Academic year 2020-2021

Structural & Multidisciplinary Optimization
Computer Project: Building your Topology Optimization tool

Optional Computer Assignment #3

1. Introduction

The following exercises are based on "*A 99-line topology optimization code written in Matlab*" (Sigmund, 2001). Both the code and a preprint of the paper can be downloaded from the web site www.topopt.dtu.dk or from the course web page www.ingveh.ulg.ac.be. The exercises require some basic knowledge of structural mechanics, finite element analysis and optimization theory.

You will be asked to elaborate your own topology optimization tool by adding several features to the basic code provided initially. The different steps of this process will be carried out through a sequence of improvement steps. At the end of the course you will be able to solve design problems using your personal code.

The accompanying paper (Sigmund, 2001) will give a lot of hints to solve the first part of the project. Other important information can be found in the Appendices.

2. Getting started

- Download the proposed starting code top.m from web site and read carefully the reference papers Sigmund (2001)
- Run the default MBB-beam example by writing top(60,20,0.5,3.0,1.1) in MATLAB command line.
- Start experimenting with the code.
- Keep an original version of the MATLAB code and start editing new versions clearly identified.

3. Experiment with your first topology optimization code

The basic code that you got solves the problem of optimizing the material distribution in the case of the MBB-beam such that its compliance is minimized. It is requested to investigate and develop several extensions of the code to extend the scope of the basic tool.

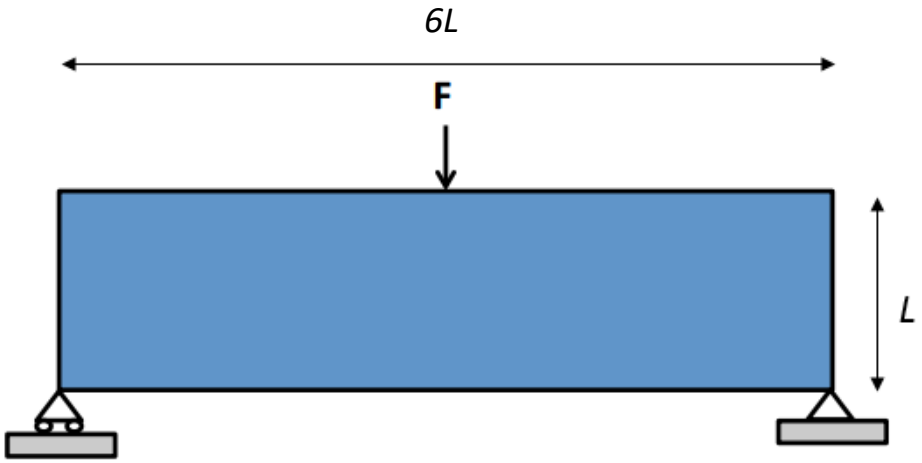

Fig 1 : MBB-beam problem : geometry and boundary conditions

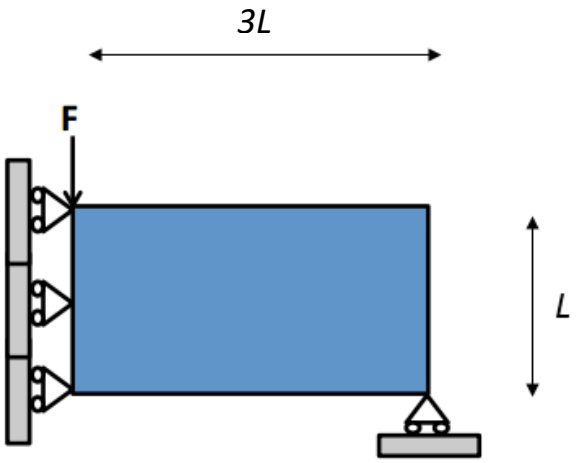

Fig 2 : MBB-beam problem : design model accounting for symetry conditions

3.1 Test the influence of the numerical parameters

Use the original MATLAB code to investigate the influence of the FE discretization (nelx*nely) upon the optimal topology. Perform also an investigation of

- The filter size r_{min} = 1.6, 2.0, 4.0
- The penalization parameter to assess their influence over the optimal layout. Test p=1.0, 1.6, 2.0, 3.0, 4.0, 8.0

- The impact of the bound over the volume V=15%, 30%, 45%, 60%
- The minimum gauge of the density on the optimal compliance value. Xmin = 0.1, 0.01, 0.001

For your test, solve the optimization problem for a constant value of the penalization parameter and for a variable value. Discuss the results.

