \mathbb{R}^{nel\times ndim\cdot nnp}$ is the compatibility matrix.

- Kinematics

$$\boldsymbol{\varepsilon} = [1/L]\boldsymbol{\Delta}\mathbf{L}, \tag{2}$$
$$\boldsymbol{\Delta}\mathbf{L} = \mathbf{A}\mathbf{u}, \tag{3}$$

combined yield,

$$\boldsymbol{\varepsilon} = [1/L]\mathbf{A}\mathbf{u}. \tag{4}$$

Here $\mathbf{u} \in \mathbb{R}^{ndim\cdot nnp\times 1}$ is the vector of nodal displacements, $\boldsymbol{\varepsilon} \in \mathbb{R}^{nel\times 1}$ is the vector of bar strains, $\boldsymbol{\Delta}\mathbf{L} \in \mathbb{R}^{nel\times 1}$ is the vector of bar deformations, and $[1/L] \in \mathbb{R}^{nel\times nel}$ is the diagonal matrix relating $\boldsymbol{\Delta}\mathbf{L}$ and $\boldsymbol{\varepsilon}$.

- Constitutive relation

$$\boldsymbol{\sigma} = [E]\boldsymbol{\varepsilon} \tag{5}$$

Here $\boldsymbol{\sigma} \in \mathbb{R}^{nel\times 1}$ is the vector of bar stresses and $[E] \in \mathbb{R}^{nel\times nel}$ is the diagonal matrix of Young's moduli.

- Resultant definition

$$\mathbf{R} = [A]\boldsymbol{\sigma} \tag{6}$$

Here $[A] \in \mathbb{R}^{nel\times nel}$ is the diagonal matrix of the area of the bar cross sections.

In the lecture these four relations have been combined to obtain a single equation representing equilibrium in terms of the vector of displacements $\mathbf{u}$,

$$\boxed{\mathbf{F} = \mathbf{K}\mathbf{u},} \tag{7}$$

where,

$$\begin{aligned}\mathbf{K} &:= \mathbf{A}^T[A][E][1/L]\mathbf{A},\\ &= \mathbf{A}^T[AE/L]\mathbf{A}\ .\end{aligned}$$

and $[AE/L]$ is a diagonal matrix representing the axial stiffness of the bars,

$$\mathbf{R} \quad = \quad [AE/L]\mathbf{L} \; .$$

2.2 Data structure

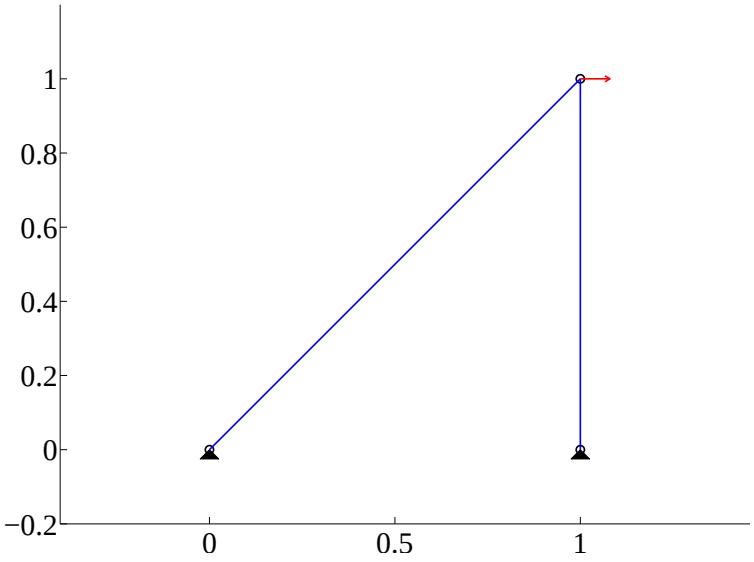

Figure 2: Truss example

To solve the truss problem numerically, one needs more than a picture. One requires an abstract representation of the truss structure along with the boundary conditions. Various representations are possible and below we present the data structure we will be using in our implementation. The data structure represents the truss shown in Figure 2. This is identical to the problem you have treated in the lecture.

2.3 Element-by-element assembly of the compatibility matrix $\mathbf{A}$

It has been introduced in the lecture that the $\mathbf{A}$ can be formed systematically. Consider the compatibility matrix of Figure 2,

$$\mathbf{A} = \begin{bmatrix} \mathbf{e}_{31}^T & \mathbf{0} & \mathbf{e}_{13}^T \\ \mathbf{0} & \mathbf{e}_{32}^T & \mathbf{e}_{23}^T \end{bmatrix},$$

which relates the bar deformations $\mathbf{\Delta L}$ with the nodal displacements $\mathbf{u}$,

$$\begin{aligned} \mathbf{\Delta L} &= \mathbf{A}\mathbf{u}, \\ \begin{bmatrix} \Delta L_1 \\ \Delta L_2 \end{bmatrix} &= \begin{bmatrix} \mathbf{e}_{31}^T & \mathbf{0} & \mathbf{e}_{13}^T \\ \mathbf{0} & \mathbf{e}_{32}^T & \mathbf{e}_{23}^T \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix}. \end{aligned}$$

The rows of $\mathbf{A}$ correspond to the bars, and the columns correspond to the nodes. Each bar only links to two nodes, so there are only two entries per row. In order to assemble $\mathbf{A}$, one can iterate through the bars starting from bar 1 to bar nel, inserting one row at a time.

