Exercise 9: Topology optimization

Hands-on introduction to topology optimization

The following exercises are intended to give some practical experience with topology optimization algorithms. They are based on an educational compliance minimization topology optimization code freely available on the internet [1].

Assignment

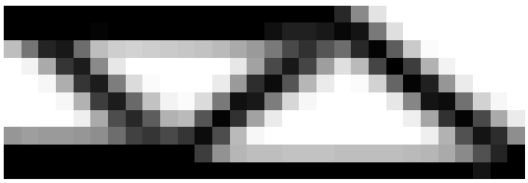
Open '`top.m`' in the Matlab editor, and have a look at the structure of the program, as indicated by the comments. See if you can recognize the main steps in the density-based topology optimization procedure.
A more in-depth description can be found in [2], which is also available on Blackboard under Course Materials.

Also look at the input arguments of `top.m`:

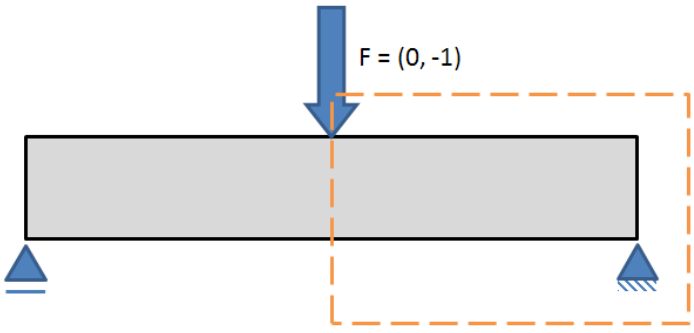
```
function top(nelx,nely,volfrac,penal,rmin);
```

`nelx, nely`:	nr of elements in x and y direction
`volfrac`:	volume fraction [0–1], the amount of material to use
`penal`:	penalization exponent [1–5], controlling the amount of intermediate material (gray)
`rmin`:	filtering radius, determining the feature size

Start Matlab, change the working directory to where you have saved `top.m`, and type the command
`>> top(30, 10, 0.5, 3, 1.5)` (+ hit Enter)
to see if everything works correctly. You should see a topology optimization process that results, after 72 iterations, in a structure like this:



This is the right half (orange box below) of a symmetric design problem to find the stiffest structure (minimum compliance) for this situation, where 50% of the domain can be filled with material (black):



Copy your result, and the rest that follows below, into Powerpoint.

9.1 Influence of mesh refinement

a) To study the effect of mesh refinement on the final design, vary `nelx` and `nely` and leave the other parameters to their values used before. Make sure to keep them in a 3:1 ratio. If you increased the mesh too much, and the program takes too long to complete, interrupt the process by pressing Ctrl-C and try again with smaller numbers. Collect a couple of results (3—4) in a table:

Mesh used	Obtained design	Iterations	Compliance (objective)
<nelx, nely>	<Copy-paste result image>	<nr of iterations>	<final obj. value>

b) What are your conclusions?

9.2 Influence of penalization exponent

a) Next, choose a reasonable mesh size with some detail but which does not take too long to optimize, and make modifications to the penalization power `penal` (in the range 1–5). Make a table as the one below for a few penalization values, e.g. 1, 2 and 3:

Penalization power	Obtained design	Iterations	Compliance (objective)
<value>	<Copy-paste result image>	<nr of iterations>	<final obj. value>

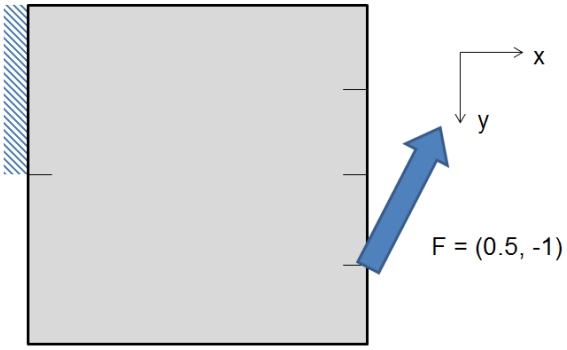
b) Based on these observations, how can you summarize the influence of the penalization power? What value would you recommend?