3.2 Implement other boundary and load conditions

Change the boundary conditions (support and loading) in order to be able to solve different problems. Try for instance the famous following example, called Michel beam problem

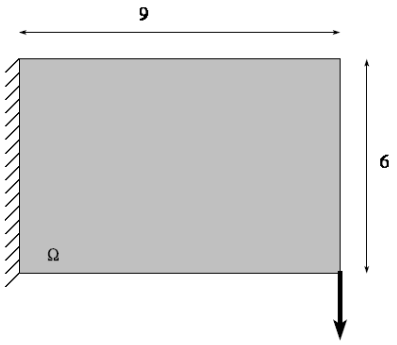

Fig 3 : Michel cantilever beam problem

Study the impact of other support conditions upon the optimal layout. Investigate the following issues too:

- Does the magnitude of the loading influence the design?
- How the support conditions modify the layout?
- Does the direction of the forces change the design?

3.3 Try to solve other design problems

Generalize your code by finding a way to define easily other geometries. To validate your developments, select a couple of problems out of the ones that are suggested below and solve them using topology optimization:

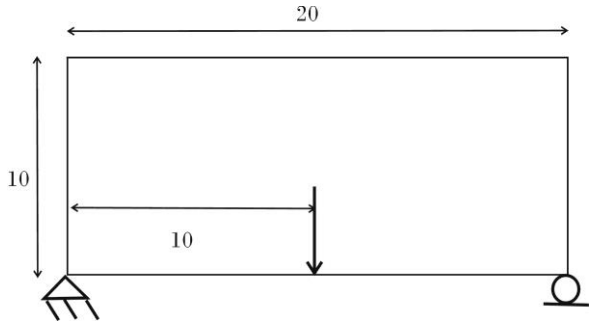

Fig 4 : Bridge problem 1. Point load. One end supported. One end clamped.

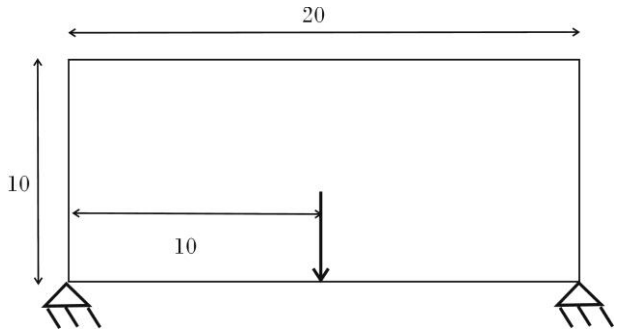

Fig 5 : Bridge problem 2. Point load. Both ends are clamped.

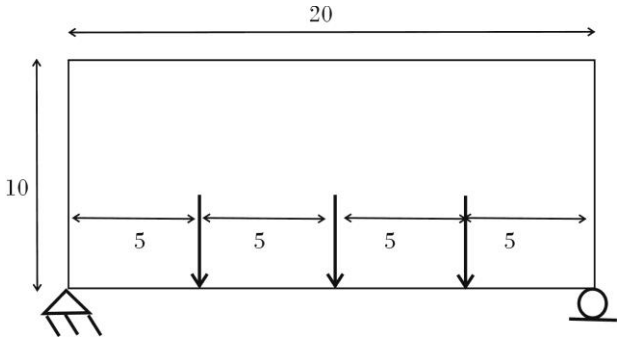

Fig 6 : Bridge problem 3. 3 point loads applied simultaneously.
One end supported. One end clamped.