In our implementation, you will be asked to complete a function,

```
function [Ae] = assembleAe_truss(mesh,ie)
```

where given the element number ie and `mesh` data structure, return $\mathbf{A}_e$, the element contribution to the compatibility matrix, defined as

$$\begin{aligned} \Delta L_{ie} &= \mathbf{A}_e \begin{bmatrix} \mathbf{u}_A \\ \mathbf{u}_B \end{bmatrix}, \\ \mathbf{A}_e &:= \begin{bmatrix} \mathbf{e}_{BA}^T, \mathbf{e}_{AB}^T \end{bmatrix}. \end{aligned}$$

Here is is assumed that the element connects to nodes A and B. Once you have the $\mathbf{A}_e$, you will have to insert it into the correct location of the compatibility matrix $\mathbf{A}$ in the function,

```
function [A] = assembleA(mesh)
```

3 Exercise

3.1 Download material and set initial path

1. Download the file `truss.zip` into your `ce130n/programs/` directory and unzip it.
2. Go to the `ce130n/programs/truss/exercise/` directory, and execute the file `init.m`. This will set the necessary paths to run the files.

YOU MUST RUN THE FILE init.m EVERYTIME YOU START UP MATLAB.

3.2 Understanding the data structure

In this exercise you will be asked to construct data structures which represents some truss structures to get familiar with the data structure.

1. Load the sample data structure and plot the truss structure.

   ```
   >> mesh_data_truss_example;
   >> plotmesh(mesh);
   ```

   The file `mesh_data_truss_example.m` contains the data structure for this truss stored in the variable `mesh`, and the function `plotmesh.m` plots the data structure `mesh`.

2. Construct data structures for the following two truss structures shown in the Figures 3 and 4.

   using the file `mesh_data_truss_example.m` as a starting point.

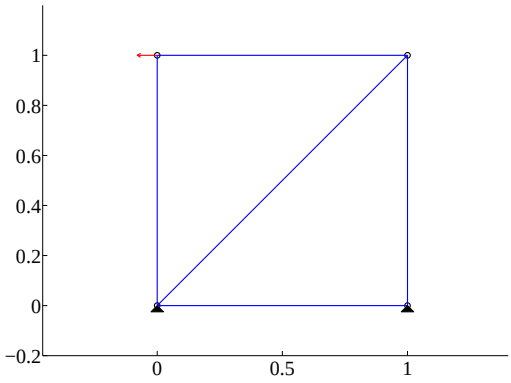

Figure 3: Truss Exercise 1

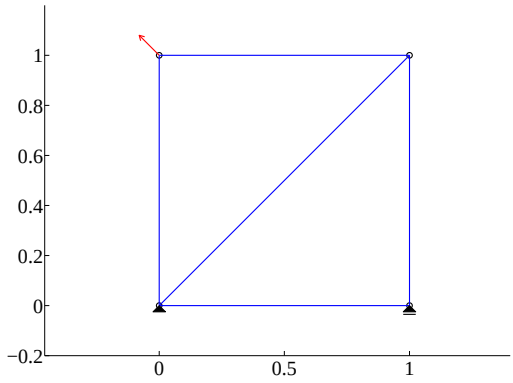

Figure 4: Truss Exercise 2

The boundary conditions are,

- Figure 3: The structure is pinned at both supports and has a load of $(-1, 0)$ at the top left node.
- Figure 4: The structure is pinned at the left support and roller at the right support, and has a load of $(-1, 1)$ at the top left node.

CHECKPOINT: Show the data structure for the two trusses and the generated plots. Explain how you made the structures satisfy the boundary conditions.

3.3 Construct compatibility matrix A

1. Complete the routine
   `ce130n/programs/truss/element/assembleAe_truss.m`. To test the function one must first load the mesh structure for the problem and then run the function,

```
>> mesh_data_truss_example;
>> [Ae] = assembleAe_truss(mesh,1);
```

   Use the data structure in Figure 2 to test your function. Should the function be complete and correct, `Ae` should give you the element `1`'s contribution to the compatibility matrix. Check this with the results in your lecture.

2. Complete the routine
   `ce130n/programs/truss/core/assembleA.m`. To test the function one must first load the mesh structure for the problem and then run the function,

```
>> mesh_data_truss_example;
>> [A] = assembleA(mesh);
```

   Use the data structure in Figure 2 to test your function. Should the function be complete and correct, `A` should give you the compatibility matrix. Check this with the results in your lecture.

   CHECKPOINT: Show the obtained compatibility matrix $\mathbf{A}$ and explain how you completed the code.

3.4 Construct diagonal matrices $[E], [A], [1/L]$

1. Complete the routines
   `ce130n/programs/truss/core/assembleKe.m`,
   `ce130n/programs/truss/core/assembleKa.m`,
   `ce130n/programs/truss/core/assembleKl.m`.
   These construct the $[E]$,$[A]$, and $[1/L]$ matrices.

   To test the function one must first load the mesh structure for the problem and then run the function,

```
>> mesh_data_truss_example;
>> [Kl] = assembleKl(mesh);
```

   Use the data structure in Figure 2 to test your function.

   CHECKPOINT: Explain what you had to do to make the code work.