If you like, play around with the volume fraction [0–1] and the filter radius [0.5–5] as well. Include a picture of the result in your report.

9.3 Changing the problem definition

Topology optimization can be used for all sorts of problem configurations. As an illustration, you will change the applied load in this exercise.

a) We are going to model the following configuration:



(note the somewhat unusual coordinate system used)

Go to line 79 in `top.m`, `% DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)`
To change the load into a load in the middle of the right edge of the domain, change the load definition into:

```
% DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
%F(2,1) = -1;
node = (nelx+1)*(nely+1) - round(0.25*(nely+1));
F(2*node-1,1) = 0.5; % x-direction load
F(2*node,1)   = -1;  % y-direction load
```

To change the boundary conditions, change the lines below the load definition into:

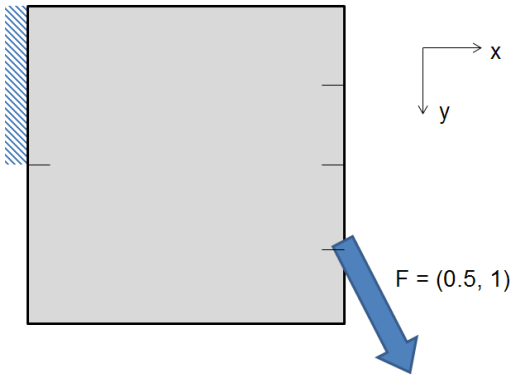
```
%fixeddofs   = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
fixeddofs   = 1:(2*round((nely+1)/2));
```

The rest remains as before, so the total problem definition section now should look like this:

```
% DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
%F(2,1) = -1;
node = (nelx+1)*(nely+1) - round(0.25*(nely+1));
F(2*node-1,1) = 0.5; % x-direction load
F(2*node,1)   = -1;  % y-direction load
%fixeddofs   = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
fixeddofs   = 1:(2*round((nely+1)/2));
alldofs     = [1:2*(nely+1)*(nelx+1)];
freedofs    = setdiff(alldofs,fixeddofs);
```

Explanation (in case you want to know what this is all about, see also [2]):
Degrees of freedom (DOFs) are arranged in `top.m` starting in the left upper corner, increasing first vertically, then horizontally. The number of nodes in vertical direction is (`nely+1`), and in horizontal direction (`nelx+1`), i.e. one more than the number of elements. Each node has 2 degrees of freedom (u+v), so the number of DOFs in vertical direction is 2(`nely+1`), etc.
The first line (node = …) defines that the node where the load is applied lies at quarter of the vertical nodes away from the final node (nelx+1)*(nely+1). The lines below that define the x- and y-component of load vector F, where the node number is multiplied by 2 to get the DOF index.
For the boundary conditions, the fixed DOFs are set to be those of the first to the half-way node on the left-hand side of the domain. This half-way nodenumber is given by round((nely+1)/2).

b) Choose a nice mesh size with `nelx = nely` (square domain) and use `penal` = 3. Run the optimization and collect the result.
c) Change the loads (x, y or both) into different values and observe the effect this has on the design. For examle, try this one:



Do the shapes that are generated make some sense?

Feel free to try this problem also with non-square domains, or to make other variations in the loads or boundary conditions. For a faster version of this program, check out [3]: this version allows you to solve bigger problems, that give more detailed solutions.

References

[1] http://www.topopt.dtu.dk of DTU, the Technical University of Denmark. From the 'Applets and Software' menu, choose 'MATLAB program' (`top.m`).
[2] Ole Sigmund (2001). *A 99 line topology optimization code written in Matlab*, Structural & Multidisciplinary Optimization 21(2), 120–127, http://dx.doi.org/10.1007/s001580050176.
[3] http://www.topopt.dtu.dk of DTU, the Technical University of Denmark. From the 'Applets and Software' menu, choose 'New MATLAB program' (`top88.m`).