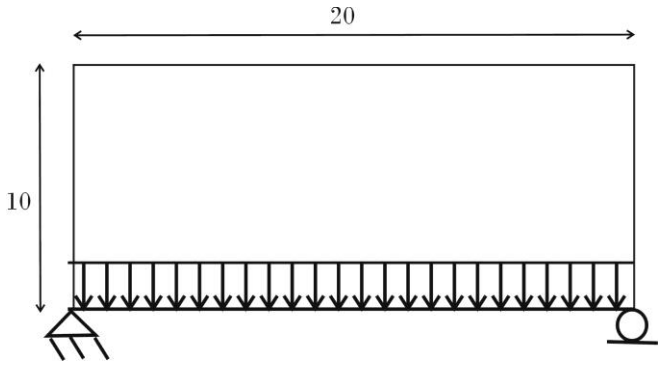

Fig 7 : Bridge problem 4. Distributed load. One end supported. One end clamped.

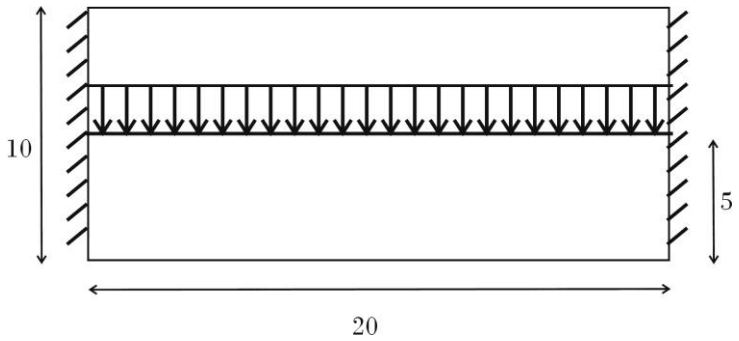

Fig 8 : Bridge problem 5. Distributed load at mid elevation. One end supported. Clamped boundaries

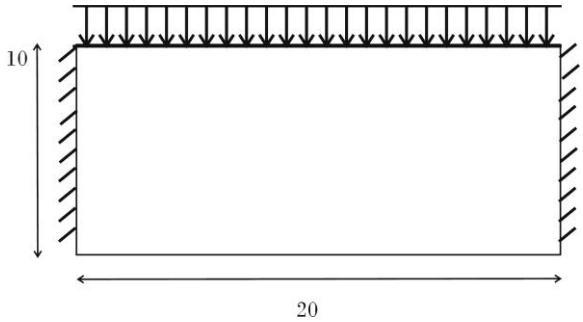

Fig 9 : Bridge problem 6. Distributed load at top elevation. Clamped boundaries.

3.4 Implement passive elements

In some applications, it is usual to have zones in which one has to enforce either full material or void (holes). Modify your code to account for zones prescribed to take the minimum density (nearly zero) and other ones in which the density is one. Compare the optimal compliance once you have introduced such zones in the design domain.

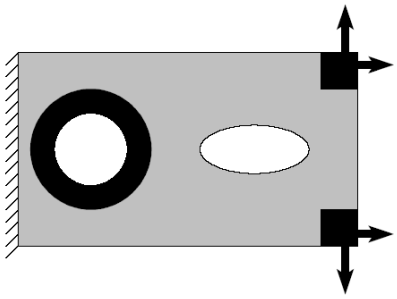

Fig. 10: Prescribed zones with void and full material densities

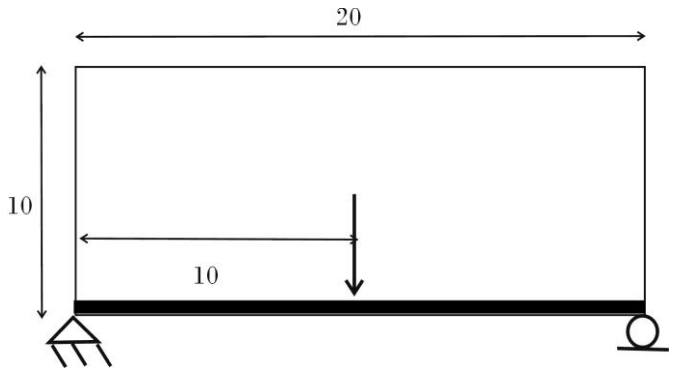

Fig 11 : Bridge problem 7. Point load. Impose two layers of elements to enforce the presence of the road. One end supported. One end clamped.

4. Deadline and milestones

The HW3 conducts to bonus points for the final mark of the project part.

Solving partially the HW3 is allowed. Marks will be given for the parts of the assignment that are correctly achieved.

