CVIE 546 Structural Design Optimization

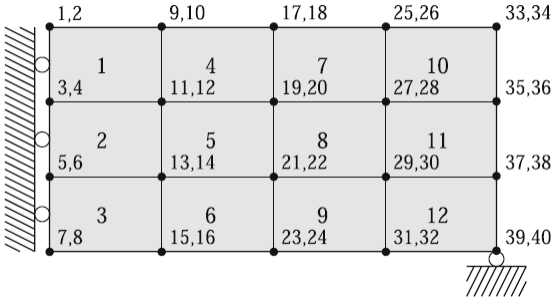
Tutorial on 88-line Topology Optimization

Ref: Andreassen, E., Clausen, A., Schevenels, M., Lazarov, B. S., & Sigmund, O. (2011). Efficient topology optimization in MATLAB using 88 lines of code. Structural and Multidisciplinary Optimization, 43, 1-16.

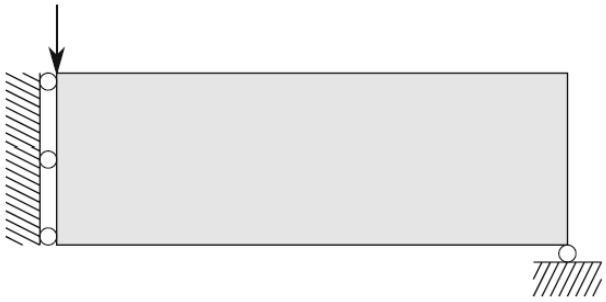
Background (Messerschmitt–Bölkow–Blohm, MBB, beam**)**

Element and node numbering from top to bottom, left to right.

Singular number: horizontal DOF, Plural number: vertical DOF.



Example mesh



Default design domain (half MBB beam)

Suggestion:

Create a copy of the original MATLAB file for each task/modification.

Use the help function to learn about the MATLAB functions.

Task 1: Change the inputs

The following figure shows the optimization results from the default settings with the following command:

top88(60,20,0.5,3,1.5,1)

Please try to change volume fraction (e.g., volfrac=0.2, 0.7), penalty (e.g., penal=1, 2, 5), filter size (e.g., rmin=0.5, 3) , filter type (e.g., ft=2) and see how these change affect your optimization results.

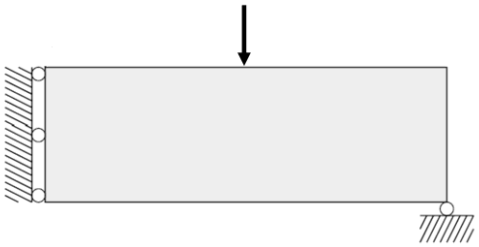
Optimization results from default settings

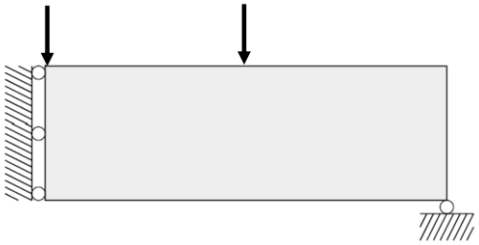
Task 2: Change the loading

Let's try to change the loading from the center to the middle of the half span.

Hint: modify Line 19 F vector

Extension: (1) let's try to add two loading points (both center and middle of the half span). (2) change the ratio of these two loads and see the impacts. *Further thoughts: distributed loads?*

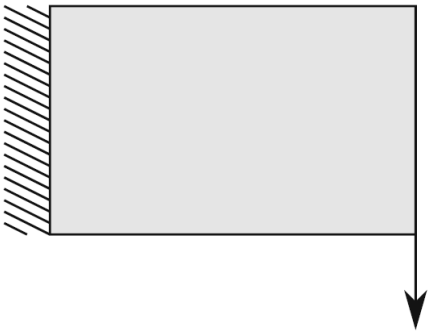




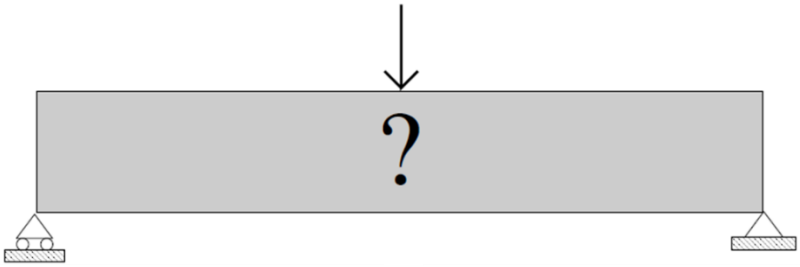
Task 3: Change the boundary conditions

Let's try to change the entire setting to a cantilever beam.

Hint: modify Line 19 F vector AND Line 21 fixeddofs vector



Extension: Try change the entire setting to the full MBB beam. *Further thoughts: distributed loads along the entire span? Additional support at middle from simply supported to continuous beam?*



Task 4: Consider multiple load cases

Let's consider two load cases for the cantilever beam.

