# Lab 4

Let’s use our progress in 2D truss analysis to solve a real-world design problem. You have been contracted to design a structure that can support a **2kN load** cantilevered over a cliff by a distance of 8m so that it can be loaded onto a cargo ship passing below. A schematic is given below – all that is missing is your truss design.

Chart

Description automatically generated

In your final lab report, you must submit

* a diagram of your truss design with the boundary conditions clearly labeled
* a plot showing the deformed position of the truss, with each member plotted in a different color to indicate the level of axial stress in the member
* the vertical displacement of the 2kN load
* the total weight of your truss design

Finally, at the bottom of your report, compute the *product* of the 2kN load’s vertical displacement and the total weight of your design (units should be ). We will award a special prize to the student with the lowest value of this number.

# Notes / Rules:

1. Your truss must rest on the concrete foundation that’s already been poured on the cliff. No drilling into the cliff face below the concrete foundation and attaching truss members there – your construction workers would definitely call OSHA and throw a fit.
2. We have created another version of the *TrussGrabber2D* script, which is already pre-scaled for the design template shown on the last page. So, no need to enter a reference length this time around, just start designing.
3. You must use rectangular stock steel bars, but you get to decide the area of the stock cross section. Make sure you choose a cross section that prevents any member of your truss from exceeding the yield stress of steel! We will check to make sure all members are below yield. If you want to get fancy, you *are* allowed to assign cross sectional area on a per-member basis, but this is not required.

1. To make this problem more realistic, you *must* account for gravity loading on your truss, in addition to the application of the 2kN load. This seems like it might be tricky; it’s actually not so bad. If you know the length of a given member, you can calculate its weight by:

Once you know the weight of a member, you need to resolve that load into nodal forces. If we assume that the density and cross-sectional area are constant along the length of each member, that just means dividing the weight by two, and applying that load to the nodes at each end, in the Y direction.

You have been calculating the length of truss members in previous labs – here’s some code which does this for you using the information in the NC and CM matrices:

%Initialize a vector of element lengths

L = zeros(size(CM,1),1);

%Loop over the elements (each row in the CM matrix corresponds to an element)

for i = 1:size(CM,1)

%Extract the X coords and Y coords of the nodes in this element

X = NC(CM(i,:),1);

Y = NC(CM(i,:),2);

% Compute Length of this element

L(i) = sqrt((X(1)-X(2))^2+(Y(1)-Y(2))^2);

end

If you want to get really fancy, you can use some advanced Matlab functionality to perform these calculations. Beware though, fancy is not always better, especially for debugging!

% Matlab Jedi version of the above for loop

L = arrayfun(@(i) sqrt(sumsqr(diff(NC(CM(i,:),:)))), 1:size(CM,1))';

1. You *must* use at least 12 members in this truss. This means you’ll have (at least) 12 local stiffness matrices to assemble into your global matrix. So, you might want to start thinking of clever ways to assemble the stiffness matrix *programmatically*, using only the information contained in the connectivity matrix (CM) and nodal coordinate matrix (NC). Here’s some basics to get you started:

%Loop over the elements (each row in the CM matrix corresponds to an element)

for i = 1:size(CM,1)

%Extract the X coords and Y coords of the nodes in this element

X = NC(LM(i,:),1);

Y = NC(LM(i,:),2);

% Compute Length of this element (hint: we just talked about this part)

L(i) = ?

% Compute Theta of this element (hint: look up atan2 / atan2d)

Thetas(i) = ?

% Local Stiffness Matrix for this Element (hint: write the barstiffness func!)

k = (E\*A/L(i)) \* barstiffness(Thetas(i));

% Which global DOF does this element contain?

id = ?;

% Store local matrix in global matrix

K(id,id) = ?

end

1. In order to calculate the axial stress in each element, you’re going to have to use the linear elasticity formula that relates stress to strain:

It looks like a step backwards, but we can actually get the axial stress in each member pretty easily. You’ve already obtained the length of each element before any loads are applied (the L vector). Once you solve for the displacements at each node using the direct stiffness method, just add those displacement to the original nodal coordinates. This will give you the deformed position of each node! With this info, you can repeat the element -by-element length calculation that you did earlier, and get the deformed length of each member. Once you know the original length of a member, and the stretched length, you should be able to calculate strain...

1. The plotting stuff can be a bit tricky. Go ahead and use this this code for generating the plot that superimposes the deformed truss with stress contours on top of the original, undeformed truss.

%% Plot undeformed and deformed mesh

SF = 20; %scale factor for visualization of deformed results

% This is an easy way to add find the new nodal positions after loads are applied, assuming you have the original nodal coordinates NC as well as a global displacement vector D, which has dimensions [2\*numNodes , 1]

NC\_2 = [NC(:,1)+SF\*D(1:2:end) NC(:,2)+SF\*D(2:2:end)];

% For this part to work, you will have to compute a column vector called “Stress,” which contains the stress of each element. The dimensions of this vector will be [numElements , 1]. You will also have to have the function “redblue.m” in your working directory!

Contours = 100;

Scolor = floor(rescale(Stress,1,Contours));

C = redblue(Contours);

colormap(redblue(Contours));

% Loop over the elements

figure(1),clf

for i = 1:nEl

%Extract the original X coords and Y coords of the nodes in this element

X0 = NC(LM(i,:),1);

Y0 = NC(LM(i,:),2);

%plot undeformed elements

plot(X0,Y0,'k','linewidth',1.5);

hold on

%Extract the original X coords and Y coords of the nodes in this element

X2 = NC\_2(LM(i,:),1);

Y2 = NC\_2(LM(i,:),2);

%plot deformed elements

plot(X2,Y2,'linewidth',2,'color',C(Scolor(i),:));

end

axis equal

colorbar

caxis([-max(-min(S),max(S)), max(-min(S),max(S))])

title(sprintf('Deformed Results, S11 in MPa, ScaleFactor = %.0f',SF))