Reports must be written using a word processor (MS Word or Latex). Reports are bounded to max 20 pages. Appendices (not counted in the 20 pages) include all computer codes and additional material that might be necessary.

Reports have to be in pdf format and should entitled Name_Firstname_HW3.pdf.

The report and its attachments for HW3 of Topology optimization have to be posted by emails to Pierre Duysinx (p.duysinx@uliege.be), to Pablo Alarcon (palarcon@uliege.be), and to Denis Trillet (dtrillet@uliege.be) by December 24, 2020 at 12:00 CET.

5. References

Sigmund 0. (2001). A 99-line topology optimization code written in MATLAB. Structural and Multidisciplinary Optimization, 21, 120-127.

A. APPENDICES

A.1 Plotting displacements

Insert the following lines in your program instead of the current plotting line:

```
% colormap(gray); imagesc(-x); axis equal; axis tight; axis off; pause(1e-6);
colormap(gray); axis equal;
for ely = 1:nely
  for elx = 1:nelx
    n1 = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx +ely;
    Ue = 0.005*U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2; 2*n1+1;2*n1+2],1);
    ly = ely-1; lx = elx-1;
    xx = [Ue(1,1)+lx Ue(3,1)+lx+1 Ue(5,1)+lx+1 Ue(7,1)+lx ]';
    yy = [-Ue(2,1)-ly -Ue(4,1)-ly -Ue(6,1)-ly-1 -Ue(8,1)-ly-1]';
    patch(xx,yy,-x(ely,elx))
  end
end
drawnow; clf;
```

A.2 Efficient matrix assembly for large problems

To handle large problems, you may substitute the transparent and easy-to-read stiffness matrix assembly:

```
K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
F = sparse(2*(nely+1)*(nelx+1),1); U = zeros(2*(nely+1)*(nelx+1),1);
for elx = 1:nelx
  for ely = 1:nely
    n1 = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx +ely;
    edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
    K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
  end
end
```

with the much more efficient assembly using a list of triplets:

```
I = zeros(nelx*nely*64,1); J = zeros(nelx*nely*64,1); X = zeros(nelx*nely*64,1);
F = sparse(2*(nely+1)*(nelx+1),1); U = zeros(2*(nely+1)*(nelx+1),1);
ntriplets = 0;
for elx = 1:nelx
  for ely = 1:nely
    n1 = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx +ely;
    edof = [2*n1-1 2*n1 2*n2-1 2*n2 2*n2+1 2*n2+2 2*n1+1 2*n1+2];
```

```
    xval = x(ely,elx)^penal;
    for krow = 1:8
      for kcol = 1:8
        ntriplets = ntriplets+1;
        I(ntriplets) = edof(krow);
        J(ntriplets) = edof(kcol);
        X(ntriplets) = xval*KE(krow,kcol);
      end
    end
  end
end
K = sparse(I,J,X,2*(nelx+1)*(nely+1),2*(nelx+1)*(nely+1));
```

A.3 The strain displacement matrix

```
bmat = [-1/2 0 1/2 0 1/2 0 -1/2 0
  0 -1/2 0 -1/2 0 1/2 0 1/2
  -1/2 -1/2 -1/2 1/2 1/2 1/2 1/2 -1/2];
```

A.4 The constitutive matrix for plane stress

```
Emat = E/(1-nu^2)*[ 1 nu 0
  nu 1 0
  0 0 (1-nu)/2];
```

A.5 The mass matrix

```
m0 = [4/9 0 2/9 0 1/9 0 2/9 0
  0 4/9 0 2/9 0 1/9 0 2/9
  2/9 0 4/9 0 2/9 0 1/9 0
  0 2/9 0 4/9 0 2/9 0 1/9
  1/9 0 2/9 0 4/9 0 2/9 0
  0 1/9 0 2/9 0 4/9 0 2/9
  2/9 0 1/9 0 2/9 0 4/9 0
  0 2/9 0 1/9 0 2/9 0 4/9]/4/(nelx*nely);
```

A.6 The thermal stiffness matrix

```
KE = [2/3 -1/6 -1/3 -1/6
     -1/6 2/3 -1/6 -1/3
     -1/3 -1/6 2/3 -1/6
     -1/6 -1/3 -1/6 2/3
     ];
```