In the case of two load cases, the force and displacement vectors must be defined as two-column vectors, which means that lines 19 and 20 are changed to:

```
F = sparse([2*(nely+1)*nelx+2, 2*(nely+1)*(nelx+1)], [1 2],[1 -1],2*(nely+1)*(nelx+1),2);
U = zeros(2*(nely+1)*(nelx+1),2);
```

The support conditions (line 21) are defined in the same way as in the previous subsection. The equilibrium equations must be solved for both load cases, which is accomplished by changing line 56 as follows:

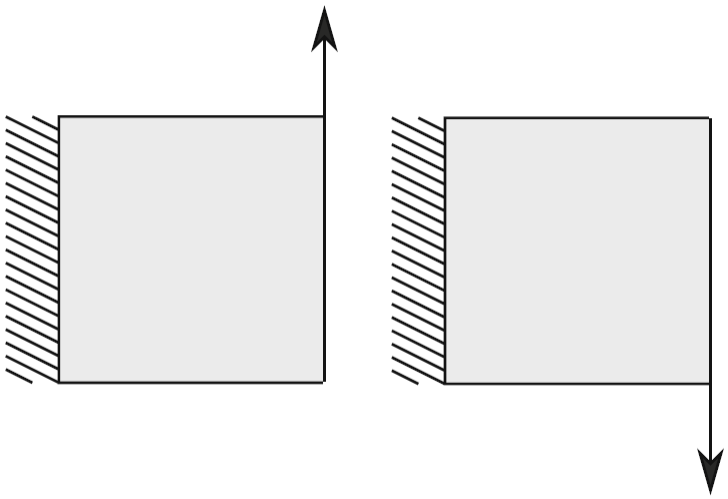
```
U(freedofs,:) = K(freedofs,freedofs)\F(freedofs,:);
```

The objective function is now defined as the sum of two compliances:

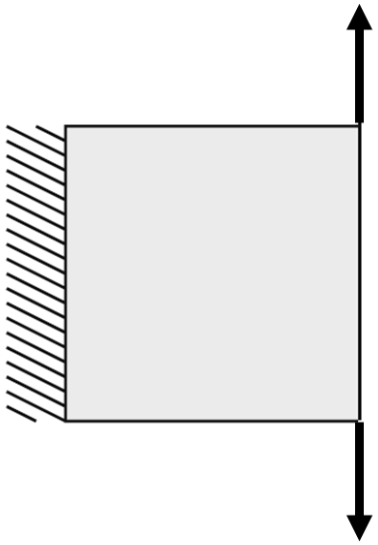
$$c(\mathbf{x}) = \sum_{i=1}^{2} \mathbf{U}_i^{\mathrm{T}} \mathbf{K} \mathbf{U}_i$$

Lines 58–60 are thus replaced with the following code:

```
c=0;
dc=0;
for i = 1:size(F,2)
Ui = U(:,i);
ce = reshape(sum((Ui(edofMat)*KE).*Ui(edofMat),2), nely,nelx);
c = c + sum(sum((Emin+xPhys.^penal*(E0-Emin)).*ce));
dc = dc - penal*(E0-Emin)*xPhys.^(penal-1).*ce;
end
```



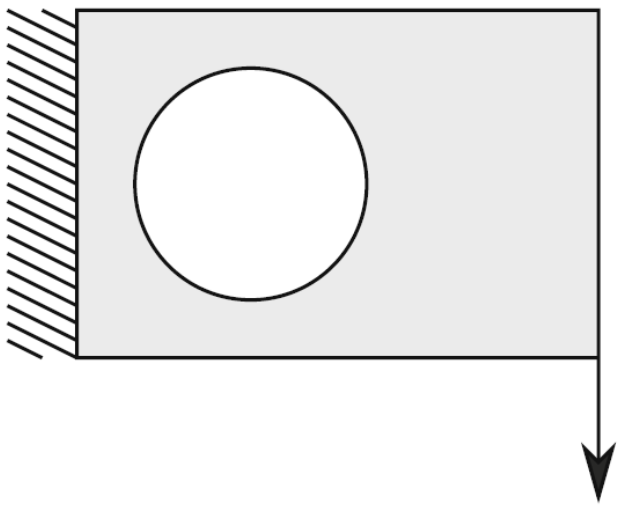
Extension: Try compare with the topology optimization results that consider two loads in the same load case.



Task 5: Passive elements

In some cases, certain areas of the design domain may be required to be void or solid (e.g. to allow for the passage of a pipe or to support a secondary structure). This can be easily accomplished by defining elements with a density fixed to be zero or one.

Let's set a circular region of the design domain with radius nely/3 and center (nely/2, nelx/3) is fixed to be void.



In order to distinguish between active and passive elements, a nely ×nelx matrix passive is defined with 0 at elements free to change, 1 at elements fixed to be void, and 2 at elements fixed to be solid:

```
passive = zeros(nely,nelx);
for i = 1:nelx
for j = 1:nely
if sqrt((j-nely/2)^2+(i-nelx/3)^2)< nely/3
passive(j,i) = 1;
end
end
end
```

These lines must be inserted in the 88 line code before the start of the optimization loop. The optimality criteria method must be modified by adding the following code between lines 78 and 79:

```
xPhys(passive==1) = 0;
xPhys(passive==2) = 1;
```

Extension: Let's try to define uniform distributed loading and top surface as solid. This can be imagined as traffic on a bridge deck, which must be solid.